ZigBee RF4CE

ZigBee Input Device (ZID) Standard

Version 1.0

ZigBee Document 105557r18ZB	
October 24th, 2012	
Sponsored by: ZigBee Alliance	
Accepted by	This document has been accepted for release by the ZigBee Alliance Board of Directors
Abstract	This specification defines the protocol infrastructures and services available to applications operating on the ZigBee RF4CE platform using the ZID profile.
Keywords	ZigBee, RF4CE, ZigBee Input Device, ZID, HID, pointing, profile, multitouch

October 24th, 2012

This page is intentionally blank

Notice of use and disclosure

The ZigBee Specification is available to individuals, companies and institutions free of charge for all non-commercial purposes (including university research, technical evaluation, and development of non-commercial software, tools, or documentation). No part of this specification may be used in development of a product for sale without becoming a member of ZigBee Alliance.

Elements of ZigBee Alliance specifications may be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party may or may not be a member of ZigBee). ZigBee is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.

This document and the information contained herein are provided on an "AS IS" basis and ZigBee DISCLAIMS ALL WARRANTIES EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO (A) ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OF THIRD PARTIES (INCLUDING WITHOUT LIMITATION ANY INTELLECTUAL PROPERTY RIGHTS INCLUDING PATENT, COPYRIGHT OR TRADEMARK RIGHTS) OR (B) ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NONINFRINGEMENT. IN NO EVENT WILL ZIGBEE BE LIABLE FOR ANY LOSS OF PROFITS, LOSS OF BUSINESS, LOSS OF USE OF DATA, INTERRUPTION OF BUSINESS, OR FOR ANY OTHER DIRECT, INDIRECT, SPECIAL OR EXEMPLARY, INCIDENTIAL, PUNITIVE OR CONSEQUENTIAL DAMAGES OF ANY KIND, IN CONTRACT OR IN TORT, IN CONNECTION WITH THIS DOCUMENT OR THE INFORMATION CONTAINED HEREIN, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE. All Company, brand and product names may be trademarks that are the sole property of their respective owners.

The above notice and this paragraph must be included on all copies of this document that are made.

ZigBee Alliance, Inc.

2400 Camino Ramon, Suite 375

San Ramon, CA 94583

Participants

When this document was release, the RF4CE SIG Technical Working Group leadership was composed of the following members:

Michael Sallis: *Chair*

Joseph Reddy: *Vice chair*

Nick Shepherd *Technical Editor*

Contributions were made to this document by the following members:

Sorin Aliciuc
Victor Berrios
Bram Van den Bosch
Francesco Cimino
Jan Van Eetvelde
Chris Gray
Chris Heiny
Phil Jamieson
Paul Jeon
Razvan Mihai Lucaci
Yuichi Morioka
Masahiro Nakano
Kundok Park
Stefano Pascali
Nicu Penisoara
Bill Reams
Joseph Reddy
Michele Sardo
Koichi Sato
Stig Torud
Udo Walter

Table of Contents

List of Figures

List of Tables

1 Introduction

This document specifies the ZigBee RF4CE profile 0x02: ZigBee Input Device (ZID) which interfaces to the ZigBee RF4CE network layer (see [R1]). The ZID profile defines commands and procedures to enable devices such as mice, touchpads, keyboards, wands, Riva wheels or RC pointers.

1.1 Scope

The RF4CE ZID profile defines the over-air commands and mechanisms required to allow a Human Interface Device (HID) class device to communicate with a host. The profile defines two types of device: a HID class device and a HID adaptor. The HID class device can be used to support devices such as keyboards, mice or touchpads and the HID adaptor can be used to communicate through a HID class driver to some PC or other embedded host. The RF4CE ZID profile does not define any host side behavior beyond the HID adaptor.

1.2 Definitions

Controller	A PAN participant that has ZigBee RF4CE functionality.
Device	An object that has IEEE 802.15.4 functionality.
HID adaptor	A node, supporting host side functionality, from which a host can be constructed.
HID class device	A node, supporting HID side functionality, from which a human interface device can be constructed.
Host	An object implementing a HID class driver which parses and interprets HID reports.
Node	A device that has ZigBee RF4CE and ZID profile functionality.
Originator	The node from which an RF4CE ZID profile transmission is sent.
Pair	A logical association between two nodes.
PAN	A personal area network as defined in IEEE 802.15.4.
PAN coordinator	A device that can create its own PAN.
PAN participant	A device that can join a PAN.
RC network	Multiple, interoperating, RC PANs.
RC PAN	A PAN consisting exclusively of ZigBee RF4CE nodes.
Recipient	The node to which an RF4CE ZID profile transmission is directed.
Target	A PAN coordinator that has ZigBee RF4CE functionality.

1.3 Conformance levels

The following words, used throughout this document, have specific meanings:

May	A key word indicating a course of action permissible within the limits of the standard (*may* equals *is permitted*).
Shall	A key word indicating mandatory requirements to be strictly followed in order to conform to the standard; deviations from shall are prohibited (*shall* equals *is required to*).
Should	A key word indicating that, among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others; that a certain course of action is preferred but not necessarily required; or, that (in the negative form) a certain course of action is deprecated but not prohibited (*should* equals *is recommended that*).

1.4 Abbreviations

CE	Consumer electronics
GDP	Generic device profile
HID	Human interface device
ID	Identifier
IEEE	Institute of electrical and electronic engineers
LQI	Link quality indicator
MAC	Medium access control
NIB	Network information base
NLDE	Network layer data entity
NLME	Network layer management entity
NWK	Network
ORG	Originator
PAN	Personal area network
PHY	Physical
POS	Personal operating space
RC	Remote control
REC	Recipient
RF	Radio frequency
RF4CE	Radio frequency for consumer electronics
SAP	Service access point
USB-IF	Universal serial bus implementers forum
ZID	ZigBee input device

1.5 Conventions

1.5.1 Number formats

In this specification hexadecimal numbers are prefixed with the designation “0x” and binary numbers are prefixed with the designation “0b”. All other numbers are assumed to be decimal.

1.5.2 Transmission order

The frames in this specification are described as a sequence of fields in a specific order. All frame formats are depicted in the order in which they are transmitted by the PHY, from left to right, where the leftmost bit is transmitted first in time. Bits within each field are numbered from 0 (leftmost and least significant) to k-1 (rightmost and most significant), where the length of the field is k bits. Fields that are longer than a single octet are sent to the MAC in the order from the octet containing the lowest numbered bits to the octet containing the highest numbered bits.

1.5.3 Message sequence charts

During this specification, message sequence charts are used to illustrate the flow of various operations. Instances are labeled with the layer (APL for the application or NWK for the network) followed by the

node type (ORG for the originator or REC for the recipient). Primitives are shown in normal style but, for simplicity, without the entity prefix (i.e. NLDE or NLME), e.g. "NLME-PAIR.response" becomes "PAIR.response". Over the air command frames are labeled in italic text.

1.5.4 Reserved values

Unless otherwise specified, all reserved fields appearing in a frame structure shall be set to zero on transmission and ignored upon reception. Reserved values appearing in multi-value fields (c.f. Table 2) shall not be used.

1.6 References[1]

[R1] ZigBee RF4CE Specification, ZigBee Alliance document 094945, Version 1.0.1, November, 2010.

[R2] Universal Serial Bus, Device Class Definition for Human Interface Devices (HID), Version 1.11, June 27th, 2001.

[R3] IEEE Standard 802.15.4-2006, IEEE Standard Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), IEEE Computer Society, September 2006.

[R4] ZigBee RF4CE: Device Type List, ZigBee Alliance document 094950.

[R5] ZigBee RF4CE Generic Device Profile Specification, ZigBee Alliance document 11xxx, Version 1.0.0, March 2011.

[R6] Certification Status of ZID and GDP Commands and Features, ZigBee Alliance document 120425.

[1] The version and date information in these references was correct at the time this document was released.

2 General description

The RF4CE ZID profile defines the over-air commands and mechanisms required to allow a HID class device to communicate with a host. The profile defines two types of device: a HID class device and a HID adaptor. The HID class device can be used to support devices such as keyboards, mice or touchpads and the HID adaptor can be used to communicate through a HID class driver to some PC or other embedded host. A HID class device would be realized by either an RF4CE controller or an RF4CE target. Conversely, a HID adaptor would only be realized by an RF4CE target since more resources would need to be available.

The RF4CE ZID profile does not define any host side behavior beyond the HID adaptor.

2.1 Architectural components

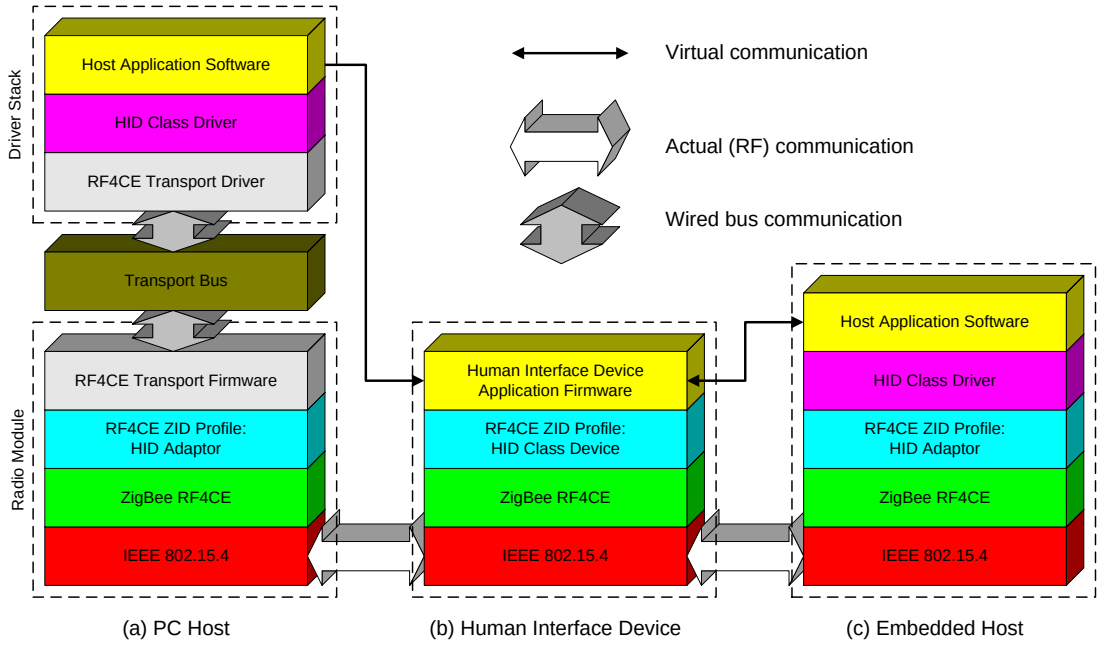

Figure 1 – Illustration of a HID class device and host architecture

In order to describe the architecture of the ZID profile, it is useful to fit it within the context of an overall HID system. Figure 1 illustrates an example HID system containing a human interface device (b) with a PC host (a) and with an embedded host (c) such as a TV. Each device and stack component is described in the following sections. Note that not all components will be present in all implementations.

2.1.1 Device descriptions

2.1.1.1 Human interface device

The human interface device is designed to allow a human user to control the operation of some equipment such as a PC or a TV. Its application provides controls to the user with which they interact to generate various stimuli, which, in turn, are converted by the HID class device into machine recognizable commands.

2.1.1.2 PC host

The PC host is typically (but not always) split into two parts: a radio module and a driver stack which communicate through some wired transport bus, which could be, for example, USB or RS232. The radio module may act as a proxy for the remote HID class device, informing the HID class driver of the appropriate descriptor hierarchy for each paired device.

2.1.1.3 Embedded host

The embedded host is similar to the PC host except that the entire function will likely be integrated, possibly on the RF4CE radio board itself. An example of an embedded host would be a TV that supports pointing functionality.

Unlike on the PC host, the HID class driver on the embedded host does not need to support a general HID report interpreter. Rather, the driver need only support those parts of the interpreter that will be expected from devices supported by the embedded host.

2.1.2 Stack components

2.1.2.1 IEEE 802.15.4

The IEEE 802.15.4 PHY/MAC stack [R3] facilitates the low level RF transport.

2.1.2.2 ZigBee RF4CE

The ZigBee RF4CE stack [R1] facilitates compliance to the ZigBee RF4CE standard.

2.1.2.3 RF4CE ZID profile: HID class device

The RF4CE ZID profile HID class device allows a compliant device to communicate with a host via the standard HID class protocol [R2] by passing report descriptors containing pertinent pieces of information which, when interpreted by the host, indicates the desired user interaction using the human interface device.

2.1.2.4 Human interface device application firmware

The human interface device application firmware enables the specific functionality of the device. For example, firmware that interprets mouse movements and converts them to HID reports.

2.1.2.5 RF4CE ZID profile: HID adaptor

The RF4CE ZID profile HID adaptor serves as an interface between the RF4CE stack and the HID class driver. The HID adaptor is responsible for establishing RF4CE and HID connections between the host and remote RF4CE HID class devices. Once the RF4CE and HID connections are established, the HID adaptor processes RF4CE ZID profile commands arriving from the RF4CE stack and, as necessary, passing them to the HID class driver (possible via a wired transport) and vice versa.

2.1.2.6 RF4CE transport firmware

The RF4CE transport firmware, used together with a wired transport and the RF4CE transport driver, is an optional embedded component that facilitates communication between the HID adaptor and the HID class driver over a wired transport bus, such as USB or RS232. This component processes commands received from the HID adaptor for transmission over the wired transport bus and vice versa.

2.1.2.7 RF4CE transport driver

The RF4CE transport driver, used together with a wired transport and the RF4CE transport firmware, is an optional host component that facilitates communication between the HID class driver and the HID adaptor over a wired transport bus, such as USB or RS232. This component processes commands received from the HID class driver for transmission over the wired transport bus and vice versa.

2.1.2.8 HID class driver

The HID class driver interprets and drives the HID class facility on the host, permitting specialized HID application software to facilitate the appropriate HID function.

2.1.2.9 Host application software

The host application software implements the function which is being controlled by the HID class device (e.g. to move a cursor around the screen in response to reports from a mouse device).

3 ZID command frames

For details of the certification status of commands and features in this specification, implementers should consult the latest version of ZigBee Alliance document Certification Status of ZID and GDP Commands and Features, [R6] .

The ZID profile command frame format shall conform to the GDP generic command frame format [R5] .

The ZID profile shall utilize the GDP commands listed in Table 1.

Table 1 – GDP command frames utilized by the ZID profile

GDP command code	Description	HID adaptor		HID class device	
		Tx	Rx	Tx	Rx
0x00	Generic response	M	M	M	M
0x01	Configuration complete	-	M	M	-
0x02	Heartbeat	-	M	O	-
0x03	Get attributes	M	M	M	M
0x04	Get attributes response	M	M	M	M
0x05	Push attributes	-	M	M	-

For commands defined in the ZID profile, the GDP command frame sub-field of the frame control field shall be set to 0 and the command code sub-field shall be set to one of the non reserved values listed in Table 2.

Table 2 – Values of the command code field for the ZID profile

ZID command code	Description	HID adaptor		HID class device		Reference
		Tx	Rx	Tx	Rx	
0x00	Reserved	-	-	-	-	-
0x01	Get report	M	-	-	M	3.2
0x02	Report data	M	M	M	M	3.3
0x03	Set report	O	-	-	O	3.4
0x04 - 0x0f	Reserved	-	-	-	-	-

M = Mandatory, O = Optional

3.1 ZID profile requirements for GDP command frames

3.1.1 Generic command frame

3.1.1.1 Data pending sub-field

This sub-field shall be set to 0 if the frame is transmitted by a HID class device and ignored if the frame is received by the HID adaptor.

3.1.2 Generic response command frame

The ZID profile shall utilize the response codes listed in Table 3.

Table 3 – Values of the response code field

Response code value	Origin	Description
0x00	GDP	Successful
0x01	GDP	Unsupported request
0x02	GDP	Invalid parameter
0x03	GDP	Configuration failure
0x40	ZID	Invalid report ID
0x41	ZID	Missing fragment
All other values	-	Reserved

3.1.3 Get attributes command frame

3.1.3.1 Effect on receipt

If the recipient node is not in its configuration state, for each attribute identifier included in the command frame, the recipient node shall check that it does not correspond to an attribute with an "*aplHID*" prefix (as specified in Table 6).

If it does correspond to an attribute with an "*aplHID*" prefix, the recipient node shall set the attribute status field of the corresponding attribute status record to indicate an illegal request and shall not include the attribute length and attribute value fields. It shall then move on to the next attribute identifier.

If it does not correspond to an attribute with an "*aplHID*" prefix, the recipient node shall continue to process the frame.

3.1.4 Push attributes command frame

3.1.4.1 Effect on receipt

If the recipient node is not in its configuration state, for each attribute identifier included in the command frame, the recipient node shall check that it does not correspond to an attribute with an "*aplHID*" prefix (as specified in Table 6).

If it does correspond to an attribute with an "*aplHID*" prefix, the recipient node shall set the response code of the generic response command frame to indicate an invalid parameter and ignore the rest of the frame.

If it does not correspond to an attribute with an "*aplHID*" prefix, the recipient node shall continue to process the frame.

3.2 Get report command frame

The get report command frame allows the HID adaptor to request a report from a HID class device.

3.2.1 Command frame format

The get report command frame shall be formatted as illustrated in Figure 2.

Bits: 8	8	8
Frame control	Report type	Report identifier
GDP header	ZID payload	

Figure 2 – Format of the get report command frame

3.2.1.1 Report type field

The report type field is 8 bits in length and specifies the type of the report being requested. This field shall contain one of the non reserved values listed in Table 4.

Table 4 – Values of the report type field

Report type value	Description
0x00	Reserved
0x01	Input
0x02	Output
0x03	Feature
0x04 – 0xff	Reserved

3.2.1.2 Report identifier field

The report identifier field is 8 bits in length and specifies the identifier of the report being requested.

3.2.2 When generated

The get report command frame is generated when a node wishes to determine a report from a remote node.

The originator shall direct the get report command frame to a specific pairing reference in its pairing table.

The report being requested shall be specified by the report type and report identifier fields.

If the transmission of the get report command frame was successful, the originator shall then request that the NWK layer enable its receiver and wait either for a maximum duration of *aplcMaxResponseWaitTime* or until a report data or generic response command frame is received from the recipient. If a report data or generic response command frame is not received within *aplcMaxResponseWaitTime*, the originator shall terminate the operation. If a report data or generic response command frame is received within *aplcMaxResponseWaitTime*, the originator is notified of the result of its get report attempt.

3.2.3 Effect on receipt

On receipt of the get report command frame, the recipient shall process the request and respond with either a report data or a generic response command frame.

If the report type and report identifier fields correspond to a valid descriptor on the recipient, it shall respond with a report data command frame containing the requested report.

If the report identifier field does not correspond to a valid descriptor on the recipient, it shall respond with a generic response command frame with a response code indicating an invalid report ID.

If any parameter is out of range, the recipient shall respond with a generic response command frame with a response code indicating an invalid parameter.

3.3 Report data command frame

The report data command frame allows a remote node to respond to a get report command frame or to send an unsolicited report.

3.3.1 Command frame format

The report data command frame shall be formatted as illustrated in Figure 3.

Bits: 8	Variable	Variable	Variable	Variable
Frame control	Report data record 0	Report data record 1	…	Report data record *n*-1
GDP header	ZID payload			

Figure 3 – Format of the report data command frame

The report data command frame can contain n ($1 \leq n$) report data records, each uniquely identified by the report identifier. Each report data record shall be formatted as illustrated in Figure 4.

Bits: 8	8	8	Variable
Report size	Report type	Report identifier	Report data

Figure 4 – Format of a report data record

3.3.1.1 Report size field

The report size field is 8 bits in length and specifies the length in bytes of the following fields within the report data record, i.e. the combined lengths of the report type, report identifier and report data fields of the current report data record.

3.3.1.2 Report type field

The report type field is 8 bits in length and specifies the type of the report being requested. This field shall contain one of the non reserved values listed in Table 4.

3.3.1.3 Report identifier field

The report identifier field is 8 bits in length and specifies the identifier of the report being requested.

3.3.1.4 Report data field

The report data field is of variable length and contains the contents of the report being requested.

3.3.2 When generated

The report data command is generated either in response to a valid get report command frame or when sending an unsolicited report to a remote node.

If the report data command frame is being sent in response to a get report command frame, the originator shall direct it to the same pairing table reference from which the original get report command frame was received. Otherwise, the originator shall direct the report data command frame to a specific pairing reference in its pairing table.

If the report data command frame is being sent in response to a get report command frame, only a single report data record shall be included which shall match the report that was requested. If the report data command frame is being sent unsolicited, the HID class device may send as many report data records that can fit into a single RF4CE frame.

For each report data record included, the report size field shall specify the length of the remaining fields of the report data record, i.e. the combined lengths of the report type, report identifier and report data fields of the current report data record. The report type and report identifier fields specify the report being sent. The report data field shall contain the actual report.

3.3.3 Effect on receipt

On receipt of the report data frame, the recipient is either notified of the results of its original get report request or of the current value of the report(s) from the originator.

The recipient shall not send any response to this frame.

3.4 Set report command frame

The set report command frame allows a node to send a report to a remote node. During configuration, a HID class device can use this command frame to transfer any NULL reports it may define to the HID adaptor.

3.4.1 Command frame format

The set report command frame shall be formatted as illustrated in Figure 5.

Bits: 8	8	8	Variable
Frame control	Report type	Report identifier	Report data
GDP header	ZID payload		

Figure 5 – Format of the set report command frame

3.4.1.1 Report type field

The report type field is 8 bits in length and specifies the type of the report being sent. This field shall contain one of the non reserved values listed in Table 4. However, if used during configuration to transfer NULL reports to the HID adaptor, the report type field shall be set to 0x01 (input).

3.4.1.2 Report identifier field

The report identifier field is 8 bits in length and specifies the identifier of the report being sent.

3.4.1.3 Report data field

The report data field is of variable length and contains the contents of the report being sent.

3.4.2 When generated

The set report command frame is generated when a node wishes to send a report to a remote node, possibly to configure initial settings.

The originator shall direct the set report command frame to a specific pairing reference in its pairing table.

The report being sent shall be specified by the report type and report identifier fields. The report itself shall be contained in the report data field.

If the transmission of the set report command frame was successful, the originator shall then request that the NWK layer enable its receiver and wait either for a maximum duration of *aplcMaxResponseWaitTime* or until a generic response command frame is received from the recipient. If a genric response command frame is not received within *aplcMaxResponseWaitTime*, the originator shall terminate the operation. If a generic response command frame is received within *aplcMaxResponseWaitTime*, the originator is notified of the result of its set report attempt.

3.4.3 Effect on receipt

On receipt of the set report command frame, the recipient shall respond with a generic response command frame.

If the report type and report identifier fields correspond to a valid descriptor on the recipient, it shall respond with a generic response command frame with a response code indicating success.

If the report identifier field does not correspond to a valid descriptor on the recipient, it shall respond with a generic response command frame with a response code indicating an invalid report ID.

If any parameter is out of range, the recipient shall respond with a generic response command frame with a response code indicating an invalid parameter.

4 ZID profile constants and attributes

4.1 ZID profile constants

The ZID profile shall utilize the following GDP constants [R5] :

- *aplcMaxPairIndicationWaitTime*
- *aplcMaxResponseWaitTime*
- *aplcMaxRxOnWaitTime*
- *aplcMinKeyExchangeTransferCount*

The additional constants that define the characteristics of the ZID profile are presented in Table 5.

Table 5 – Additional ZID profile constants

Constant	Description	Value
aplcIdleRateGuardTime	The length of time the HID adaptor will repeat a report from a remote HID class device when its idle rate is non-zero, before a direct report command frame must be received from the remote HID class device.	1500ms
aplcMaxConfigPrepTime	The maximum time a HID class device shall take to construct a push attributes command frame and send it to the HID adaptor during its configuration state.	200ms
aplcMaxConfigWaitTime	The maximum time the HID adaptor shall wait to receive a push attributes command frame from a HID class device during its configuration state.	300ms
aplcMaxNonStdDescCompsPerHID	The maximum number of non standard descriptor components that can be supported on a HID class device. The HID adaptor shall be able to store this many non standard descriptor components for each paired HID class device.	4
aplcMaxNonStdDescCompSize	The maximum size of a single non standard descriptor component. The HID adaptor shall be able to store descriptor components of this size for each component stored on a device.	256 bytes
aplcMaxNonStdDescFragmentSize	The maximum size of a non standard descriptor fragment that will fit in an over-the-air frame.	80 bytes
aplcMaxNullReportSize	The maximum size of a NULL report. The HID adaptor shall be able to store NULL reports of this size for each component stored on a device.	16 bytes
aplcMaxReportRepeatInterval	The maximum time between consecutive report data transmissions.	100ms
aplcMaxStdDescCompsPerHID	The maximum number of standard descriptor components that can be supported on a HID class device. The HID adaptor shall be able to store this many standard descriptor components for each paired HID class device.	12

Constant	Description	Value
aplcMinIntPipeUnsafeTxWindowTime	Minimum window time permitted when using the interrupt pipe before a safe transmission is required.	50ms

4.2 ZID profile attributes

The ZID profile defines attributes required to manage the way the ZID profile operates. These attributes are summarized in Table 6. The "HID" and "Adpt" columns indicate for a HID class device and the HID adaptor, respectively, whether an attribute is mandatory (M), optional (O) or not required (-).

Table 6 – ZID profile attribute summary

Attribute[2]	Identifier	Type	Ref	HID	Adpt	Default
Reserved	0x00 – 0x81	-	-	-	-	-
aplKeyExchangeTransferCount	0x82	8-bit unsigned integer	4.2.1	M	O	0x24
Reserved	0x83 – 0x9f	-	-	-	-	-
aplZIDProfileVersion	0xa0	16-bit unsigned integer	4.2.2	M	M	0x0100
aplIntPipeUnsafeTxWindowTime	0xa1	16-bit unsigned integer	4.2.3	M	-	*aplcMinIntPipeUnsafe-TxWindowTime*
aplReportRepeatInterval	0xa2	8-bit unsigned integer	4.2.4	M	-	(0.5 * *aplcMax-ReportRepeatInterval*)
Reserved	0xa3 – 0xdf	-	-	-	-	-
aplHIDParserVersion	0xe0	16-bit unsigned integer	4.2.5	M	M	0x0111
aplHIDDeviceSubclass	0xe1	8-bit unsigned integer	4.2.6	M	-	0x00
aplHIDProtocolCode	0xe2	8-bit unsigned integer	4.2.7	M	-	0x00
aplHIDCountryCode	0xe3	8-bit unsigned integer	4.2.8	M	M	0x00

[2] Attributes with an "*aplHID*" prefix shall be supported on each HID class device. Conversely, a host device shall support the ability to request and store these attributes for each paired HID class device. These attributes shall only be exchanged during the configuration state; attempts to exchange these attributes at other times will result in an error.

Attribute[2]	Identifier	Type	Ref	HID	Adpt	Default
aplHIDDeviceReleaseNumber	0xe4	16-bit unsigned integer	4.2.9	M	M	*Application specific*
aplHIDVendorId	0xe5	16-bit unsigned integer	4.2.10	M	M	*Application specific*
aplHIDProductId	0xe6	16-bit unsigned integer	4.2.11	M	M	*Application specific*
aplHIDNumEndpoints	0xe7	8-bit unsigned integer	4.2.12	M	-	*Application specific*
aplHIDPollInterval	0xe8	8-bit unsigned integer	4.2.13	M	-	*Application specific*
aplHIDNumStdDescComps	0xe9	8-bit unsigned integer	4.2.14	M	-	*Application specific*
aplHIDStdDescCompsList	0xea	List of 8-bit unsigned integers	4.2.15	O	-	*Application specific*
aplHIDNumNullReports	0xeb	8-bit unsigned integer	4.2.16	M	-	*Application specific*
Reserved	0xec – 0xef	-	-	-	-	-
aplHIDNumNonStdDescComps	0xf0	8-bit unsigned integer	4.2.17	M	-	*Application specific*
aplHIDNonStdDescCompSpec-i	0xf0 + *i*	Component specification	4.2.18	O	-	*Application specific*

4.2.1 *aplKeyExchangeTransferCount* attribute

As defined in the GDP [R5] .

4.2.2 *aplZIDProfileVersion* attribute

The *aplZIDProfileVersion* attribute is a 16-bit unsigned integer and shall specify the version of the ZID profile specification to which the device was designed. The format of this value shall be 0xJJMN representing a version JJ.M.N, where JJ is the major version number, M is the minor version number and N is the sub-minor version number. For this version of the ZID profile specification, the *aplZIDProfileVersion* attribute shall be set to the value 0x0100, representing v1.0.0.

4.2.3 *aplIntPipeUnsafeTxWindowTime* attribute

The *aplIntPipeUnsafeTxWindowTime* attribute is 16 bits in length and specifies the maximum time, during which transmissions use the unacknowledged, single channel service, permitted before an acknowledged, multiple channel transmission is required. This value shall be specified in the range (*aplcMinIntPipeUnsafeTxWindow* – 0xffff).

4.2.4 *aplReportRepeatInterval* attribute

The *aplReportRepeatInterval* attribute is an 8-bit unsigned integer and shall specify the interval in milliseconds at which a repeated report data command frame is transmitted to the HID adaptor. If the value of this attribute is zero, the HID class device shall not auto-repeat reports on behalf of the application. This value shall be specified in the range (0 – *aplcMaxReportRepeatInterval*).

4.2.5 *aplHIDParserVersion* attribute

The *aplHIDParserVersion* attribute is a 16-bit unsigned integer and shall specify the version of the core HID specification to which the device was designed. The format of this value shall be 0xJJMN representing a version JJ.M.N, where JJ is the major version number, M is the minor version number and N is the sub-minor version number. E.g., if the device was designed to comply with HID specification v1.1.1 (which is written v1.11 in [R2]), the *aplHIDParserVersion* attribute would need to be set to the value 0x0111.

This attribute is used in the HID descriptor.

4.2.6 *aplHIDDeviceSubclass* attribute

The *aplHIDDeviceSubclass* attribute is an 8-bit unsigned integer and shall specify the HID subclass code of the HID class device. This value shall be set to one of the non reserved values listed in Table 7.

Table 7 – Values of the *aplHIDDeviceSubclass* attribute

aplHIDDeviceSubclass	Description
0x00	No subclass
0x01	Boot interface subclass
0x02 – 0xff	Reserved

This attribute is used in the interface descriptor.

4.2.7 *aplHIDProtocolCode* attribute

The *aplHIDProtocolCode* attribute is an 8-bit unsigned integer and shall specify the HID protocol code of the HID class device supporting the boot protocol. This value shall be set to one of the non reserved values listed in Table 8.

Table 8 – Values of the *aplHIDProtocolCode* attribute

aplHIDProtocolCode	Description
0x00	None
0x01	Keyboard
0x02	Mouse
0x03 – 0xff	Reserved

This attribute is used in the interface descriptor.

4.2.8 *aplHIDCountryCode* attribute

The *aplHIDCountryCode* attribute is an 8-bit unsigned integer and shall specify the country for which the hardware is localized.

The valid country codes are listed in the HID specification [R2] .

This attribute is used in the HID descriptor.

4.2.9 *aplHIDDeviceReleaseNumber* attribute

The *aplHIDDeviceReleaseNumber* attribute is a 16-bit unsigned integer and shall specify the release number of the device itself. The format of this value shall be 0xJJMN representing a version JJ.M.N, where JJ is the major version number, M is the minor version number and N is the sub-minor version number.

This attribute is used in the device descriptor.

4.2.10 *aplHIDVendorId* attribute

The *aplHIDVendorId* attribute is a 16-bit unsigned integer and shall specify the USB-IF assigned identifier corresponding to the vendor of the device.

This attribute is used in the device descriptor.

4.2.11 *aplHIDProductId* attribute

The *aplHIDProductId* attribute is a 16-bit unsigned integer and shall specify a manufacturer assigned product identifier for the device.

This attribute is used in the device descriptor.

4.2.12 *aplHIDNumEndpoints* attribute

The *aplHIDNumEndpoints* attribute is an 8-bit unsigned integer and shall specify the number of endpoints on the device in addition to the mandatory control endpoint. This value shall be set to 0x01 if only the mandatory input endpoint is used by the device or 0x02 if both the mandatory input endpoint and the optional output endpoint are used by the device.

This attribute is used in the interface descriptor.

4.2.13 *aplHIDPollInterval* attribute

The *aplHIDPollInterval* attribute is an 8-bit integer and shall specify the polling interval for all endpoint data transfers, in 125us units. This value shall be set in the range 1 to 16 and is used as an exponent to calculate the value:

$$Poll\ Interval = 2^{(aplHIDPollInterval - 1)}$$

This attribute is used in the endpoint descriptor.

4.2.14 *aplHIDNumStdDescComps* attribute

The *aplHIDNumStdDescComps* attribute is an 8-bit integer in the range 0x00 - *aplcMaxStdDescCompsPerHID* and shall specify the number of RF4CE ZID profile defined standard HID descriptor components supported by the device (see clause 9).

4.2.15 *aplHIDStdDescCompsList* attribute

The *aplHIDStdDescCompsList* attribute is of length equal to the value of the *aplHIDNumStdDescComps* attribute and shall specify the list of standard RF4CE ZID profile defined HID descriptor components supported by the device (see clause 9).

4.2.16 *aplHIDNumNullReports* attribute

The *aplHIDNumNullReports* attribute is an 8-bit integer in the range 0x00 – *aplHIDNumNonStdDescComps* and shall specify the number of NULL reports (each corresponding to a non standard descriptor component) defined by the HID class device. These NULL reports can be

transferred via successive set report command frames sent to the HID adaptor during configuration. Note that it is possible that not all non standard descriptor components may have corresponding NULL reports.

4.2.17 *aplHIDNumNonStdDescComps* attribute

The *aplHIDNumNonStdDescComps* attribute is an 8-bit unsigned integer in the range 0x00 – *aplcMaxNonStdDescCompsPerHID* and shall specify the number of non standard HID descriptor components stored on the device. These descriptor components can be transferred via successive push attribute command frames sent to the HID adaptor; only one descriptor component shall be included with each push attribute command frame.

This attribute is used in the HID descriptor.

4.2.18 *aplHIDNonStdDescCompSpec-i* attribute

The *aplHIDNonStdDescCompSpec-i* attributes (where $1 \leq i \leq (aplHIDNumNonStdDescComps + 1)$) shall specify the actual HID descriptor components supported on the device, as defined in the HID specification [R2] . Each component specification shall be formatted as illustrated in Table 9.

Table 9 – Format of a HID descriptor component specification

Offset	Size (bytes)	Description
0	1	The HID descriptor type for this component. Valid values are: 0x22 – report 0x23 – physical All other values are reserved.
1	2	The HID component size in bytes. If the component size is greater than *aplcMaxNonStdDescFragmentSize* then the component must be sent to the HID adaptor using the component fragmentation mechanism described in 5.1.5.
3	1	The HID component identifier. For report descriptor components, this field indicates the report identifier for this component (0x80 – 0xff, 0x00 – 0x7f is reserved for standard components). For physical descriptor set components, this field indicates the physical descriptor set identifier for this component (0x00 – 0xfe, 0xff is reserved).
4	1 – *aplcMaxNonStd-DescCompSize*	The contents of the HID descriptor component in the format as defined in the HID specification [R2] .

These attributes are used in the HID and associated class-specific descriptors.

5 Functional description

This section describes the functionality that nodes must support to comply with the RF4CE ZID profile.

5.1 Generic description

The ZID profile shall conform to the policy and procedures specified in the GDP [R5] .

5.1.1 Discovery/pairing procedure

Devices implementing the ZID profile shall support the push-button pairing protocol, specified in the GDP. In the push-button pairing procedure, the pairing originator shall be synonymous with a HID class device and the pairing recipient with the HID adaptor.

On receipt of a pairing push button stimulus from the user, the application shall instigate the discovery/pairing procedure according to the type of the node. The procedure for each type of node is described in the following sub-clauses.

In this version of the specification, pairing between two HID adaptors shall not be permitted.

5.1.2 Transmission model

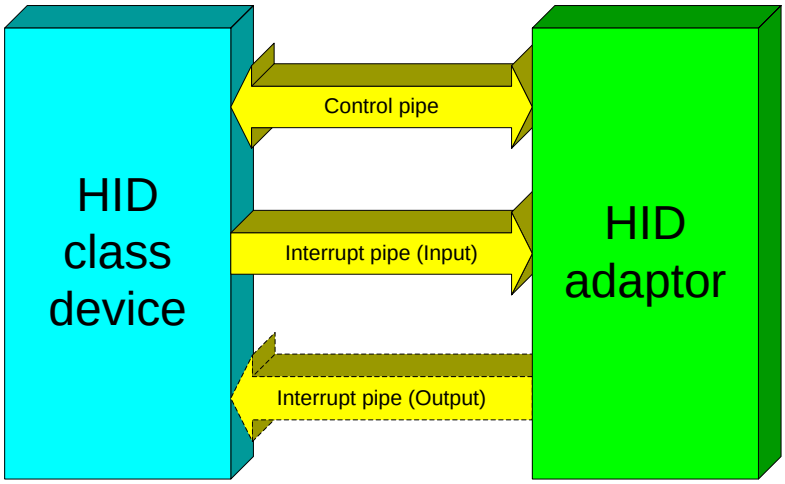

Figure 6 – ZID transmission model

Figure 6 illustrates the transmission model defined by the HID specification [R2] . Three communication pipes are defined: a mandatory bi-directional control pipe, a mandatory interrupt pipe from the HID class device to the host and an optional interrupt pipe from the host to the HID class device. The control pipe permits information to be exchanged between a HID class device and a host or for a host to poll a HID class device for its reports. The input interrupt pipe permits the HID class device to transfer low latency or asynchronous reports to the host. Finally, the output interrupt pipe, if defined, permits the host to transfer low latency or asynchronous reports to the HID class device.

The RF4CE protocol does not provide the communication services to directly support the equivalent of a control pipe or interrupt pipes, providing instead only a single communications mechanism. However, different levels of service can be achieved through the use of a variety of transmission modes.

In this profile, the application may choose whether to send report data via the control pipe or the interrupt pipe, depending on the efficiency needs and security requirements of the application. The control pipe shall be used in all other cases. These options are summarized in Table 10.

Table 10 – Transmission options per transmission type

Transmission	Control pipe Unicast	Control pipe Broadcast	Interrupt pipe Unicast
Report data (not covered by 5.4.2)	✓	✓	✓
Report data (covered by 5.4.2)	✓	×	✓
All other ZID profile commands	✓	×	×

✓ = permitted, × = not permitted

5.1.2.1 Control pipe transmissions

Transmissions sent via the "control pipe" from a node to its recipient shall be sent using either the unicast, acknowledged and multiple channel transmission service or the broadcast service. The Control pipe broadcast can be single channel or multiple channel.

5.1.2.2 Interrupt pipe transmissions

Transmissions sent via an "interrupt pipe" from a node to its recipient shall be sent using a combination of the unicast, unacknowledged and single channel transmission service and the unicast, acknowledged and multiple channel transmission service. Transmissions sent via an "interrupt pipe" shall not use the broadcast service. The transmission mode ensures that the unicast, unacknowledged and single channel transmission service is permitted to be used for a maximum duration of *aplIntPipeUnsafeTx-WindowTime* without the node attempting to re-confirm the channel.

The first transmission on the interrupt pipe after the node has started shall always use the unicast, acknowledged and multiple channel transmission service, thus performing the initial acquisition of the HID adaptor channel. The node shall then use the unicast, unacknowledged and single channel transmission service for a maximum duration of *aplIntPipeUnsafeTxWindowTime*. It shall then use the unicast, acknowledged and multiple channel transmission service for the next transmission, thus allowing the HID class device to reacquire the HID adaptor should channel conditions have forced the HID adaptor to switch to an alternative channel. The node shall repeat this procedure for all subsequent transmissions on the interrupt pipe. This concept is illustrated in Figure 7.

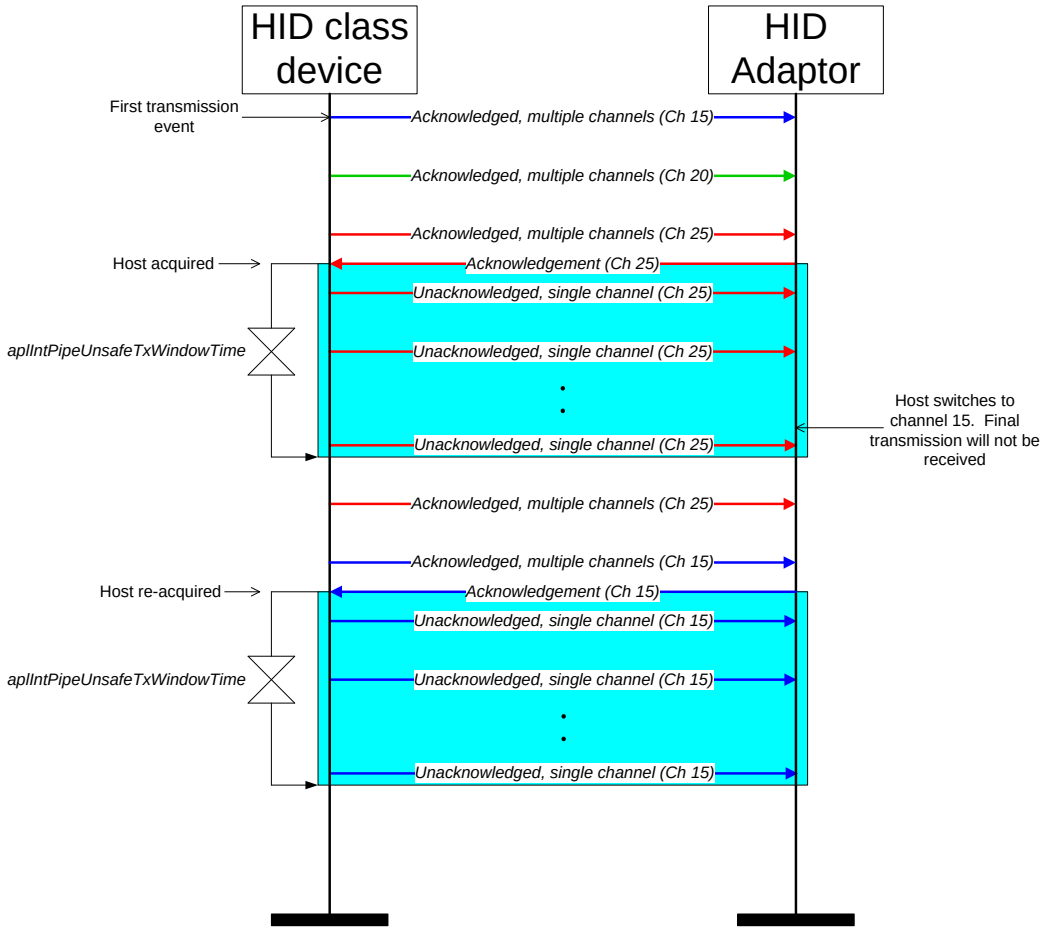

Figure 7 – Interrupt pipe transmissions

5.1.3 Communication with a HID class device

The RF4CE ZID profile provides a mechanism to allow the HID adaptor to communicate with a sleeping HID class device. This can be accomplished in two ways:

1. The HID class device can periodically check in with the HID adaptor to give it the opportunity to send a message back. This can be accomplished by transmitting a heartbeat command frame to the HID adaptor.
2. The HID adaptor can request, within a command frame, that the HID class device keep its receiver on for a maximum duration of *aplcMaxRxOnWaitTime* to allow the HID adaptor to send it a message. This can be accomplished by setting the data pending sub-field of the frame control field of a command frame transmitted to the HID class device.

The HID class device may transmit a heartbeat command frame to the HID adaptor under the control of its application. If the HID adaptor responds with a generic response frame with a status indicating success, it has no data to send to the HID class device.

It is recommended that a HID class device that defines output reports (i.e., sent from the host to the HID class device) periodically transmits heartbeat command frames to the HID adaptor.

5.1.3.1 HID adaptor usage

In order to request that a HID class device leaves its receiver on following the completion of the current command, to allow the HID adaptor to send further commands, the HID adaptor shall set the data pending sub-field of the frame control field of a transmitted command frame to 1. If a response to this command is expected, the HID adaptor shall ensure that its receiver is enabled to allow the response to be received, as described in the individual command frame descriptions. The HID adaptor may then send a further command frame to the HID class device.

5.1.3.2 HID class device effect on receipt

On receipt of a command frame with the data pending sub-field of the frame control field set to 1, a HID class device shall first complete the command, as appropriate, i.e., if the command requires a response, it shall process the command and then generate and transmit a response back to the HID adaptor, as described in the individual command frame descriptions.

The HID class device shall then request that the NWK layer enable its receiver and wait either for a maximum duration of *aplcMaxRxOnWaitTime* or until a command frame is received from the HID adaptor. If a command frame is not received within *aplcMaxRxOnWaitTime*, the HID class device shall terminate the operation and disable its receiver. If a command frame is received within *aplcMaxRxOnWaitTime*, the HID class device shall attempt to process it.

5.1.3.3 Example sequence chart

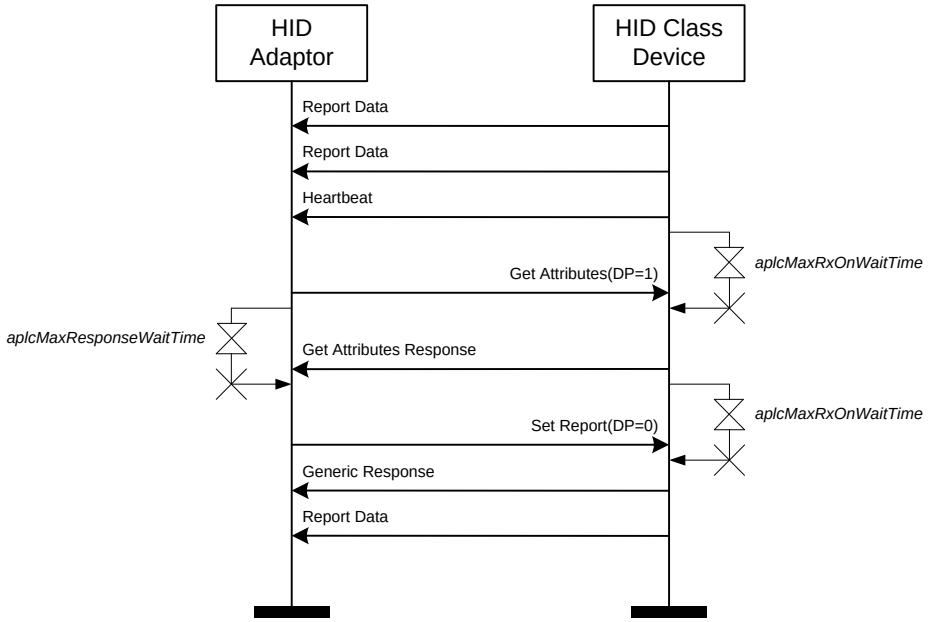

Figure 8 – Example sequence chart illustrating the use of the data pending sub-field

Figure 8 illustrates the use of the heartbeat command frame and the data pending (DP) sub-field of the frame control field of a command frame to enable communication to a sleeping HID class device. In this example, the HID class device sends report data command frames as normal but periodically sends a heartbeat command frame to the HID adaptor. The HID class device then enables its receiver for a maximum duration of *aplcMaxRxOnWaitTime*.

During this time, the HID adaptor takes the opportunity to send a get attributes command frame back to the HID class device. It also wants to send an updated output report to the HID class device so it sets the data pending sub-field of the frame control field to 1. The HID class device processes the get attributes command frame and responds back with a get attributes response command frame. As the HID adaptor indicated data is still pending, the HID class device then again enables its receiver for a maximum duration of *aplcMaxRxOnWaitTime*.

During this time, the HID adaptor sends a set report command frame back to the HID class device. As it has nothing further to communicate to the HID class device, it sets the data pending sub-field of the frame control field to 0. The HID class device processes the set report command frame and responds back with a generic response command frame. As the data pending sub-field was set to 0, the HID class device does not re-enable its receiver and settles back into its normal operation of sending report data command frames.

5.1.4 Support for boot protocols

The ZID profile permits the use of boot protocols to be used for mouse and keyboard devices. However, unlike normal HID procedures, the ability to support the boot protocol is indicated by the HID class device itself but the boot mechanism is handled by the HID adaptor. If requested to do so, the HID adaptor truncates mouse or keyboard reports sent by remote HID class devices to match the appropriate boot protocol reports (see section 5.2.3). Consequently, any non standard mouse or keyboard reports may only append fields to the appropriate standard boot protocol reports.

Devices that wish to support the boot protocol shall indicate the fact by setting *aplHIDDeviceSubclass* to 0x01 and *aplHIDProtocolCode* as appropriate for the device. These devices do not have to explicitly define an appropriate boot protocol report descriptor since its format is well defined (see sections 9.1 and 9.2).

If a mouse or keyboard device wishes to utilize the standard report formats (which also define the boot protocol), it shall indicate the fact via the *aplHIDNumStdDescComps* and *aplHIDStdDescCompsList* attributes, as appropriate for the device type.

If a device wishes to extend the standard report formats it may do so by defining non-standard report descriptors via the *aplHIDNumNonStdDescComps* and *aplHIDNonStdDescCompSpec-i* attributes. However, if the device indicates it also supports the boot protocol, the first 3 bytes (for a mouse report) and 8 bytes (for a keyboard input report) shall match the standard report formats illustrated in Figure 25 and Figure 28, respectively.

5.1.5 Fragmentation of descriptor components

If a HID class device pushes a non standard descriptor component with a size greater than *aplcMaxNonStdDescFragmentSize* bytes, it shall be sent to the HID adaptor using descriptor component fragmentation. Otherwise, the non standard descriptor component shall be sent complete.

Descriptor component fragments shall be sent using the format illustrated in Figure 9. This attribute value payload shall be placed in the attribute value fields of a push attributes command frame.

1	**2**	**1**	**1**	**Variable**
Descriptor type	Descriptor component size	Component identifier	Fragment identifier	Descriptor component fragment
Attribute value payload				

Figure 9 – Format of a fragmented descriptor component

The descriptor type, descriptor component size and component identifier fields shall be copied from the component specification of the associated descriptor component. These fields shall be identical for each fragment of the same descriptor component.

The fragment identifier field shall contain a unique and incrementing value for each fragment sent. A fragment with an identifier of 0x00 shall be assumed to be the first. Each subsequent fragment send shall have an identifier value of the last fragment sent incremented by one.

The descriptor component fragment field shall contain the actual fragments of the descriptor component being sent. Each descriptor component fragment shall contain at most *aplcMaxNonStdDescFragmentSize* bytes. The first *aplcMaxNonStdDescFragmentSize* bytes of the descriptor component shall be included in the fragment with identifier 0x00, the second *aplcMaxNonStdDescFragmentSize* bytes in the fragment with identifier 0x01, etc. until all of the descriptor component bytes have been sent. The final fragment shall contain less than or equal to *aplcMaxNonStdDescFragmentSize* bytes.

On receipt of a push attributes command frame containing an attribute *aplHIDNonStdDescCompSpec-i* with a component specification indicating a size greater than *aplcMaxNonStdDescFragmentSize* bytes, the HID adaptor shall assume that the descriptor component is fragmented. For each descriptor component attribute, each fragment shall contain the same descriptor type, descriptor component size and component identifier fields.

The first fragment received for a specific descriptor component shall contain a fragment identifier equal to 0x00 and shall be used to verify all subsequent fragments relating to the same descriptor component. Each subsequently received fragment shall have a fragment identifier incrementing by one. If a fragment is received with a fragment identifier that was not expected, the HID adaptor shall generate and transmit a generic response command frame with a response code indicating a missing fragment. Otherwise, the HID adaptor shall generate and transmit a generic response command frame with a response code indicating success for each fragment.

The HID adaptor shall defragment the descriptor component by placing the fragment with identifier equal to 0x00 into the first *aplcMaxNonStdDescFragmentSize* bytes of the descriptor component, the fragment with identifier equal to 0x01 into the second *aplcMaxNonStdDescFragmentSize* bytes, etc. until all of the descriptor component bytes have been reconstructed.

5.2 HID adaptor operation

A host device intending to conform to the RF4CE ZID profile shall define a HID adapter component, as illustrated in Figure 1. A HID adaptor shall be constructed from an RF4CE target. The HID adaptor component is described in this section.

5.2.1 HID adaptor proxy table

In order to be able to act as a proxy for paired HID class devices and to be able to re-proxy to the host in the event of a HID adaptor power cycle, the HID adaptor shall be able to store the set of HID related attributes, listed in Table 11, for each paired HID class device. An entry shall be created in the proxy table when a HID class device is first paired to the HID adaptor and populated with the attributes extracted from the HID class device during its configuration stage or set to their default values, as described below. The HID adaptor shall ensure that the proxy table can be preserved following a power cycle.

In the event of a warm start, following a power cycle, the HID adaptor shall be able to read these values and re-proxy to the host on behalf of each HID class device during the proxy state.

Table 11 – Attributes required in a single proxy table entry

HID attribute	Type	Maximum size (bytes)
aplHIDParserVersion	16-bit unsigned integer	2
aplHIDDeviceSubclass	8-bit unsigned integer	1
aplHIDProtocolCode	8-bit unsigned integer	1
aplHIDCountryCode	8-bit unsigned integer	1
aplHIDDeviceReleaseNumber	16-bit unsigned integer	2
aplHIDVendorId	16-bit unsigned integer	2
aplHIDProductId	16-bit unsigned integer	2

HID attribute	Type	Maximum size (bytes)
aplHIDNumEndpoints	8-bit unsigned integer	1
aplHIDPollInterval	8-bit unsigned integer	1
aplHIDNumStdDescComps	8-bit unsigned integer	1
aplHIDStdDescCompsList	List of 8-bit unsigned integers	$StdComp_{max}$
aplHIDNumNonStdDescComps	8-bit unsigned integer	1
aplHIDNonStdDescCompSpec-i	List of descriptor components	$NonStdComp_{max} * (NonStdComp_{size} + 3)$
aplHIDNumNullReports	8-bit unsigned integer	1
aplDeviceIdleRate	8-bit unsigned integer	1
aplCurrentProtocol	8-bit unsigned integer	1
aplNullReportSpecList	List of NULL reports	$NonStdComp_{max} * (NullReport_{size} + 2)$

Thus:

$$Total\ storage\ (bytes) = 18 + StdComp_{max} + NonStdComp_{max} * (NonStdComp_{size} + NullReport_{size} + 5)$$

Where:

$StdComp_{max}$ = *aplcMaxStdDescCompsPerHID*

$NonStdComp_{max}$ = *aplcMaxNonStdDescCompsPerHID*

$NonStdComp_{size}$ = *aplcMaxNonStdDescCompSize*

$NullReport_{size}$ = *aplcMaxNullReportSize*

5.2.1.1 *aplHID* attributes

The attributes labeled with the prefix "*aplHID*" shall be obtained directly from the remote HID class device during the configuration state. Detailed information on each of these attributes can be found in sub-clause 4.2.

If the value of the *aplHIDNumNonStdDescComps* attribute is greater than zero, then the HID adaptor shall ensure that each application-defined non standard descriptor component is obtained from the HID class device before notifying the host.

5.2.1.2 *aplDeviceIdleRate* attribute

The *aplDeviceIdleRate* attribute is an 8-bit unsigned integer and shall specify the idle rate of the HID class device, and shall take one of the values listed in Table 12. By default this attribute shall be set to 0x00 (repetition disabled).

Table 12 – Values of the *aplDeviceIdleRate* attribute

***aplDeviceIdleRate* value**	**HID idle rate**
0x00	Infinite (repetition disabled)
0x01	0.004s
0x02	0.008s
…	…
0xfe	1.016s
0xff	1.020s

The idle rate specifies how quickly HID class device reports are passed to the host by the HID adaptor, which shall manage the idle rate concept on behalf of the remote HID class device. Due to the low power and data rate capabilities of the RF4CE standard, the HID class device itself is not expected to support changes to its idle rate.

5.2.1.3 *aplCurrentProtocol* attribute

The *aplCurrentProtocol* attribute is an 8-bit unsigned integer and shall specify the current protocol currently in use by the HID class device. This value shall be set to one of the non-reserved values listed in Table 13. By default this attribute shall be set to 0x01 (report protocol).

Table 13 – Values of the *aplCurrentProtocol* attribute

***aplCurrentProtocol* value**	**Description**
0x00	Boot
0x01	Report
0x02 – 0xff	Reserved

5.2.1.4 *aplNullReportSpecList* attribute

The *aplNullReportSpecList* attribute is a list of NULL report specifications and shall specify the actual NULL reports supported on the device, each corresponding to a particular non standard descriptor component (note that not all non standard descriptor components have an associated NULL report). Each NULL report specification shall be formatted as illustrated in Table 14.

Table 14 – Format of a NULL report specification

Offset	**Size (bytes)**	**Description**
0	1	The HID report identifier to which this NULL report corresponds. This field shall be specified in the range 0x80 – 0xff (0x00 – 0x7f is reserved for standard components).
1	1	The NULL report size in bytes.
2	1 – *aplcMax-NullReportSize*	The contents of the NULL report in the format as defined by the corresponding non standard descriptor component.

5.2.2 HID adapter state machine

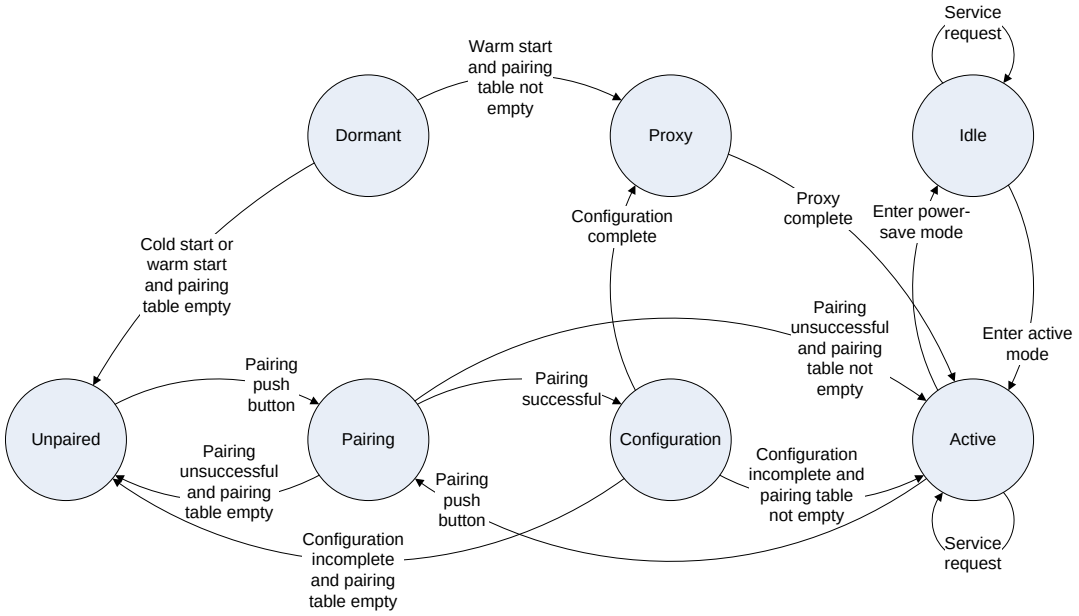

Figure 10 – HID adapter state machine

A host implementing the HID adapter component of the RF4CE ZID profile should implement the state machine shown in Figure 10 and summarized in Table 15.

Table 15 – HID adapter (host side) state transitions

State	Action	Condition	Next state
Dormant	HID adaptor performs a cold or warm start, at the discretion of the application, in order to initialize itself.	Cold start or warm start and pairing table is empty.	Unpaired
		Warm start and pairing table is not empty.	Proxy
Unpaired	HID adaptor is ready to be paired with a HID class device but cannot otherwise communicate.	Pairing push button pressed.	Pairing
Pairing	HID adaptor performs the pairing procedure with a HID class device.	Pairing is successful.	Configuration
		Pairing is unsuccessful and the pairing table is empty.	Unpaired
		Pairing unsuccessful and the pairing table not empty.	Active

State	Action	Condition	Next state
Configuration	HID adaptor waits to be interrogated by the HID class device to determine its compatibility. HID adaptor collects the HID related attributes communicated from the HID class device. If all of the HID related attributes are not successfully collected from the HID class device, the HID adaptor unpairs the HID class device, removing its entry from the pairing table.	All HID related attributes are successfully collected from the HID class device.	Proxy
		All HID related attributes are not successfully collected from the HID class device and the pairing table is not empty.	Active
		All HID related attributes are not successfully collected from the HID class device and the pairing table is empty.	Unpaired
Proxy	HID adaptor builds appropriate descriptors from the remote HID class device and informs the HID class driver of its existence, exchanging HID commands as necessary. If proxy is entered from the dormant state following a warm start, the HID adaptor informs the HID class driver of all stored remote HID class devices.	All appropriate HID class device information is exchanged with the HID class driver.	Active
Active	HID adaptor switches into its normal operating state during which the receiver shall be enabled to service requests from remote nodes.	Pairing push button pressed.	Pairing
		ZID profile request is serviced.	Active
		HID adaptor chooses or is switched into its power save mode.	Idle
Idle	RF4CE power-saving mode is activated such that the receiver is periodically enabled for the active period.	ZID profile request is serviced.	Idle
		HID profile request is serviced. Application chooses to enter active mode.	Active

5.2.2.1 Dormant state

The *dormant* state represents the state of the HID adaptor before it is initialized using either the RF4CE cold or warm start procedures. Typically, if the HID adaptor has not been used before or a hard reset has occurred, the device should start in this state and the cold start procedure should be followed. Conversely, if the HID adaptor has been power cycled from any other state, the device should transition to this state and the warm start procedure should be followed. In this state, the transceiver is not enabled so the node is effectively dormant.

On first activation of the HID adaptor, e.g. when power is applied for the first time, it shall instigate the RF4CE cold start procedure by issuing the NLME-RESET.request primitive to the NLME with the SetDefaultNIB parameter set to TRUE. On receipt of a successful reset response notification from the NLME, via the NLME-RESET.confirm primitive, the HID adapter shall initialize the NIB attributes as described in [R5] . The HID adapter then transitions unconditionally to the *unpaired* state.

On activation of the HID adaptor following a power cycle, it shall instigate the RF4CE warm start procedure by issuing the NLME-RESET.request primitive to the NLME with the SetDefaultNIB parameter set to FALSE and wait for a successful reset response notification from the NLME, via the NLME-RESET.confirm primitive. If the pairing table is empty, the HID adaptor then transitions to the *unpaired* state. If the pairing table is not empty, the HID adaptor then transitions to the *proxy* state.

5.2.2.2 Unpaired state

The *unpaired* state represents the state of the HID adapter after it is initially activated or when the RF4CE pairing table is empty. In this state the HID adaptor is powered but the transceiver is not enabled.

On receipt of a pairing push button stimulus from the user, the HID adapter transitions to the *pairing* state.

5.2.2.3 Pairing state

The *pairing* state represents the state of the HID adapter while it is undergoing the RF4CE pairing procedure in order to connect to another RF4CE node with which to communicate.

On entry into the *pairing* state, the HID adapter shall instigate the RF4CE discovery/pairing procedure by issuing the NLME-AUTO-DISCOVERY.request primitive to the NLME.

For the automatic discovery, the HID adaptor shall set the following parameters: the automatic discovery response mode duration shall be set to 0x1c9c38 (30s); the profile identifier list disclosed as supported by the node shall contain at least the value 0x02 (the ZID profile identifier) and the list of device types supported by the node shall contain at least one entry taken from the RF4CE device type list [R4] .

On receipt of a successful automatic discovery response mode notification from the NLME, via the NLME-AUTO-DISCOVERY.confirm primitive, the HID adaptor shall request that the NWK layer enable its receiver and wait either for *aplcMaxPairIndicationWaitTime* or until a corresponding pairing request notification is received from the NLME, via the NLME-PAIR.indication primitive. If the expected pairing request notification is not received within *aplcMaxPairIndicationWaitTime*, the HID adaptor shall terminate the procedure and perform no further processing. If the notification indicated that a provisional pairing table entry was created, the HID adaptor shall decide whether to respond to the pair. The procedure for deciding whether to respond to a pair request is out of the scope of this specification. If the HID adaptor chooses to respond to the pair request, it shall instigate it with the network layer by issuing the NLME-PAIR.response primitive to the NLME.

If the notification indicated a key exchange transfer count that was less than *aplcMinKeyExchange-TransferCount*, the HID adaptor shall respond indicating that the pair request was not permitted. If the notification indicated that there is no capacity for the new entry in the pairing table, the HID adaptor shall respond indicating an identical (i.e. no capacity) status and perform no further processing. If the notification indicated that the pairing request corresponded to a duplicate entry, the HID adaptor shall respond with a successful status and perform no further processing. The HID adaptor shall instigate the response by issuing the NLME-PAIR.response primitive to the NLME

If the transmission of the subsequent pair response command and subsequent link key exchange, as necessary, was successful, indicated through the NLME-COMM-STATUS.indication primitive from the NLME, the HID adaptor shall be considered paired to the device with the indicated reference into the pairing table and the HID adaptor transitions to the configuration state.

If any part of the pairing procedure fails with the pairing table still having at least one entry remaining, the HID adapter transitions to the *active* state. If any part of the pairing procedure fails leaving the pairing table empty, the HID adapter transitions to the *unpaired* state.

5.2.2.4 Configuration state

The *configuration* state represents the state of the HID adapter after it has successfully paired with another RF4CE node but before it has identified the node to the host as a HID class device. During this state, the HID adapter collects attribute information from the remote node to allow it to proxy for it to the host.

The HID adaptor first waits until the HID class device interrogates it for compatibility. To do this, the HID adaptor shall request that the NWK layer enable its receiver and wait either for a maximum duration of *aplcMaxConfigWaitTime* or until a get attributes command frame is received from the remote node. If any other command frame is received, it shall be discarded. If a get attributes command frame is received within *aplcMaxConfigWaitTime*, the HID adaptor shall attempt to process it. If the attributes are valid, the HID adaptor shall generate and transmit a corresponding get attributes response command frame, containing the requested attributes, back to the remote node. Any invalid attributes shall be discarded. If a get attributes command frame is not received within *aplcMaxConfigWaitTime*, the HID adaptor shall terminate the operation.

The HID adapter then collects attribute information from the remote node. The HID adapter shall collect attribute information by waiting for push attribute command frames to be received from the HID class device.

The HID adapter needs to collect the attributes required to construct appropriate USB descriptors, as listed in Table 16.

Table 16 – ZID profile attributes by USB descriptor

Attribute	Used in USB descriptor	Reference
aplHIDVendorId	Device	4.2.10
aplHIDProductId	Device	4.2.11
aplHIDDeviceReleaseNumber	Device	4.2.9
aplHIDNumEndpoints	Interface	4.2.12
aplHIDDeviceSubclass	Interface	4.2.6
aplHIDProtocolCode	Interface	4.2.7
aplHIDPollInterval	Endpoint	4.2.13
aplHIDParserVersion	HID	4.2.5
aplHIDCountryCode	HID	4.2.8
aplHIDNumStdDescComps	HID	4.2.14
aplHIDNumNonStdDescComps	HID	4.2.17
aplHIDNumNullReports	-	4.2.16
aplHIDStdDescCompsList[3]	Report	4.2.15
aplHIDNonStdDescCompSpec-i[3]	Report	4.2.18

[3] The *aplHIDStdDescCompsList* attribute is only required if *aplHIDNumStdDescComps* > 0 and the *aplHIDNonStdDescCompSpec-i* attributes are only required if *aplHIDNumNonStdDescComps* > 0.

To collect the attribute information from the remote node, the HID adaptor shall request that the NWK layer enable its receiver and wait either for a maximum duration of *aplcMaxConfigWaitTime* or until a push attributes command frame is received from the remote node. If any other command frame is received, it shall be discarded. If a push attributes command frame is received within *aplcMaxConfigWaitTime*, the HID adaptor shall attempt to process it. If the attributes are valid, the HID adaptor shall store the attributes in its proxy table. Any invalid attributes shall be discarded. If a push attributes command frame is not received within *aplcMaxConfigWaitTime*, the HID adaptor shall terminate the operation.

The HID adaptor shall then request that the NWK layer enable its receiver and wait either for a maximum duration of *aplcMaxConfigWaitTime* or until a push attributes, set report or configuration complete command frame is received from the remote node. If any other command frame is received, it shall be discarded. If a push attributes command frame is received within *aplcMaxConfigWaitTime*, the HID adaptor shall attempt to process it. If the attributes are valid, the HID adaptor shall store the attributes in its proxy table. Any invalid attributes shall be discarded. If a set report command frame is received within *aplcMaxConfigWaitTime*, the HID adaptor shall attempt to process it. If the report identifier corresponds to a non standard descriptor component already received, the HID adaptor shall store the (NULL) report in its proxy table entry *aplNullReportList*. If the report identifier does not correspond to a non standard descriptor component already received, the HID adaptor shall discard the frame.

The HID adaptor shall repeat the process until a configuration complete command frame is received or a command frame is not received within a maximum duration of *aplcMaxConfigWaitTime*.

The HID adaptor shall then verify that it has received all of the attributes listed in Table 16 to allow it to create an entry in its proxy table for the device. If the received attributes allow a proxy table entry to be created, the HID adaptor shall create an entry in its proxy table with the received attributes. If a configuration complete command frame was received, the HID adaptor shall generate and transmit a generic response command frame back to the HID class device with a status code indicating success.

If the received attributes do not allow a proxy table entry to be created, the HID adaptor shall generate and transmit a generic response command frame back to the HID class device with a status code indicating a configuration failure.

This procedure is illustrated in Figure 11.

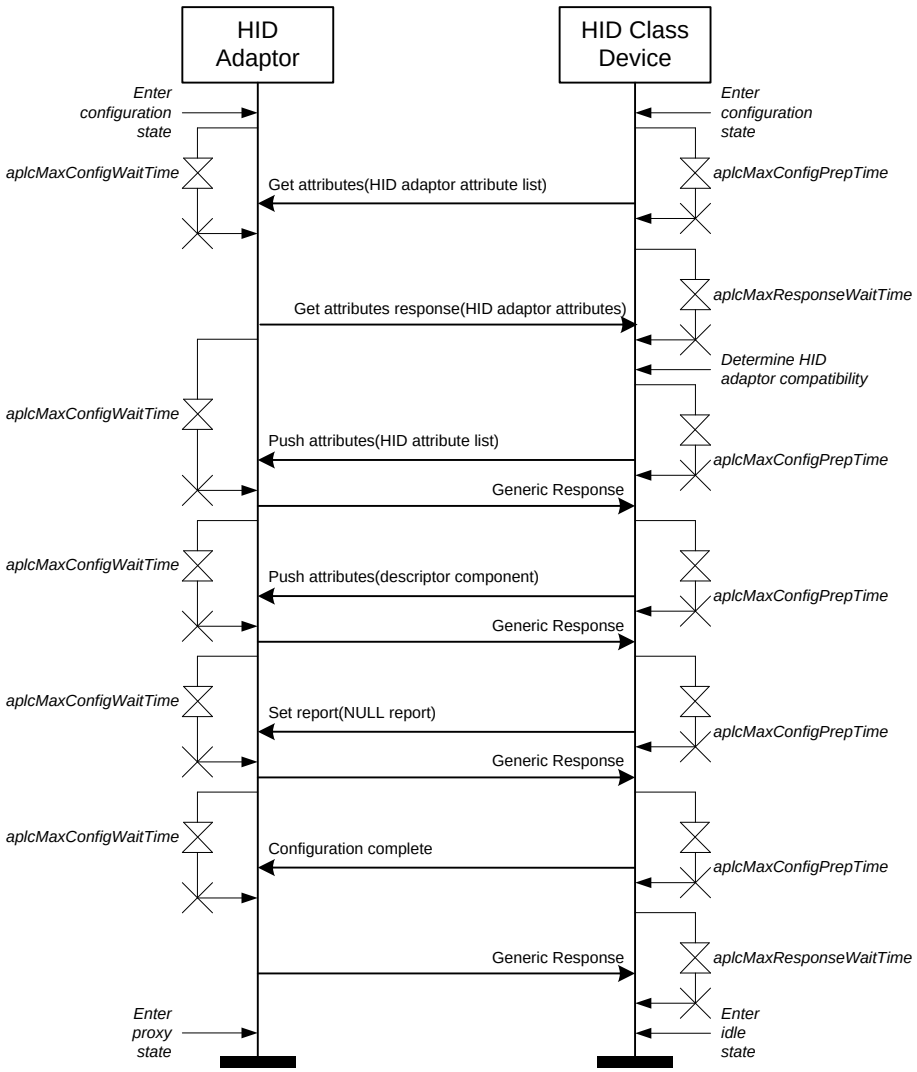

Figure 11 – Use of push attributes command frames in the configuration state

If all of the attributes listed in Table 16 are collected, the HID adapter transitions to the *proxy* state. If all of the attributes listed in Table 16 are not collected, the HID adaptor shall unpair the HID class device by issuing the NLME-UNPAIR.request primitive to the NLME. On receipt of a notification from the NLME of the unpair attempt, via the NLME-UNPAIR.confirm primitive, the HID adaptor determines if the pairing table is now empty. If the pairing table is empty, the HID adaptor transitions to the *unpaired* state. If the pairing table is not empty, the HID adaptor transitions to the *active* state.

5.2.2.5 Proxy state

The *proxy* state represents the state of the HID adaptor after it has successfully collected all HID attributes from a remote HID class device or after a warm start with a non-empty pairing table.

If the HID adaptor has entered the *proxy* state after a warm start, it shall construct appropriate device, configuration, interface, endpoint, HID, report and optional physical descriptor set for all the HID class devices currently stored in its pairing table (see clause 8 for details), based on the HID attributes of each device stored from previous configurations. Once constructed, the HID adapter shall notify the HID class driver of the presence of each HID class device.

If the HID adaptor has entered the *proxy* state after a successful configuration, it shall construct appropriate device, configuration, interface, endpoint, HID, report and optional physical descriptor set for the remote node (see clause 8 for details). Once constructed, the HID adapter shall notify the HID class driver of the presence of a new HID class device.

The HID adaptor then transitions unconditionally to the *active* state.

5.2.2.6 Active state

The *active* state represents the normal operating state of the HID adaptor. During this mode, the receiver shall be enabled to allow the HID adaptor to rapidly react to incoming HID requests from its paired, remote HID class devices. If a HID request, such as a report, is received from a remote node, the HID adaptor shall process the request and pass it to the HID class driver. Data received from a HID class device that has not been configured correctly shall be discarded. It then remains in the *active* state.

In the event that the host device chooses to enter its power-saving mode, the HID adaptor transitions to the *idle* state.

On receipt of a pairing push button stimulus from the user, the device transitions to the *pairing* state.

5.2.2.7 Idle state

The *idle* state represents the power-saving mode of the HID adaptor. During this mode, the receiver shall be periodically enabled (at the RF4CE duty cycle) for the RF4CE active period. During the time the receiver is enabled, the HID adaptor may process incoming HID requests from its paired, remote devices and pass them to the HID class driver. Data received from a HID class device that has not been configured correctly shall be discarded.

In the event that the host chooses to return to active mode, via some application specific means, the HID adaptor transitions to the *active* state.

5.2.3 Servicing requests from the HID class driver

Once the HID class driver has been notified of the presence of the HID class device, it may issue a number of requests to the HID adaptor, directed at specific interfaces which correspond to proxy table entries. It is imperative that these requests are dealt with in a timely fashion so all the required USB descriptors for the remote node must be available on request; this is the reason that the RF4CE ZID profile collects and constructs these descriptors before informing the HID class driver. These requests and their actions are described in the following sub-clauses.

Other requests between the HID class driver and the HID adaptor which could be used to instigate certain RF4CE specific functions, e.g., pairing, are out of scope for this specification and so must be provided as application specific features.

5.2.3.1 Get_Descriptor request

The HID adaptor shall access the requested descriptor for the appropriate HID class device (or generate it from the proxy table) and pass the descriptor to the HID class driver.

5.2.3.2 Get_Report request

The HID adaptor shall create a get report command frame and transmit it to the appropriate HID class device. On receipt of the corresponding report data command frame, the HID adaptor shall pass the report to the HID class driver.

5.2.3.3 Set_Report request

The HID adaptor shall create a set report command frame and transmit it to the appropriate HID class device.

5.2.3.4 Get_Idle request

The HID adaptor shall extract the idle rate of the appropriate HID class device from the proxy table (*aplDeviceIdleRate* attribute) and return its value to the HID class driver.

5.2.3.5 Set_Idle request

The HID adaptor shall update the idle rate of the appropriate HID class device in the proxy table (*aplDeviceIdleRate* attribute). The behavior of the HID adaptor in its reporting to the HID class driver, as proxy for a remote HID class device, is dependent on the value of the *aplDeviceIdleRate* attribute.

If the value of *aplDeviceIdleRate* is non-zero, when the next report is received from the associated remote HID class device, the HID adaptor shall forward the report to the HID class driver and start a timer to expire after *aplcIdleRateGuardTime*. This *guard* timer prevents the HID adaptor constantly sending reports to the HID class driver if communication is lost with the remote HID class device.

The HID adaptor shall then resend the last report received from the remote HID class device to the HID class driver at the specified idle rate. Each time a report is received from the remote HID class device, the HID adaptor shall update the report it sends to the HID class driver with the contents of the received report and re-kick the guard timer, again to expire after *aplcIdleRateGuardTime*.

If the guard timer expires, the HID adaptor shall assume it has temporarily lost contact with the remote HID class device and notify the HID class driver accordingly. If a NULL report is defined in the proxy table corresponding to the identifier of the report being sent, the HID adaptor shall send it to the HID class driver. The HID adaptor shall send no further repeated reports to the HID class driver until contact is reestablished with the remote HID class device.

If the value of *aplDeviceIdleRate* is equal to zero, when the next report is received from the associated remote HID class device, the HID adaptor shall forward the report to the HID class driver and start a timer to expire after *aplcIdleRateGuardTime*. This *guard* timer allows the HID adaptor to determine if communication is lost with the remote HID class device.

Each time a report is received from the remote HID class device, the HID adaptor shall forward the report to the HID class driver and re-kick the guard timer, again to expire after *aplcIdleRateGuardTime*.

If the guard timer expires, the HID adaptor shall assume it has temporarily lost contact with the remote HID class device and notify the HID class driver accordingly. If a NULL report is defined in the proxy table corresponding to the identifier of the last report sent, the HID adaptor shall send it to the HID class driver.

5.2.3.6 Get_Protocol request

The HID adaptor shall extract the current protocol for the appropriate HID class device from the proxy table (*aplCurrentProtocol* attribute) and return its value to the HID class driver.

5.2.3.7 Set_Protocol request

If the request is directed to a HID class device which supports the boot interface subclass (according to its *aplHIDDeviceSubclass* attribute), the HID adaptor shall update the current protocol of the appropriate HID class device in the proxy table (*aplCurrentProtocol* attribute). If the HID class device does not support the boot interface subclass this request shall be ignored.

When the boot protocol is requested, the HID adaptor shall truncate incoming mouse or keyboard reports to the first 3 and 8 bytes, respectively, before passing them to the HID class driver. At other times, the HID adaptor shall pass incoming reports directly to the HID class driver without truncation.

5.3 HID class device operation

A human interface device intending to conform to the RF4CE ZID profile shall define a HID class device component, as illustrated in Figure 1. A HID class device shall be constructed from either an RF4CE controller or an RF4CE target. The HID class device component is described in this section.

5.3.1 HID class device state machine

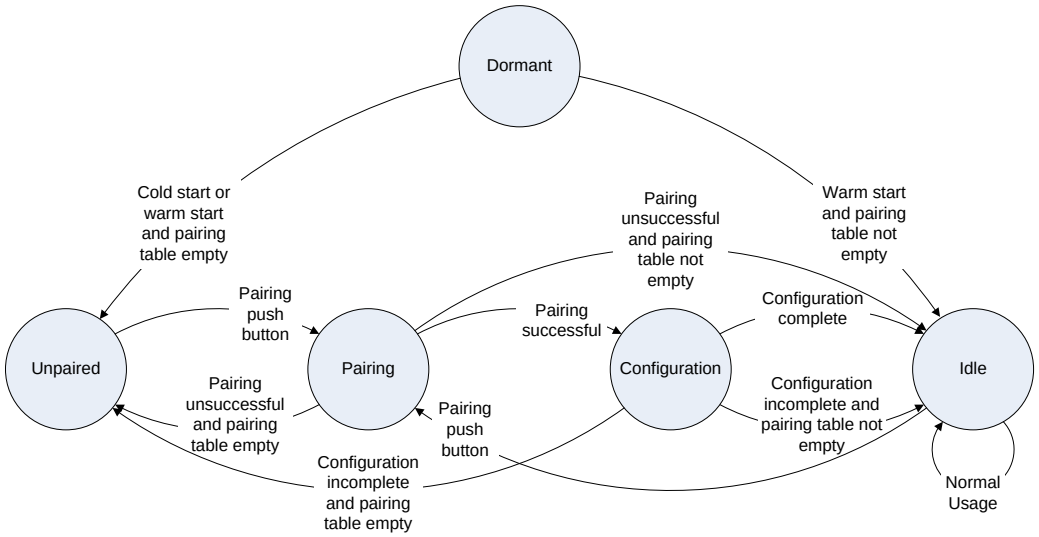

Figure 12 – HID class device state machine

A HID class device should implement the state machine shown in Figure 12 and summarized in Table 17.

Table 17 – HID class device state transitions

State	Action	Condition	Next state
Dormant	HID class device performs a cold or warm start, at the discretion of the application, in order to initialize itself.	Cold start or warm start and pairing table is empty.	Unpaired
		Warm start and pairing table is not empty.	Active
Unpaired	HID class device is ready to be paired with a host device but cannot otherwise communicate.	Pairing push button pressed.	Pairing
Pairing	HID class device performs the pairing procedure.	Pairing is successful.	Configuration
		Pairing is unsuccessful and the pairing table is empty.	Unpaired
		Pairing unsuccessful and the pairing table not empty.	Idle

State	Action	Condition	Next state
Configuration	HID class device interrogates the HID adaptor to determine its compatibility. If compatible, the HID class device then communicates its HID related attributes (i.e., those with an *aplHID* prefix) to the HID adaptor to allow it to proxy the device. If the HID adaptor is not compatible or all of the HID related attributes are not successfully communicated to the HID adaptor, the HID device removes its entry with the HID adaptor from the pairing table.	All HID related attributes successfully communicated.	Idle
		All HID related attributes are not successfully communicated to the HID adaptor and the pairing table is not empty.	Idle
		All HID related attributes are not successfully communicated to the HID adaptor and the pairing table is empty.	Unpaired
Idle	HID class device switches into its normal operating state during which the device can instigate reports to the host. If desired, the application may choose to enable the receiver under application specific conditions.	Pairing push button pressed.	Pairing
		Device undergoes normal usage, e.g. button pressed or pointer moved.	Idle

5.3.1.1 Dormant state

The *dormant* state represents the state of the HID class device before it is initialized using either the RF4CE cold or warm start procedures. Typically, if the HID class device has not been used before or a hard reset has occurred, the device should start in this state and the cold start procedure should be followed. Conversely, if the HID class device has been power cycled from any other state, the device should transition to this state and the warm start procedure should be followed. In this state, the transceiver is not enabled so the node is effectively dormant.

On first activation of the HID class device, e.g. when batteries are first inserted, the HID class device shall request that the RF4CE stack perform the cold start procedure by issuing the NLME-RESET.request primitive with the SetDefaultNIB parameter set to TRUE. On receipt of a successful reset response notification from the NLME, via the NLME-RESET.confirm primitive, the HID class device shall initialize the NIB attributes as described in [R5] . The device then transitions unconditionally to the *unpaired* state.

On activation of the HID class device following a power cycle, it shall instigate the RF4CE warm start procedure by issuing the NLME-RESET.request primitive to the NLME with the SetDefaultNIB parameter set to FALSE and wait for a successful reset response notification from the NLME, via the NLME-RESET.confirm primitive. If the pairing table is empty, the HID class device then transitions to the *unpaired* state. If the pairing table is not empty, the HID class device then transitions to the *idle* state.

5.3.1.2 Unpaired state

The *unpaired* state represents the state of the device after it is initially activated or when the RF4CE pairing table is empty. In this state the node is powered but the transceiver is not enabled.

On receipt of a pairing push button stimulus from the user, the device transitions to the *pairing* state.

5.3.1.3 Pairing state

The *pairing* state represents the state of the device while it is undergoing the RF4CE pairing procedure in order to connect to another RF4CE node with which to communicate.

On entry into the *pairing* state, the HID class device shall instigate the discovery/pairing procedure with the network layer by issuing the NLME-DISCOVERY.request primitive to the NLME.

For the discovery, the HID class device shall set the following parameters: the discovery duration shall be set to 0x00186a (100ms); the profile identifier list disclosed as supported by the node shall contain at least the value 0x02 (the ZID profile identifier); the list of profile identifiers by which incoming discovery response command frames are matched shall contain at least the value 0x02 (the ZID profile identifier) and the list of device types supported by the node shall contain at least one entry taken from the RF4CE device type list [R4] .

On receipt of an unsuccessful confirmation of discovery from the network layer, via the NLME-DISCOVERY.confirm primitive, the HID class device shall terminate the procedure and perform no further processing.

On receipt of a successful confirmation of discovery from the network layer with more than one node descriptor, via the NLME-DISCOVERY.confirm primitive, the HID class device shall discard the information and terminate the procedure, performing no further processing.

On receipt of a successful confirmation of discovery from the network layer, via the NLME-DISCOVERY.confirm primitive, containing exactly one node descriptor, the HID class device shall instigate a pair request by issuing the NLME-PAIR.request primitive to the NLME, ensuring that the KeyExTransferCount parameter is set to *aplKeyExchangeTransferCount*.

On receipt of a successful confirmation of pairing from the network layer, via the NLME-PAIR.confirm primitive, the HID class device shall be considered paired to the remote node with the indicated reference into the pairing table. The node then transitions to the *configuration* state.

If any part of the pairing procedure fails with the pairing table still having at least one entry remaining, the HID class device transitions to the *idle* state. If any part of the pairing procedure fails leaving the pairing table empty, the HID class device transitions to the *unpaired* state.

5.3.1.4 Configuration state

The *configuration* state represents the state of the HID class device after it has successfully paired with another RF4CE node but before it has identified itself to the paired host as a HID class device. The HID class device shall remain in this configuration mode until all of its HID related attributes are communicated to the HID adaptor.

The HID class device first needs to interrogate the HID adaptor to determine its compatibility. To do this, the HID class device shall generate and transmit a get attributes command frame, requesting the *aplZIDProfileVersion*, *aplHIDParserVersion*, *aplHIDCountryCode*, *aplHIDDeviceReleaseNumber*, *aplHIDVendorId* and *aplHIDProductId* attributes, to the HID adaptor. The get attributes command frame shall be generated and transmitted to the HID adaptor within a maximum duration of *aplcMaxConfigPrepTime*. The HID class device shall then request that the NWK layer enable its receiver and wait either for a maximum duration of *aplcMaxResponseWaitTime* or until a get attributes response command frame is received from the HID adaptor. If any other command frame is received, it shall be discarded.

If a get attributes response command frame is received from the HID adaptor, the HID class device shall determine whether it is compatible, based on the attributes. The procedure for determining whether the HID adaptor is compatible is beyond the scope of this specification. If the HID adaptor is not compatible with the HID class device or a get attributes response command frame is not received from the HID adaptor, the HID class device shall consider the configuration a failure.

If the HID adaptor is compatible, the HID class device shall communicate those attributes required to allow the HID adaptor to construct appropriate USB descriptors, as listed in Table 16, or to determine device specific properties. To do this, the HID class device shall generate and transmit push attributes command frames. Each push attribute command frame shall be generated and transmitted to the HID adaptor within a maximum duration of *aplcMaxConfigPrepTime*, as illustrated in Figure 11.

The HID class device shall first push all the attributes listed in Table 16 with the exception of any non-standard descriptor components, *aplHIDNonStdDescCompSpec-i* in a single push attributes command frame. If the number of standard descriptor components supported (*aplHIDNumStdDescComps*) is equal to zero, the HID class device shall not include the *aplHIDStdDescCompsList* attribute.

If the number of non standard descriptor components supported (*aplHIDNumNonStdDescComps*) is greater than zero, the HID class device shall then push the descriptor component specifications defined by the HID class device, *aplHIDNonStdDescCompSpec-i*, where *i* has a value between 1 and *aplHIDNumNonStdDescComps*. Due to their potential length, each descriptor component specification shall be pushed using separate push attributes command frames, fragmented as necessary.

If the number of NULL reports supported (*aplNumNullReports*) is greater than zero, the HID class device shall generate and transmit successive set report command frames, one for each NULL report supported on the HID class device.

If all relevant attributes and NULL reports are successfully communicated to the HID adaptor, the HID class device shall generate and transmit a configuration complete command frame to the HID adaptor. The HID class device shall then request that the NWK layer enable its receiver and wait either for a maximum duration of *aplcMaxResponseWaitTime* or until a command frame is received from the recipient. If a generic response command frame is not received within *aplcMaxResponseWaitTime*, the HID class device shall terminate the operation and disable its receiver. If any other command frame is received, it shall be discarded.

If the configuration attempt fails due to some communications error or if the received generic response command frame indicated a configuration error, the HID class device shall remove the entry for the HID adaptor from its pairing table and then determine if the pairing table is now empty. If the pairing table is empty, the HID class device transitions to the *unpaired* state. If the pairing table is not empty, the HID class device transitions to the *idle* state.

Report data shall not be transmitted by the HID class device if it has failed to successfully communicate its full set of attributes to the HID adaptor.

5.3.1.5 Idle state

The *idle* state represents the normal state of the HID class device at the point it is paired with at least one other RF4CE node and has exchanged its HID information with the HID adaptor. During this state, the HID class device shall transmit report data command frames at the discretion of the application.

When the user interacts with the HID class device, e.g. the mouse is moved or a keyboard button is pressed, the application shall construct an appropriate HID report and the HID class device shall transmit via a report data command frame to the paired host. The application can choose to send the report to the host using either the control channel or the interrupt channel.

If the user interaction is not stopped (e.g. a key on a keyboard is not released) within *aplReport-RepeatInterval* (where *aplReportRepeatInterval* is non zero), the HID class device shall retransmit the last report data command frame to the HID adaptor. This procedure shall be repeated at a rate of *aplReportRepeatInterval* until the user interaction is stopped.

After transmitting a report data command frame, the HID class device remains in the *idle* state.

If the HID class device wishes to periodically check in with the HID adaptor in order to determine whether it has any relevant data to send, the HID class device should follow the procedures described in sub-clause 5.1.3.

On receipt of a pairing push button stimulus from the user, the HID class device transitions to the *pairing* state.

5.4 Security

5.4.1 ZID profile command frames

All ZID profile command frames other than report data command frames shall be transmitted with security enabled using the control pipe, i.e. the command frame shall be passed to the NWK layer with

the security mode bit of the transmission options set to 1. If a ZID profile command frame, which is not a report data command frame, is received without security enabled it shall be discarded.

5.4.2 Report data command frames

Keyboard and keypad devices (i.e. those that support the standard keyboard report in sub-clause 9.2) that can potentially transmit sensitive information, e.g., credit card numbers, passwords or confidential emails, shall transmit report data command frames using unicast transmission and with security enabled, i.e. the report shall be passed to the NWK layer with the security mode bit of the transmission options set to 1.

If such a report data command frame (which contains at least one report data record requiring security) is received without security, then the full report data command frame shall be discarded.

Other devices shall use security at the discretion of the application.

6 Multitouch position and gesture reports

6.1 Introduction

For the purposes of this specification, we consider a 2D touch device to be either a *touchpad* (opaque 2D sensor, such as the trackpad on a laptop) or a *touchscreen* (transparent 2D sensor laid over a text or graphics based display device, such as the touchscreen on a smart phone). Processing of touch information before it is reported to the host may be done entirely on the touch device, entirely on the HID class device, or a combination of the two.

Figure 13 shows the primary attributes of a single point of contact on the touch device. At any time, there may be zero, one, or multiple such points of contact. All such points, and changes in their state, must be unambiguously reported to the host.

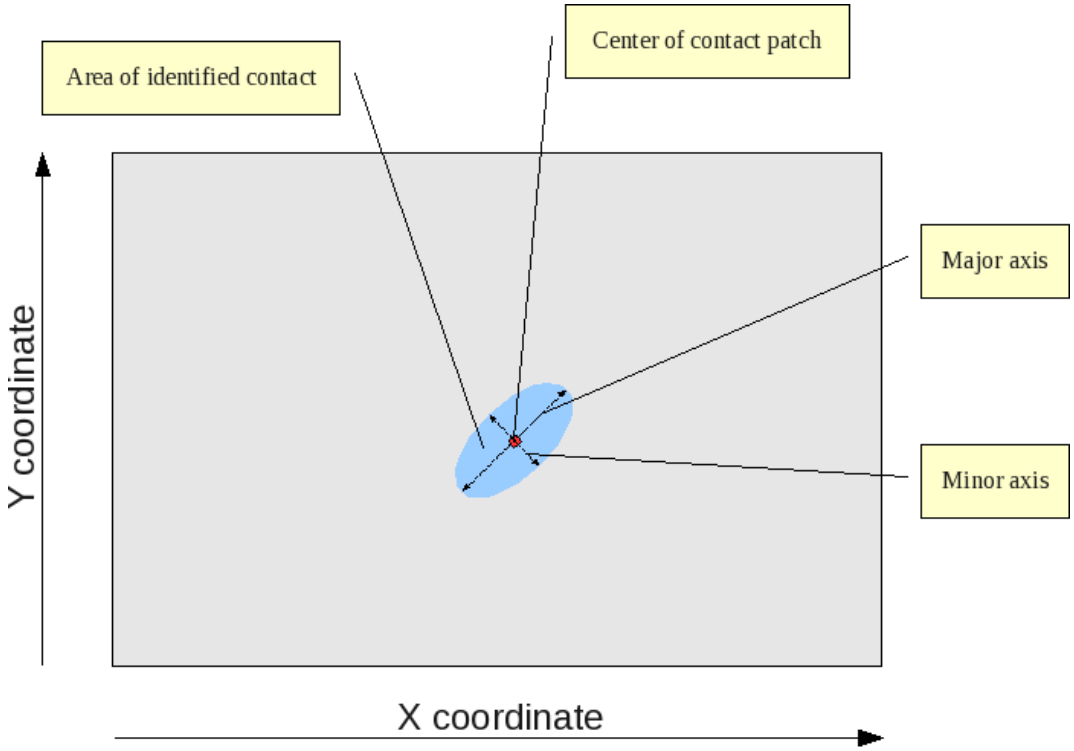

Figure 13 – Attributes of a single contact

Currently available multitouch sensors are capable of reporting anywhere from 2 to 10 contacts, depending on the particular sensor and its application.

6.1.1 General reporting principles

Continuous reporting of touch and gesture data for all possible contact points and gestures would entail a prohibitive cost in terms of power consumption and bandwidth. To eliminate this, the remote will only report changes in the state of the touch and gesture data.

As an example, let's assume a report period of 12.5 milliseconds, or 80 reports per second. A finger lands on the touch sensor at position X_1,Y_1, remains motionless for half a second, move slightly to position X_2,Y_2, remains motionless for another half second, and then departs from the touch sensor exactly one second after it first arrived. If contact data reports were sent continuously during this time, a total of 80 reports (880 bytes) would be transmitted.

For multitouch, the amount of data transmitted can be quite substantial, scaling linearly with the number of contact reports supported. If the touch sensor supported reports for 5 contacts, in the continuous reporting scenario it would send 5 reports of 11 bytes each x 80 report periods = 4400 bytes of data during a single contact lasting one second.

Because we only transmit the reports that indicate changes in state, we need far fewer reports to describe the interesting parts of the finger's behavior:

1. finger arrives at X_1,Y_1
2. finger moves to X_2,Y_2
3. finger departs, last location was X_2, Y_2

Only three reports are sent, for a total of 33 data bytes. Additional synchronization overhead would amount to one byte per reporting period, so the total transmitted would be 36 bytes.

Note: these figures do not include any USB or RF4CE overhead on the reports.

6.1.2 Multitouch reporting

A report from a multitouch capable touch device is made up of the following information.

- The report index, which increases sequentially and allows the host to detect dropped/missed reports;
- The number of contacts being reported;
- Data for each contact being reported.

Reports are issued only when the data for at least one contact changes. All present or departing contacts are reported at this time.

6.1.3 Gesture reporting

Gesture recognition may be done by the touch sensor, by the host, or by both. There are advantages and disadvantages to each – for example, the touch sensor typically has the best information about the location and movement of the finger(s), but is also generally very tightly constrained in terms of memory and CPU resources. The host typically has CPU cycles and RAM to spare, and has much better information about the context in which the gesture is happening, but has less or slightly delayed information about finger locations and motion.

In the context of an RF4CE implementation, this division of labor is somewhat more complicated, since the microprocessor in the remote control is a host (from the touch sensor's point of view), but it is not the final consumer of the touch or gesture information (that is done by the host system). For the purposes of this specification we can consider the touch sensor + remote micro as a unit, since we are primarily concerned with communicating data from the remote to the host.

Something to bear in mind is that although many touch sensors are capable of recognizing gestures, not all such sensors support the entire suite listed below. For example, a sensor might support tap recognition, but not more complex gestures such as flick or rotate. Additionally, we need to consider that some touch sensors will support additional gestures outside those listed below.

6.1.4 Gestures

As a rough rule of thumb, gestures can be divided into two kinds:

- transitory gestures, such as taps and flicks, that are quick enough to be handled within a single report; and
- continuous gestures, such as scrolling, which need to be handled across several reports.

Table 18 summarizes some of the most frequently used gestures and the data needed to make use of them.

Table 18 – Frequently used gestures

Gesture	Data	Description	Remarks
Tap	Location X, Y – 12 bits each Number of fingers – 3 bits	Finger arrives on touch sensor, and departs soon after without moving very far.	Often used to activate a control.
Tap and a half	Location X, Y – 12 bits each Number of fingers – 3 bits	Finger arrives on touch sensor, departs soon after without moving very far, returns very soon in nearly the same location.	Often used to initiate a dragging operation.
Double tap	Location X, Y – 12 bits each Number of fingers – 3 bits	Finger arrives on touch sensor, departs soon after without moving very far, returns very soon in nearly the same location, and departs again soon after.	Often used to open or activate applications or folders.
Long tap	Location X, Y – 12 bits each Number of fingers – 3 bits	Contact arrives on touch sensor, and departs a moderate time after without moving very far.	Sometimes called 'press'. Often used to activate the secondary function of a control. There can be issues with disambiguating this from other gestures (such as plain tap).
Flick	Direction – 3 bits. Number of fingers – 3 bits Distance in screen units – 12 bits.	Contact arrives on the touch sensor, moves rapidly and mostly parallel to the X or Y axis, and departs. For very large sensors, flick may also be interpreted along the diagonal axis as well. This capability is optional.	Often used to switch between UI 'pages', or to initiate scrolling.
Linear scroll	Direction – 3 bits. Number of fingers – 3 bits Distance in screen units – 12 bits.	Contact arrives near the edge of the touch sensor, and moves up/down or left/right along that edge.	This is one of the few gestures where the expected UI action can be known to the touch sensor.
Circular scroll	Direction – Clockwise or counter clockwise. 2 bits. Number of fingers – 3 bits Distance in screen units – 12 bits.	Finger moves in a roughly circular direction on the surface of the touch sensor.	Initiation of circular scrolling can be a tricky issue.
Pinch	Relative motion of fingers – closer together or further apart - 1 bit. Distance between fingers in screen units – 12 bits. Location of fingers or center of pinch (optional).	Two fingers arrive on the surface of the touch sensor, and move towards/away from each other	Usually used for zoom functions, but can be used for other operations (for example, open or close of GUI containers).
Rotate	Direction of rotation – clockwise or counter clockwise. 1 bit. Magnitude of rotation.	Exactly two fingers arrive on the surface of the touch sensor, and move clockwise or counterclockwise around an approximately common center while maintaining the same distance.	Usually used to rotate pictures, but can be used for other operations (for example, operation of a 'rotary' control on a GUI).

This is not meant to be an exhaustive set of all gestures that are (or have been) used anywhere.

For the tapping and scrolling gestures, 3 bits are allocated for the finger count, allowing up to 7 fingers to be reported for such gestures. In practical terms, we are unlikely to see more than 4 fingers for these gestures, with 5 being the sensible maximum.

6.1.5 Advanced gestures

Many modern GUIs also recognize certain advanced gestures, such as drawing an @ symbol to launch email. These gestures are both implementation dependent and highly context sensitive - their recognition and processing is best done on the host system. For this reason, this class of gesture does not come within the scope of the RF4CE ZID specification.

6.1.6 Continuous vs transitory gestures

Tap, tap and a half, double tap, long tap and flick are transitory gestures. That is, the user's action can be encapsulated in a single report.

Linear and circular scroll, pinch, and rotate are continuous gestures. That is, a single action by the user is spread across the multiple reports from the touch sensor/remote. This makes it desirable to transmit an "end of gesture" report to indicate to the host that the user is no longer engaged in a particular gesture sequence.

For Linear Scroll and Circular Scroll gestures, this is done by setting the finger count field to 0. In this case the data fields of the gesture may be set to zero, or may be used to indicate any additional actions that may have occurred between the last report and the termination of the gesture.

For Pinch and Rotate gestures, this is done by clearing the "finger present" bit in the report. In this case the data fields of the gesture may be set to zero, or may be used to indicate any additional actions that may have occurred between the last report and the termination of the gesture.

The host needs to be aware that the user may pause during the execution of a continuous gesture, sometimes for lengthy periods of time. For example, when using a circular scrolling gesture to scan through a lengthy text or menu, the user may scroll forward to an area of interest, pause a while to read the information on the screen without lifting their finger from the sensor, and then continue scrolling further along or even reverse direction to revisit some previous part of the text or menu.

6.2 Touch sensor properties report

In order for multitouch and gesture report data to be meaningful, the host must know certain properties of the touch sensor, such as resolution, reported coordinate limits, position of the coordinate system origin, whether gestures are supported, number of finger reports supported, and so on.

The touch sensor properties report is 7 bytes in length and shall be formatted as illustrated in Figure 14.

All fields marked *Reserved* shall be set to zero and ignored on reception.

This report shall be communicated to the host via a report data command frame before any touch reports are sent.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Gestures	Reliable index	Origin		Number of additional contacts			
1	$Resolution_x$							
2	$Resolution_y$							
3	$MaximumCoordinate_x[11:4]$							
4	$MaximumCoordinate_x[3:0]$				$MaximumCoordinate_y[11:8]$			
5	$MaximumCoordinate_y[7:0]$							
6	*Reserved*					Shape		

Figure 14 – Format of the touch sensor properties report

6.2.1 Origin field

The origin field is 2-bits in length and indicates the orientation of the coordinate system used by the touch sensor. This field shall take one of the values listed in Table 19.

Table 19 – Values of the origin field

Origin value b_1b_0	Description
00	Origin is at lower left of the touch sensor. X coordinates increase as you move from left to right; Y coordinates increase as you move from bottom to top. This is the default setting for most touch sensors.
01	Origin is at upper left of the touch sensor. X coordinates increase as you move from left to right; Y coordinates increase as you move from top to bottom.
10	Origin is at upper right of the touch sensor. X coordinates increase as you move from right to left; Y coordinates increase as you move from top to bottom.
11	Origin is at lower right of the touch sensor. X coordinates increase as you move from right to left; Y coordinates increase as you move from bottom to top.

6.2.2 Gestures field

The gestures field is 1-bit in length and indicates the presence or absence of gesture support on the touch sensor/remote. If this field is set to 1, gesture reporting is supported. For the most part, the host can determine which gestures are supported by the report descriptors provided. In some cases, additional properties reports will be needed to indicate further details relating to gesture capabilities.

6.2.3 Number of additional contacts field

All touch sensors report position data for at least one contact.

The number of additional contacts field is 4-bits length and indicates the number of additional contacts that are reported beyond that one contact. The valid range for this field is [0…15]. The total number of contacts reported by the touch sensor/host is computed as:

$$TotalReportedFingers = 1 + NumberOfAdditionalContacts$$

6.2.4 Reliable index field

The reliable index field is 1-bit in length and indicates whether the touch sensor is capable of tracking individual contact points reliably from one report to the next, and can assign a reliable index to each contact point.

When this bit is set to 1, the host can rely on the contents of the Contact Index field of the Contact Data Report to identify and track contact data across time.

If this bit is set to 0, then the touch sensor is not capable of reliably identifying and tracking contact points from one report to the next. In this case, the Contact Index field will not reliably identify contact information, and the host must assume responsibility for tracking individual contact points across time.

6.2.5 Resolution$_x$ and resolution$_y$ fields

The resolution$_x$ and resolution$_y$ fields are 7-bits in length and indicate the number of coordinate units on each axis corresponding to the distance of 1mm on the touch sensor surface.

6.2.6 Maximum coordinate$_x$ and maximum coordinate$_y$ fields

The maximum coordinate$_x$ and maximum coordinate$_y$ fields are 12-bits in length and indicate the upper bound, in the range [1…4095], on the coordinates reported on each axis by this particular touch sensor. In conjunction with the Resolution, the host can use these properties to determine the size and aspect ratio of the touch sensor. This information is important when features such as scrolling, gestures, touch regions and so on are implemented on the host side.

6.2.7 Shape field

The shape field is 3-bits in length and indicates the shape of the touch sensor. This field shall take one of values listed in Table 20.

Note that except for standard rectilinear sensors, or circular sensors with a 1:1 aspect ratio, this field is only a rough indicator as to the shape of the touch surface.

Table 20 – Values of the shape field

Shape value $b_2b_1b_0$	Shape	Comments
000	Rectilinear	Touch surface is square or rectangular.
001	Circle	Touch surface is circular (if aspect ratio is 1:1) or oval/elliptical (for all other aspect ratios).
010	Trapezoid	Touch surface is trapezoidal
011 – 110	*Reserved*	-
111	Irregular	Touch surface is an irregular shape. In the case, additional proprietary queries should be provided to enable the host to adequately characterize the touch surface boundaries.

6.3 Tap support properties report

The host can determine whether the remote supports gestures by checking the Gestures bit in the Touch Sensor Properties report, and then using the various report descriptors to determine just which gestures are supported. In some cases, though, it is desirable for the host to determine which of the available tap gestures are supported.

The tap support properties report is 4 bytes in length and shall be formatted as illustrated in Figure 15.

All fields marked *Reserved* shall be set to zero and ignored on reception.

This report shall be communicated to the host via a report data command frame before any tap reports are sent.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	*Reserved*				Long tap	Double tap	Tap-and-a-half	Single tap
1	*Reserved*							
2	*Reserved*							
3	*Reserved*							

Figure 15 – Format of the tap support properties report

Each bit set in this report indicates that the corresponding tap type is supported by the touch sensor.

6.3.1 Long tap field

The long tap field is 1-bit in length and specifies the support for the long tap gesture. If this field is set to 1, a long tap gesture is supported by the touch sensor. Otherwise, a long tap gesture is not supported.

6.3.2 Double tap field

The double tap field is 1-bit in length and specifies the support for the double tap gesture. If this field is set to 1, a double tap gesture is supported by the touch sensor. Otherwise, a double tap gesture is not supported.

6.3.3 Tap-and-a-half field

The tap-and-a-half field is 1-bit in length and specifies the support for the tap-and-a-half gesture. If this field is set to 1, a tap-and-a-half gesture is supported on the touch sensor. Otherwise, a tap-and-a-half gesture is not supported.

6.3.4 Single tap field

The single tap field is 1-bit in length and specifies the support for the single tap gesture. If this field is set to 1, a single tap gesture is supported on the touch sensor. Otherwise, a single tap gesture is not supported.

6.4 Sync report

In order to conserve power and bandwidth, only the Gesture Reports and Contact Data Reports that contain information about the change in state of the system are transmitted. This implies that the host needs a method to determine when all data for a given reporting period has been transmitted. To facilitate this, the remote will transmit a synchronization report at the end of each reporting period, after it has transmitted any/all Gesture and Contact Data for that period.

The Sync Report is one byte in length and shall be formatted as illustrated in Figure 16.

All fields marked *Reserved* shall be set to zero and ignored on reception.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	*Reserved*			Gesture	Contact count			

Figure 16 – Format of a standard sync report

The data in the Sync Report can be used as a check on the previously received data, to help detect loss of reports. The fields are described in the following sub-clauses.

6.4.1 Gesture field

The gesture field is 1-bit in length and specifies the status of the gesture activity. If this field is set to 1, gesture activity was detected during the previous reporting period. As a result, the host should have received one or more Gesture Reports since the previous Sync Report.

If this field is set to 0, no gesture activity was detected during the previous reporting period. The host should have received no Gesture Reports since the last time it received a Sync Report.

6.4.2 Contact count field

The contact count field is 4-bits in length and specifies a count of contact patches for which activity was reported during the previous reporting period. The host should have received this number of Contact Data Reports since the previous Sync Report.

Note that this is **not** the same as the number of contacts currently active on the sensor, but only the number of contacts for which a change in position or state has been reported.

6.5 Contact data report

The contact data report is 11 bytes in length and shall be formatted as illustrated in Figure 17.

Note that this data is for each contact. A report describing 4 contacts would contain 4 different instances of this data – one for each contact.

All fields marked *Reserved* shall be set to zero and ignored on reception.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Contact type				Contact index			
1	*Reserved*						Contact state	
2	Major axis orientation							
3	Pressure							
4	$Location_x$[7:0]							
5	$Location_y$[3:0]				$Location_x$[11:8]			
6	$Location_y$[11:4]							
7	Major axis length[7:0]							
8	Major axis length[15:8]							
9	Minor axis length[7:0]							
10	Minor axis length[15:8]							

Figure 17 – Format of a standard contact data report

6.5.1 Contact state field

The contact state field is 2-bits in length and is used to indicate the presence of a contact, and the reliability of the contact information.

When a contact arrives on the touch device, it is reported as being in the arriving state. The number of contacts being reported to the host is incremented by one.

The arriving state is maintained for only a single report, after which the contact's state is reported as accurate. Most contacts will then remain in the accurate state for the entire duration that the sensor is touched.

When a contact departs from the sensor surface, a final report is issued with a state of no contact. The reported position and other attributes for that contact are those of the previous report. On the subsequent report, the number of contacts being reported will be decremented by one, and no further report information will be sent for this contact.

The inaccurate contact state is a special state that indicates that the touch sensor was not able to reliably determine the location and/or other attributes of the contact. This state signifies to the host that "yes, there is a finger around here somewhere, but we can't completely characterize it".

The contact state field shall be set to one of the values listed in Table 21.

Table 21 – Values of the contact state field of the contact data report

Contact state value b_1b_0	Description
00	No contact
01	Contact arriving
10	Contact accurate
11	Contact inaccurate

6.5.2 Contact index field

The contact index field is 4-bits in length and specifies an index indicating the contact being described.

Touch devices or controllers that are capable of reliably tracking multiple contacts across time should assign an appropriate index to each contact. This value must be in the range of [0…*n*], where *n* is the maximum number of contacts that can be tracked by the touch sensor. Software on the host can then assume that a series of contact reports with a given index represent data for the same physical finger/pen/whatever, until that contact departs.

Touch devices or controllers that are not capable of reliably tracking multiple contacts across time must assign an index of zero to each contact. Software on the host must then assume the duty of tracking contacts appropriately.

6.5.3 Contact type field

The contact type field is 4-bits in length and specifies the type of the contact being described. This field shall be set to one of the values listed in Table 22.

Table 22 – Values of the contact type field of the contact data report

Contact type value $b_3b_2b_1b_0$	Description
0000	Finger
0001	Pen
0010 – 1111	*Reserved*

6.5.4 Location$_x$ and location$_y$ fields

The location$_x$ and location$_y$ fields are 12-bits in length and specify the horizontal and vertical, respectively, axis position of the centre of the contact patch.

Although these fields have a 12 bit range, most touch sensors are unlikely to use the full 12 bits. For example, a 4x3 aspect ratio sensor might report X in the range [0…2400] and Y in the range [0…1800].

This implies that the host must be able to query the sensor to determine the maximum reported coordinate on each axis, as well as the per-axis resolution in terms of the number of coordinate units per millimeter.

6.5.4.1 Indeterminate finger position

Position values at one or the other extreme of the coordinate range indicate that the contact position is not reliably known. This situation most commonly occurs when a finger or pen is near enough to the boundary of the touch sensor, but does not generate enough input signal to be accurately located.

When X is zero, it indicates the finger is near the low order edge of the X axis (left edge of the sensor in Figure 13), but cannot be located reliably. When X is equal to the MaximumCoordinate$_X$ as reported in the Touch Sensor Attributes, it indicates the finger is near the high order edge of the X axis (right edge of the sensor in Figure 13), but cannot be located reliably.

When Y is zero, it indicates the finger is near the low order edge of the Y axis (bottom edge of the sensor in Figure 13), but cannot be located reliably. When Y is equal to the MaximumCoordinate$_Y$ as reported in the Touch Sensor Attributes, it indicates the finger is near the high order edge of the Y axis (top edge of the sensor in Figure 13), but cannot be located reliably.

6.5.5 Major axis orientation field

The major axis orientation field is an 8-bit signed value and represents the orientation of the major axis of the contact patch relative to the Y axis. It is a signed value, with 0 indicating that the major touch axis is aligned parallel to the Y axis. Negative values indicate the major touch axis is inclined to the left of the Y axis; -128 indicates that the major axis is inclined to the left 90 degrees from vertical. Positive values indicate the major touch axis is inclined to the right of the Y axis; 127 indicates that the major axis is inclined to the right almost-but-not-quite 90 degrees from vertical.

For the contact patch shown in Figure 13, the major axis orientation would be positive.

6.5.6 Major and minor axis length fields

The major and minor axis length fields are 13-bits in length and specify the length of the major and minor axis, respectively, of the contact patch.

Although 13 bits are allocated for these fields, in practical terms they cannot exceed sqrt(2)*4096 – the maximum diagonal on a 1:1 aspect ratio sensor that uses the full range of the position reporting fields. For example, this value would be reported for the major axis of a finger laid across such a sensor from one corner to the other.

6.5.7 Pressure field

The pressure field is 7-bits in length and specifies a relative indicator of "how hard" the contact is pressing on the touch surface. This may be determined directly by use of a force sensor, or computed indirectly as an artifact of the contact patch signal.

6.6 Tap gesture report

The tap gesture report is 5 bytes in length and shall be formatted as illustrated in Figure 18.

All fields marked *Reserved* shall be set to zero and ignored on reception.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Type					Finger count		
1	*Reserved*							
2	$Location_x[7:0]$							
3	$Location_y[3:0]$				$Location_x[11:8]$			
4	$Location_y[11:4]$							

Figure 18 – Format of a standard tap gesture report

6.6.1 Type field

The type field is 5-bits in length and specifies the tap gesture that was detected. This field is an unsigned value in the range [0…31] and shall take one of the values listed in Table 23.

Table 23 – Values of the type field of the tap gesture report

Type value $b_4b_3b_2b_1b_0$	Description
00000	Tap
00001	Tap-and-a-half
00010	Double tap
00011	Long tap
00100 – 11111	*Reserved*

6.6.2 Finger count

The finger count field is 3-bits in length and indicates the number of fingers that were detected in the execution of the gesture. This is an unsigned value in the range [0…7]. If 8 or more fingers were detected, the finger count value should be clipped to 7.

A zero finger count is not meaningful for these gestures, and should not ever be sent by the remote.

6.6.3 $Location_x$, $location_y$

The $location_x$ and $location_y$ fields are 12-bits in length and specify the position of the center of the gesture in the range [0…4095] along the X and Y axes, respectively.

6.7 Scroll gesture report

The scroll gesture report, which includes the flick report, is 4 bytes in length and shall be formatted as illustrated in Figure 19.

All fields marked *Reserved* shall be set to zero and ignored on reception.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Type					Finger count		
1	*Reserved*							
2	Distance[3:0]				*Reserved*	Direction		
3	Distance[11:4]							

Figure 19 – Format of a standard scroll gesture report

6.7.1 Type field

The type field is 5-bits in length and indicates the type of scrolling gesture that was detected. This field shall take one of the values listed in Table 24.

Table 24 – Values of the type field of the scroll gesture report

Type value $b_4b_3b_2b_1b_0$	Description
00000	Flick
00001	Linear scroll
00010	Circular scroll
00011 - 11111	*Reserved*

The type of the scrolling gesture determines the interpretation of the direction field.

6.7.2 Finger count field

The finger count field is 3–bits in length and indicates the number of fingers that were detected in the execution of the gesture. This is an unsigned value in the range [0…7]. If 8 or more fingers were detected, the finger count value should be clipped to 7.

For Flick gestures, a finger count of 0 is not meaningful, and should never be sent by the remote.

For Linear Scroll and Circular Scroll gestures, a finger count of 0 indicates that the scroll gesture has completed and the finger/fingers are no longer touching the sensor.

6.7.3 Direction field

The direction field is 3–bits in length. The meaning of the direction field depends on the context of the type field.

6.7.3.1 Flick direction

For flick, the direction field describes the orientation of the gesture motion in terms of compass points, with North indicating the top of the touch sensor in its normal orientation. For flick, the direction field takes one of the values listed in Table 25.

Table 25 – Values of the direction field for flick

Direction value $b_2b_1b_0$	Direction	Motion
000	North	Finger travels primarily in a bottom to top direction sensor.
001	Northeast *(optional)*	Finger travels primarily from lower left to upper right.
010	East	Finger travels primarily from left to right on the sensor.
011	Southeast *(optional)*	Finger travels primarily from upper left to lower right.
100	South	Finger travels primarily in a top to bottom direction sensor.
101	Southwest *(optional)*	Finger travels primarily from upper right to lower left.
110	West	Finger travels primarily from right to left on the sensor.
111	Northwest *(optional)*	Finger travels primarily from lower right to upper left.

All sensors that support flick should support the cardinal directions (North, East, South and West). The ordinal directions (Northeast, Southeast, Southwest and Northwest) are primarily intended for use on larger sensors.

6.7.3.2 Linear scroll direction

For linear scroll gestures, the direction field describes the orientation of the gesture motion in terms of compass points, with North indicating the top of the touch sensor in its normal orientation. For linear scroll, the direction field takes one of the values listed in Table 26.

Table 26 – Values of the direction field for linear scroll

Direction value $b_2b_1b_0$	Direction	Motion
000	North	Finger travels primarily in a bottom to top direction sensor.
001	*Reserved*	-
010	East	Finger travels primarily from left to right on the sensor.
011	*Reserved*	-
100	South	Finger travels primarily in a top to bottom direction sensor.
101	*Reserved*	-
110	West	Finger travels primarily from right to left on the sensor.
111	*Reserved*	-

Linear scroll uses only the cardinal directions (North, East, South and West). For convenience, the same values are used for these directions as are used for Flick gestures. The ordinal directions (Northeast, Southeast, Southwest and Northwest) are not used with Linear Scroll, and the values for those directions are reserved.

6.7.3.3 Circular scroll direction

For circular scrolling, direction of scroll is specified in terms of clockwise or counterclockwise motion. For circular scroll, the direction field takes one of the values listed in Table 27.

Table 27 – Values of the direction field for circular scroll

Direction value $b_2b_1b_0$	Direction	Motion
000	Clockwise	Contact motion is primarily in a clockwise direction.
001	Counter clockwise	Contact motion is primarily in a counter clockwise direction.
010 – 111	*Reserved*	-

For the first report describing a circular scroll action, this is the direction of motion since the initiation of the gesture, for subsequent reports, this describes the motion since the previous report.

6.7.4 Distance field

The distance field is 12-bits in length and specifies the distance, in the range [0…4095], the contacts traveled during the gesture. The units for distance are device coordinate units. Note that this is an unsigned value, and the direction of travel can be determined from the direction field.

For the first report describing a circular scroll action, this is the distance of motion since the initiation of the gesture, for subsequent reports, this describes the distance of motion since the previous report.

6.8 Pinch gesture report

The pinch gesture report is 5 bytes in length and shall be formatted as illustrated in Figure 20.

All fields marked *Reserved* shall be set to zero and ignored on reception.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	*Reserved*						Finger present	Direction
1	Distance[7:0]							
2	*Reserved*				Distance[11:8]			
3	$Center_x$[7:0]							
4	$Center_y$[3:0]				$Center_x$[11:8]			
5	$Center_y$[11:4]							

Figure 20 – Format of a standard pinch gesture report

6.8.1 Finger present field

The finger present field is a 1-bit in length and indicates the presence of fingers on the pad. If this field is set to 1, fingers are detected on the pad. If this field is set to 0, the pinch gesture has been terminated and all other data fields shall be set to zero.

6.8.2 Direction field

The direction field is 1-bit in length and indicates the direction of motion of the two contacts since the initiation of the gesture (if this is the first report in a gesture) or the last pinch gesture report (for subsequent reports for the same action). This field shall take one of the values listed in Table 28.

Table 28 – Values of the direction field of the pinch gesture report

Direction value b_0	Description
0	Fingers are moving closer together.
1	Fingers are moving further apart.

6.8.3 Distance field

The distance field is 12-bits in length and specifies the distance, in the range [0…4095], between the two fingers of the pinch gesture. The units for distance are device coordinate units.

6.8.4 $Center_x$, $center_y$ fields

The $center_x$ and $center_y$ fields are 12-bits in length and specify the position, in the range [0…4095], of the center of the gesture along the X and Y axes respectively.

6.9 Rotation gesture report

The rotation gesture report is 5 bytes in length and shall be formatted as illustrated in Figure 21.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	*Reserved*						Finger present	Direction
1	Magnitude							

Figure 21 – Format of a standard rotation gesture report

6.9.1 Finger present field

The finger present field is 1-bit in length and indicates the presence of fingers on the pad. If this field is set to 1, fingers are detected on the pad. If this field is set to 0, the rotation gesture has been terminated and all other data fields shall be set to zero.

6.9.2 Direction field

The direction field is 1-bit in length and indicates the direction of motion of the two contacts since the initiation of the gesture (if this is the first report in a gesture) or the last rotation gesture report (for subsequent reports for the same action). This field shall take one of the values listed in Table 29.

Table 29 – Values of the direction field of the rotation gesture report

Direction value b_0	Description
0	Fingers are moving clockwise.
1	Fingers are moving counter-clockwise.

6.9.3 Magnitude field

The magnitude field is 8-bits in length and specifies the proportion of a full circle, in the range [0…255], that the contacts have moved since the initiation of the gesture (if this is the first report in an action) or the last rotation gesture report (for subsequent reports for the same action). 0 corresponds to no movement or very little movement, 255 corresponds to nearly 360 degrees of motion.

7 Revision History

Revision	Date	Details	Editor
00	March 2010	First draft	Phil Jamieson
01	March 15th, 2010	Added command descriptions.	Phil Jamieson
02	March 30th, 2010	Added length field in get attributes command. Added attribute descriptions.	Phil Jamieson
03	April, 2010	Added functional description text on the HID adaptor and HID services components.	Phil Jamieson
04	May 2010	Added transmission model for improving the efficiency of low latency transfers.	Phil Jamieson
05	July 2010	Fixed comments discussed in Cincinnati. Added USB descriptor section. Merged in gesture report document.	Phil Jamieson
06	July 2010	Moved Annex B into the main spec and modified the language for specification. Added missing attributes required for descriptor construction. Added text on standard descriptor use.	Phil Jamieson
07	August 2010	Update following basic proof read, correcting anomalies and inconsistencies. General document tidy.	Phil Jamieson
08	August 2010	Update to the interrupt pipe transmission mechanism following inputs from the group.	Phil Jamieson
09	December 2010	Update after comment resolution following TWG letter ballot.	Phil Jamieson
10	January 2011	Update after comment resolution following TWG re-circulation letter ballot.	Phil Jamieson
11	March 8th, 2011	Update following test event comments.	Phil Jamieson
12	March 22nd, 2011	Changes to incorporate the use of the GDP.	Phil Jamieson
13	May 3rd, 2011	Updated following comments from Dresden test event.	Phil Jamieson
14	May 17th, 2011	Added a fragmentation mechanism for non standard descriptor components. Changed Annex B so that collections also include the appropriate header information so they are all self-contained.	Phil Jamieson
15	June 16th, 2011	Added NULL report mechanism.	Phil Jamieson
16	August 13th, 2012	Added reference to commands and features Certification Status document; update of group Chair.	Nick Shepherd
17	August 16th, 2012	Updated following comments on 5.1.2.1, 5.1.5, 5.4.2	Nick Shepherd

8 Annex A USB descriptor generation

If the HID adaptor is connected to a USB class driver on the host, the HID adaptor will need to act as a proxy for the HID class device at the other end of the wireless link. It does this by extracting a set of attributes from the remote HID class device (during the configuration state) with which it can build the set of descriptors required to allow the USB class driver to correctly identify the remote HID class device to the host.

For each remote HID class device, the HID adaptor generates a USB descriptor hierarchy such as the one illustrated in Figure 22.

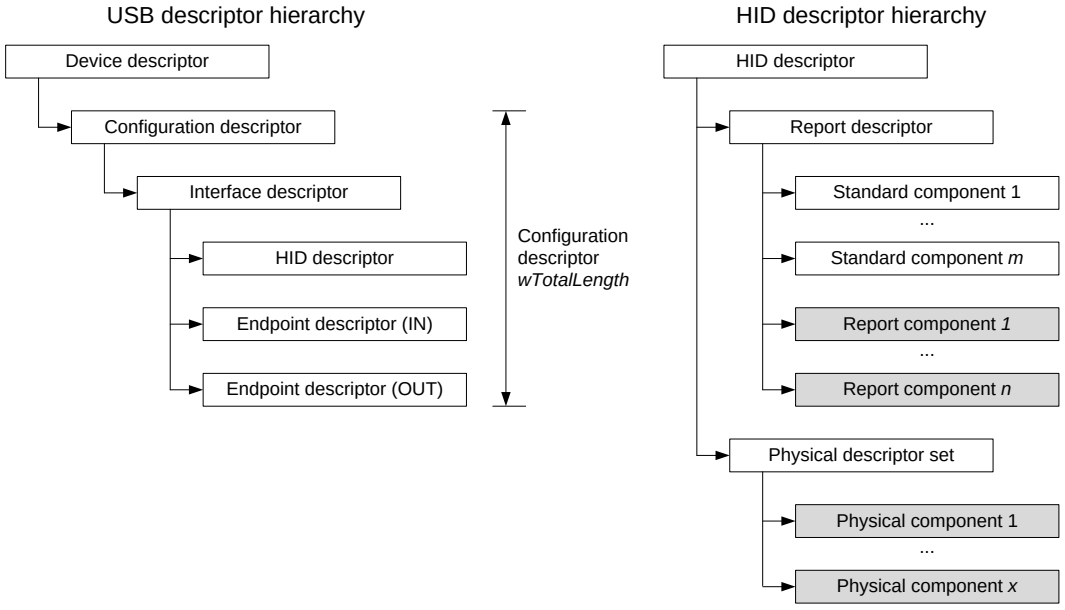

Figure 22 – USB HID descriptor hierarchy

The device, configuration, interface, HID and endpoint descriptors can be generated by the HID adaptor directly from the set of *aplHID* prefixed attributes obtained during configuration from the HID class device (see the following sections). However, the report and optional physical descriptor set are generated according to the standard and non-standard descriptor components indicated by the HID class device.

The mandatory report descriptor is specified as a combination of standard component identifiers and non-standard descriptor components (shaded). The list of standard components should be generated from the list of component identifiers specified in *aplHIDStdDescCompsList*. Each identifier should be expanded into its corresponding component byte string, as specified in clause 9. The list of non-standard components should be generated from the list of report descriptor components contained in *aplHIDNonStdDescCompSpec-i*. Each non-standard report descriptor component should have a unique component identifier (corresponding to a report identifier) and each should be specified as a self-contained component, i.e. with associated USAGE_PAGE, USAGE and COLLECTION headers and END_COLLECTION footer in the same way that the standard components are specified in clause 9. The report descriptor itself should be comprised of the concatenation of all standard components and all non-standard report descriptor components, as recommended in clause 9 (see also the compound example illustrated in Figure 23).

The optional physical descriptor set is specified entirely via the non-standard physical descriptor set components contained in *aplHIDNonStdDescCompSpec-i*. Each non-standard physical descriptor set components should each have a unique component identifier (corresponding to a physical descriptor set identifier) and each should be specified as a self-contained physical descriptor set. The physical

descriptor set itself should be comprised of the concatenation of all non-standard physical descriptor set components.

The report and physical descriptor set descriptor components should be constructed according to the rules detailed in the HID specification [R2] . It is not the responsibility of the HID adaptor to ensure HID compliance of non-standard descriptor components.

Caution: *The following sub-sections suggest sample device, configuration, interface, endpoint and HID descriptors that could be generated by the HID adaptor on behalf of the associated, paired HID class device and should, therefore, not be used verbatim. Care should be taken by an implementation of a HID adaptor to ensure that the purpose of each field is fully understood such that the descriptors are constructed correctly.*

8.1 Device descriptor

Off	Field	Size	Type	Description	Value
0	bLength	1	Number	Size of this descriptor in bytes.	0x12
1	bDescriptorType	1	Constant	Device descriptor.	0x01
2	bcdUSB	2	BCD	USB specification number to which the device complies in the format JJ.M.N, where JJ=major version number, M=minor version number and N=sub-minor version number. E.g., if USB v3.0 is used this value should be set to 0x0300.	0x0300
4	bDeviceClass	1	Class	Class Code (assigned by USB-IF). This should be set to 0x00 indicating that each interface specifies its own class code.	0x00
5	bDeviceSubClass	1	SubClass	Subclass Code (assigned by USB-IF). As bDeviceClass is equal to 0x00, this value should also be set to 0x00.	0x00
6	bDeviceProtocol	1	Protocol	Protocol Code (assigned by USB-IF). This should be set to 0x00 to indicate that protocols are not used on a device basis.	0x00
7	bMaxPacketSize0	1	Number	Maximum packet size for endpoint zero, represented as $2^{bMaxPacketSize0}$. For this specification a value of 0x06 is assumed, giving a packet size of 64 bytes.	0x06
8	idVendor	2	ID	Vendor ID (assigned by USB-IF). This value should be equivalent to *aplHIDVendorId*.	See description
10	idProduct	2	ID	Product ID (assigned by manufacturer). This value should be equivalent to *aplHIDProductId*.	See description
12	bcdDevice	2	BCD	Device release number. This value should be equivalent to *aplHIDDeviceReleaseNumber*.	See description
14	iManufacturer	1	Index	Index of manufacturer string descriptor. This value should be set to 0x00 to indicate it is unused.	0x00
15	iProduct	1	Index	Index of product string descriptor. This value should be set to 0x00 to indicate it is unused.	0x00
16	iSerialNumber	1	Index	Index of serial number string descriptor. This value should be set to 0x00 to indicate it is unused.	0x00
17	bNumConfigurations	1	Integer	Number of possible configurations. A HID class device will typically have just one configuration.	0x01

8.2 Configuration descriptor

Off	Field	Size	Type	Description	Value
0	bLength	1	Number	Size of this descriptor in bytes.	0x09
1	bDescriptorType	1	Constant	Configuration descriptor.	0x02
2	wTotalLength	2	Number	Total length in bytes of data returned for this configuration. This is equal to the combined lengths of the configuration, interface, HID and endpoint descriptors (see also Figure 22). Note that this value includes the length of the HID descriptor but not the lengths of the other HID specific descriptors (i.e. report and physical).	See description
4	bNumInterfaces	1	Number	Number of interfaces. Typically a HID class device has only one interface.	0x01
5	bConfigurationValue	1	Number	A non zero value to use as an argument to the USB device request SetConfiguration() to select this configuration. As a HID class device only has one configuration and the RF4CE ZID profile does not support this device request, this value should be set to 0x01.	0x01
6	iConfiguration	1	Index	Index of the string descriptor which describes this configuration. This value should be set to 0x00 to indicate it is unused.	0x00
7	bmAttributes	1	Bitmap	Bits 4…0 – reserved (must all be set to 0). Bit 5 – remote wakeup. Bit 6 – self powered. Bit 7 – reserved (must be set to 1). As the HID is not physically connected to the bus and must therefore be self powered, this value should be set to 0xc0.	0xc0
8	bMaxPower	1	mA	Maximum power consumption in 2mA units drawn from the bus. As the RF4CE ZID does not draw any power from the bus (as it is not physically connected to the bus), set this value to 0x00. Note that for the configuration descriptor of the HID adaptor itself, this may be a non-zero value.	0x00

8.3 Interface descriptor

Off	Field	Size	Type	Description	Value
0	bLength	1	Number	Size of the descriptor in bytes.	0x09
1	bDescriptorType	1	Constant	Interface descriptor.	0x04
2	bInterfaceNumber	1	Number	Identifying number for this interface. As a HID class device typically only has one interface, this value should be set to 0x00.	0x00
3	bAlternateSetting	1	Number	Value used to select alternative setting. As a HID class device typically does not have alternative interfaces, this value should be set to 0x00.	0x00
4	bNumEndpoints	1	Number	Number of endpoints used for this interface (excluding the default control pipe). This value should be set according to the requirements of the HID class device and shall be set to 0x02 if it supports the optional OUT interrupt pipe. Otherwise, it should be set to 0x01. This value should be equivalent to *aplHIDNumEndpoints*.	See description
5	bInterfaceClass	1	Class	Class code (assigned by the USB-IF). For a HID class device, this should be set to 0x03.	0x03
6	bInterfaceSubClass	1	SubClass	Subclass code (assigned by the USB-IF). For a HID class device, this value should be set to one of the following: 0x00 – None 0x01 – Supports a boot interface. This value should be equivalent to *aplHIDDeviceSubclass.*	See description
7	bInterfaceProtocol	1	Protocol	Protocol Code (assigned by the USB-IF). For a HID class device, this represents the boot protocol and should be set to one of the following: 0x00 – None 0x01 – Keyboard 0x02 – Mouse This value should be equivalent to *aplHIDProtocolCode.*	See description
8	iInterface	1	Index	Index of the string descriptor which described this interface. This value should be set to 0x00 to indicate it is unused.	0x00

8.4 Endpoint descriptor

Off	Field	Size	Type	Description	Value
0	bLength	1	Number	Size of this descriptor in bytes.	0x07
1	bDescriptorType	1	Constant	Endpoint descriptor.	0x05
2	bEndpointAddress	1	Endpoint	The address of this endpoint, encoded as follows: Bits 3...0 – endpoint number. Bits 6...4 – reserved and must be set to zero. Bit 7 – direction: 0 = OUT, 1 = IN. If this endpoint refers to the IN interrupt pipe, this value should be set to 0x81. If this endpoint refers to the optional OUT interrupt pipe, this value should be set to 0x02.	See description
3	bmAttributes	1	Bitmap	Bits 1...0 – transfer type: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt Bits 3...2 – reserved and must be set to zero. Bits 5...4 – usage type: 00 = Periodic 01 = Notification 10 = Reserved 11 = Reserved Bits 7...6 – reserved and must be set to zero. For a HID class device, only interrupt endpoints are defined with a periodic usage type so this value should be set to 0x03.	0x03
4	wMaxPacketSize	2	Number	Maximum packet size this endpoint is capable of sending or receiving. Historically, this value should be set to 1024.	0x0400
6	bInterval	1	Number	Interval for polling endpoint data transfers, in 125us units. This value should range from 1 to 16 and is used as an exponent for a $2^{(bInterval-1)}$ value. This value should be equivalent to *aplHIDPollInterval*.	See description

8.5 HID descriptor

Off	Field	Size	Type	Description	Value
0	bLength	1	Number	Size of this descriptor in bytes. This value should be equal to 9 (one descriptor) plus 3 for each additional descriptor.	See description
1	bDescriptorType	1	Constant	HID descriptor.	0x21
2	bcdHID	2	BCD	Numeric description representing the HID specification release in the format JJ.M.N, where JJ=major version number, M=minor version number and N=sub-minor version number. For the RF4CE ZID profile, HID v1.11 will be used. This value should be equivalent to *aplHIDParserVersion.*	See description
4	bCountryCode	1	Constant	Numeric expression identifying country code of the localized hardware. This value should be equivalent to *aplHIDCountryCode.*	See description
5	bNumDescriptors	1	Number	Numeric expression specifying the number of class descriptors, e.g. report descriptor or physical descriptor sets. This value should be equivalent to $(1 + N_p)$, where N_p is the number of physical descriptor sets specified in *aplHIDDescCompSpec-i*.	See description
6	bDescriptorType	1	Constant	Constant representing the type of the class descriptor. For this version of the specification, this should be set to 0x22 for a report descriptor.	0x22
7	wDescriptorLength	2	Number	Numeric expression that is the total size of the report descriptor.	Application specific
9	[bDescriptorType]...	1	Constant	Constant representing the type of the optional class descriptor. For this version of the specification, this should be set to 0x22 for a report descriptor.	0x22
10	[wDescriptorLength]...	2	Number	Numeric expression that is the total size of the optional descriptor.	Application specific

9 Annex B Standard report components

The RF4CE ZID profile pre-defines a number of standard descriptor components from which device reports descriptors can be compiled to simplify the HID configuration operation; these components are listed in Table 30 and specified in detail in the following sub-sections.

Table 30 – Standard RF4CE ZID profile report components

ID	Description	NULL report?	Reference
0x00	*Reserved*	-	-
0x01	Mouse	Yes	9.1
0x02	Keyboard	Yes	9.2
0x03	Contact data	No	9.3
0x04	Tap gesture	No	9.4
0x05	Scroll gesture	No	9.5
0x06	Pinch gesture	No	9.6
0x07	Rotate gesture	No	9.7
0x08	Sync	No	9.8
0x09	Touch sensor properties	No	9.9
0x0a	Tap support properties	No	9.10
0x0b – 0xff	*Reserved*	-	-

A device that supports these components only needs to set the *aplHIDNumStdDescComps* and *aplHIDStdDescCompsList* attributes which can be extracted during the configuration phase. For example, a device that supports contact data with a pinch gesture would set *aplHIDNumStdDescComps* to 2 and *aplHIDStdDescCompsList* to {0x03, 0x06}.

The HID adaptor would proxy for the device by building a report descriptor as illustrated in Figure 23.

Bytes	Tags

05 0D
09 04
A1 01
85 03
A1 00
09 00
...
C0
C0
	USAGE_PAGE(Digitizers)
	USAGE(Touch Screen)
	COLLECTION(Application)
	  REPORT_ID(3)
	  COLLECTION(Physical)
	  USAGE(Undefined)
	  Rest of contact data component from Figure 31.
	  END_COLLECTION
	END_COLLECTION

05 0D
09 04
A1 01
85 06
A1 00
09 00
...
C0
C0
	USAGE_PAGE(Digitizers)
	USAGE(Touch Screen)
	COLLECTION(Application)
	  REPORT_ID(6)
	  COLLECTION(Physical)
	  USAGE(Undefined)
	  Rest of pinch gesture component from Figure 34.
	  END_COLLECTION
	END_COLLECTION

Figure 23 – Example compound report descriptor using contact data and pinch gesture

9.1 Mouse

9.1.1 Component specification

Figure 24 illustrates the ZID profile standard component for a mouse report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (52)	Tags
05 01	USAGE_PAGE(Generic Desktop)
09 02	USAGE(Mouse)
A1 01	COLLECTION(Application)
85 01	REPORT_ID(1)
09 01	USAGE(Pointer)
A1 00	COLLECTION(Physical)
05 09	USAGE_PAGE(Button)
19 01	USAGE_MINIMUM(Button 1)
29 03	USAGE_MAXIMUM(Button 3)
15 00	LOGICAL_MINIMUM(0)
25 01	LOGICAL_MAXIMUM(1)
75 01	REPORT_SIZE(1)
95 03	REPORT_COUNT(3)
81 02	INPUT(Data, Variable, Absolute) ; Button states
75 05	REPORT_SIZE(5)
95 01	REPORT_COUNT(1)
81 01	INPUT(Constant, Variable, Absolute) ; Reserved bits
05 01	USAGE_PAGE(Generic Desktop)
09 30	USAGE(X)
09 31	USAGE(Y)
15 81	LOGICAL_MINIMUM(-127)
25 7F	LOGICAL_MAXIMUM(127)
75 08	REPORT_SIZE(8)
95 02	REPORT_COUNT(2)
81 06	INPUT(Data, Variable, Relative) ; X & Y
C0	END_COLLECTION
C0	END_COLLECTION

Figure 24 – Component for a standard mouse report

9.1.2 Report format

The mouse component generates the report illustrated in Figure 25.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved					Button3	Button2	Button1
1	X coordinate							
2	Y coordinate							

Figure 25 – Format of a standard mouse report

The format of this report defines the boot protocol for a mouse device. Devices may append additional fields to these reports (defining non-standard descriptors) but the first 3 bytes shall conform to the format illustrated in Figure 25.

9.1.3 Null report

The NULL report for the mouse component is illustrated in Figure 26.

Byte	Value
0	0x00
1	0x00
2	0x00

Figure 26 – Mouse component NULL report

9.2 Keyboard

9.2.1 Component specification

Figure 27 illustrates the ZID profile standard component for a keyboard report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (68)	Tags
05 01	USAGE_PAGE(Generic Desktop)
09 06	USAGE(Keyboard)
A1 01	COLLECTION(Application)
85 02	REPORT_ID(2)
A1 00	COLLECTION(Physical)
05 07	USAGE_PAGE(Keyboard)
19 E0	USAGE_MINIMUM(224)
29 E7	USAGE_MAXIMUM(231)
15 00	LOGICAL_MINIMUM(0)
25 01	LOGICAL_MAXIMUM(1)
75 01	REPORT_SIZE(1)
95 08	REPORT_COUNT(8)
81 02	INPUT(Data, Variable, Absolute) ; Modifier keys
75 08	REPORT_SIZE(8)
95 01	REPORT_COUNT(1)
81 01	INPUT(Constant) ; Reserved byte
75 01	REPORT_SIZE(1)
95 05	REPORT_COUNT(5)
05 08	USAGE_PAGE(LEDs)
19 01	USAGE_MINIMUM(1)
29 05	USAGE_MAXIMUM(5)
91 02	OUTPUT(Data, Variable, Absolute) ; LED status
75 03	REPORT_SIZE(3)
95 01	REPORT_COUNT(1)
91 01	OUTPUT(Constant) ; Reserved bits
75 08	REPORT_SIZE(8)
95 06	REPORT_COUNT(6)
15 00	LOGICAL_MINIMUM(0)
25 65	LOGICAL_MAXIMUM(101)
05 07	USAGE_PAGE(Keyboard)
19 00	USAGE_MINIMUM(0)
29 65	USAGE_MAXIMUM(101)
81 00	INPUT(Data, Array) ; Key arrays
C0	END_COLLECTION
C0	END_COLLECTION

Figure 27 – Component for a standard keyboard report

9.2.2 Report format

The keyboard component generates the input report illustrated in Figure 28.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Modifier keys							
1	Reserved							
2	Keycode 1							
3	Keycode 2							
4	Keycode 3							
5	Keycode 4							
6	Keycode 5							
7	Keycode 6							

Figure 28 – Format of a standard keyboard input report

The keyboard descriptor generates the output report illustrated in Figure 29.

Byte	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Reserved			Kana	Compose	Scroll lock	Caps Lock	Num Lock

Figure 29 – Format of a standard keyboard output report

The formats of these reports define the boot protocol for a keyboard device. Devices may append additional fields to these reports (defining non-standard descriptors) but the first 8 bytes (for the input keyboard report) and first byte (for the output keyboard report) shall conform to the formats illustrated in Figure 28 and Figure 29, respectively.

9.2.3 Null report

The (input) NULL report for the keyboard component is illustrated in Figure 30.

Byte	Value
0	0x00
1	0x00
2	0x00
3	0x00
4	0x00
5	0x00
6	0x00
7	0x00

Figure 30 – Keyboard component NULL report

9.3 Contact data

9.3.1 Component specification

Figure 31 illustrates the ZID profile standard component for a contact data report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (79)	Tags
05 0D	USAGE_PAGE(Digitizers)
09 04	USAGE(Touch Screen)
A1 01	COLLECTION(Application)
85 03	REPORT_ID (3)
A1 00	COLLECTION (Physical)
09 00	USAGE (Undefined)
15 00	LOGICAL_MINIMUM (0)
25 0F	LOGICAL_MAXIMUM (15)
75 04	REPORT_SIZE (4)
95 02	REPORT_COUNT (2)
81 02	INPUT (Data, Variable, Absolute) ; Contact index & type
09 00	USAGE (Undefined)
25 03	LOGICAL_MAXIMUM (3)
75 02	REPORT_SIZE (2)
95 01	REPORT_COUNT (1)
81 02	INPUT (Data, Variable, Absolute) ; Contact state
75 06	REPORT_SIZE (6)
81 01	INPUT (Constant) ; Reserved bits
09 00	USAGE (Undefined)
26 FF 00	LOGICAL_MAXIMUM (255)
75 08	REPORT_SIZE (8)
81 02	INPUT (Data, Variable, Absolute) ; Major axis orientation
09 00	USAGE (Undefined)
25 7F	LOGICAL_MAXIMUM (127)
81 02	INPUT (Data, Variable, Absolute) ; Pressure
05 01	USAGE_PAGE (Generic Desktop)
09 30	USAGE (X)
09 31	USAGE (Y)
26 FF 0F	LOGICAL_MAXIMUM (4095)
75 0C	REPORT_SIZE (12)
95 02	REPORT_COUNT (2)
81 02	INPUT (Data, Variable, Absolute) ; X & Y
09 00	USAGE (Undefined)
26 FF 1F	LOGICAL_MAXIMUM (8191)
75 10	REPORT_SIZE (16)
95 02	REPORT_COUNT (2)
81 02	INPUT (Data, Variable, Absolute) ; Maj/min axis lengths
C0	END_COLLECTION
C0	END_COLLECTION

Figure 31 – Component for a standard contact data report

9.3.2 Report format

The contact data component generates the report illustrated in Figure 17.

9.4 Tap Gesture

9.4.1 Component specification

Figure 32 illustrates the ZID profile standard component for a tap gesture report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (51)	Tags
05 0D	USAGE_PAGE(Digitizers)
09 04	USAGE(Touch Screen)
A1 01	COLLECTION(Application)
85 04	REPORT_ID (4)
A1 00	COLLECTION (Physical)
09 00	USAGE (Undefined)
15 00	LOGICAL_MINIMUM (0)
25 07	LOGICAL_MAXIMUM (7)
75 03	REPORT_SIZE (3)
95 01	REPORT_COUNT (1)
81 02	INPUT (Data, Variable, Absolute) ; Finger count
09 00	USAGE (Undefined)
25 1F	LOGICAL_MAXIMUM (31)
75 05	REPORT_SIZE (5)
81 02	INPUT (Data, Variable, Absolute) ; Type
75 08	REPORT_SIZE (8)
81 01	INPUT (CONSTANT) ; Reserved byte
05 01	USAGE_PAGE (Generic Desktop)
09 30	USAGE (X)
09 31	USAGE (Y)
26 FF 0F	LOGICAL_MAXIMUM (4095)
75 0C	REPORT_SIZE (12)
95 02	REPORT_COUNT (2)
81 02	INPUT (Data, Variable, Absolute) ; X & Y positions
C0	END_COLLECTION
C0	END_COLLECTION

Figure 32 – Component for a standard tap gesture report

9.4.2 Report format

The tap gesture component generates the report illustrated in Figure 18.

9.5 Scroll gesture

9.5.1 Component specification

Figure 33 illustrates the ZID profile standard component for a scroll gesture report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (57)	Tags
05 0D	USAGE_PAGE(Digitizers)
09 04	USAGE(Touch Screen)
A1 01	COLLECTION(Application)
85 05	REPORT_ID (5)
A1 00	COLLECTION (Physical)
09 00	USAGE (Undefined)
15 00	LOGICAL_MINIMUM (0)
25 07	LOGICAL_MAXIMUM (7)
75 03	REPORT_SIZE (3)
95 01	REPORT_COUNT (1)
81 02	INPUT (Data, Variable, Absolute) ; Finger count
09 00	USAGE (Undefined)
25 1F	LOGICAL_MAXIMUM (31)
75 05	REPORT_SIZE (5)
81 02	INPUT (Data, Variable, Absolute) ; Gesture type
75 08	REPORT_COUNT (8)
81 01	INPUT (Constant) ; Reserved byte
09 00	USAGE (Undefined)
25 07	LOGICAL_MAXIMUM (7)
75 03	REPORT_SIZE (3)
81 02	INPUT (Data, Variable, Absolute) ; Direction
75 01	REPORT_SIZE (1)
81 01	INPUT (Constant) ; Reserved bit
09 00	USAGE (Undefined)
26 FF 0F	LOGICAL_MAXIMUM (4095)
75 0C	REPORT_SIZE (12)
81 02	INPUT (Data, Variable, Absolute) ; Scroll distance
C0	END_COLLECTION
C0	END_COLLECTION

Figure 33 – Component for a standard scroll gesture report

9.5.2 Report format

The scroll gesture component generates the report illustrated in Figure 19.

9.6 Pinch gesture

9.6.1 Component specification

Figure 34 illustrates the ZID profile standard component for a pinch gesture report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (57)	Tags
05 0D	USAGE_PAGE(Digitizers)
09 04	USAGE(Touch Screen)
A1 01	COLLECTION(Application)
85 06	REPORT_ID (6)
A1 00	COLLECTION (Physical)
09 00	USAGE (Undefined)
09 22	USAGE (Finger)
15 00	LOGICAL_MINIMUM (0)
25 01	LOGICAL_MAXIMUM (1)
75 01	REPORT_SIZE (1)
95 02	REPORT_COUNT (2)
81 02	INPUT (Data, Variable, Absolute) ; Dir. & finger
75 06	REPORT_SIZE (6)
95 01	REPORT_COUNT (1)
81 01	INPUT (Constant) ; Reserved bits
09 00	USAGE (Undefined)
26 FF 0F	LOGICAL_MAXIMUM (4095)
75 0C	REPORT_SIZE (12)
81 02	INPUT (Data, Variable, Absolute) ; Distance
75 04	REPORT_SIZE (4)
81 01	INPUT (Constant) ; Reserved bits
05 01	USAGE_PAGE (Generic Desktop)
09 30	USAGE (X)
09 31	USAGE (Y)
75 0C	REPORT_SIZE (12)
95 02	REPORT_COUNT (2)
81 02	INPUT (Data, Variable, Absolute) ; Center X & Y
C0	END_COLLECTION
C0	END_COLLECTION

Figure 34 – Component for a standard pinch gesture report

9.6.2 Report format

The pinch gesture component generates the report illustrated in Figure 20.

9.7 Rotation gesture

9.7.1 Component specification

Figure 35 illustrates the ZID profile standard component for a rotation gesture report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (41)	Tags
05 0D	USAGE_PAGE(Digitizers)
09 04	USAGE(Touch Screen)
A1 01	COLLECTION(Application)
85 07	REPORT_ID (7)
A1 00	COLLECTION (Physical)
09 00	USAGE (Undefined)
09 22	USAGE (Finger)
15 00	LOGICAL_MINIMUM (0)
25 01	LOGICAL_MAXIMUM (1)
75 01	REPORT_SIZE (1)
95 02	REPORT_COUNT (2)
81 02	INPUT (Data, Variable, Absolute) ; Dir. & finger
75 06	REPORT_SIZE (6)
95 01	REPORT_COUNT (1)
81 01	INPUT (Constant) ; Reserved bits
09 00	USAGE (Undefined)
26 FF 00	LOGICAL_MAXIMUM (255)
75 08	REPORT_SIZE (8)
81 02	INPUT (Data, Variable, Absolute) ; Magnitude
C0	END_COLLECTION
C0	END_COLLECTION

Figure 35 – Component for a standard rotation gesture report

9.7.2 Report format

The rotation gesture component generates the report illustrated in Figure 21.

9.8 Sync

9.8.1 Component specification

Figure 36 illustrates the ZID profile standard component for a rotation gesture report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (36)	Tags
05 0D	USAGE_PAGE(Digitizers)
09 04	USAGE(Touch Screen)
A1 01	COLLECTION(Application)
85 08	REPORT_ID (8)
A1 01	COLLECTION (Application)
09 00	USAGE (Undefined)
15 00	LOGICAL_MINIMUM (0)
25 0F	LOGICAL_MAXIMUM (15)
75 04	REPORT_SIZE (4)
95 01	REPORT_COUNT (1)
81 02	INPUT (Data, Variable, Absolute) ; Contact count
09 00	USAGE (Undefined)
25 01	LOGICAL_MAXIMUM (1)
75 01	REPORT_SIZE (1)
81 02	INPUT (Data, Variable, Absolute) ; Gesture present
75 03	REPORT_SIZE (3)
81 01	INPUT (Constant) ; Reserved bits
C0	END_COLLECTION
C0	END_COLECTION

Figure 36 – Component for a standard sync report

9.8.2 Report format

The sync component generates the report illustrated in Figure 16.

9.9 Touch sensor properties

9.9.1 Component specification

Figure 37 illustrates the ZID profile standard component for a touch sensor properties report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (75)	Tags
05 0D	USAGE_PAGE(Digitizers)
09 04	USAGE(Touch Screen)
A1 01	COLLECTION(Application)
85 09	REPORT_ID (9)
A1 02	COLLECTION (Logical)
09 00	USAGE (Undefined)
15 00	LOGICAL_MINIMUM (0)
25 0F	LOGICAL_MAXIMUM (15)
75 04	REPORT_SIZE (4)
95 01	REPORT_COUNT (1)
81 02	INPUT (Data, Variable, Absolute) ; # additional contacts
09 00	USAGE (Undefined)
25 03	LOGICAL_MAXIMUM (3)
75 02	REPORT_SIZE (2)
81 02	INPUT (Data, Variable, Absolute) ; Origin
09 00	USAGE (Undefined)
25 01	LOGICAL_MAXIMUM (1)
75 01	REPORT_SIZE (1)
95 02	REPORT_COUNT (2)
81 02	INPUT (Data, Variable, Absolute) ; Rel. index, gestures
09 00	USAGE (Undefined)
25 7F	LOGICAL_MAXIMUM (127)
75 08	REPORT_SIZE (8)
81 02	INPUT (Data, Variable, Absolute) ; Resolution x & y
09 00	USAGE (Undefined)
15 01	LOGICAL_MINIMUM (1)
26 FF 0F	LOGICAL_MAXIMUM (4095)
75 12	REPORT_SIZE (12)
81 02	INPUT (Data, Variable, Absolute) ; Max coordinate x & y
09 00	USAGE (Undefined)
15 00	LOGICAL_MINIMUM (0)
25 07	LOGICAL_MAXIMUM (7)
75 04	REPORT_SIZE (4)
95 01	REPORT_COUNT (1)
81 02	INPUT (Data, Variable, Absolute) ; Shape
81 01	INPUT (Constant) ; Reserved bits
C0	END_COLLECTION
C0	END_COLLECTION

Figure 37 – Component for a standard touch sensor properties report

9.9.2 Report format

The touch sensor properties component generates the report illustrated in Figure 14.

9.10 Tap support properties

9.10.1 Component specification

Figure 37 illustrates the ZID profile standard component for a tap support properties report. When indicated by a HID class device in its *aplHIDStdDescCompsList* attribute, the HID adaptor shall communicate the contents of this component, incorporated into a suitable report descriptor, on request from the HID class driver.

Bytes (36)	Tags
05 0D	USAGE_PAGE(Digitizers)
09 04	USAGE(Touch Screen)
A1 01	COLLECTION(Application)
85 0A	REPORT_ID (10)
A1 00	COLLECTION (Physical)
09 00	USAGE (Undefined)
15 00	LOGICAL_MINIMUM (0)
25 01	LOGICAL_MAXIMUM (1)
75 01	REPORT_SIZE (1)
95 04	REPORT_COUNT (4)
81 02	INPUT (Data, Variable, Absolute) ; Tap support indicators
75 04	REPORT_SIZE (4)
95 01	REPORT_COUNT (1)
81 01	INPUT (Constant) ; Reserved bits
75 08	REPORT_SIZE (8)
95 03	REPORT_COUNT (3)
81 01	INPUT (Constant) ; Reserved bytes
C0	END_COLLECTION
C0	END_COLLECTION

Figure 38 – Component for a standard tap support properties report

9.10.2 Report format

The tap support properties component generates the report illustrated in Figure 15.