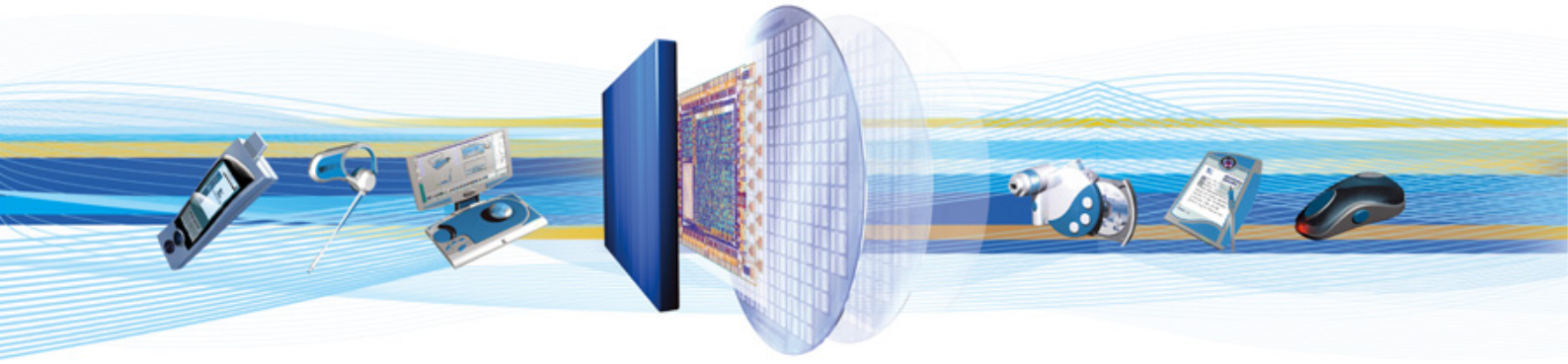
CSR Synergy Bluetooth 18.2.0

HIDH – Human Interface Device Host Profile

API Description

November 2011



Cambridge Silicon Radio Limited

Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

Tel: +44 (0)1223 692000
Fax: +44 (0)1223 692001
www.csr.com

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the Human Interface Device Host Profile (HIDH). The HIDH conforms to the host side of the Human Interface Device Profile, ref. [HIDSPEC].

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- There is one control application that manages all connected HID devices, i.e. connection establishment and release
- It is expected that the control application will store HID device information, i.e. SDP information for each device that is known to the host

It is assumed that the user has knowledge of the Human Interface Device Profile specification [HIDSPEC].

2 Description

2.1 Introduction

The Human Interface Device profile allows attaching input/output device like mice, keyboard, joystick, game pads, remote controls etc. to a host like a computer, gaming console, industrial machine, data-recording device etc. The most typical example is the desktop usage scenario where a mouse and keyboard (HID devices) are used for controlling a computer (HID host). However, usage scenarios also include collecting/transmitting non-human input/output like temperature sensors, on/off switches etc.

The HIDH supplies functionality for:

- Establishing connections with HID devices (connections with multiple devices)
- Re-establishing connection if connection is dropped
- Unplugging HID devices
- Transport of HID data

The HIDH does not provide functionality for security handling, this should be handled from the application through the Security Controller.

2.2 Reference Model

The HIDH interfaces to the Connection Manager (CM) and to the Service Discovery Server (SDS) through the CM. The application must interface to the HIDH profile and to the Security Controller (SC) in order to handle encryption initiation.

The application may be split into a control and one or more user applications, where the control application handles connection control and the user application is the application that processes the HID data. There may be one or more user application per connected HID devices.

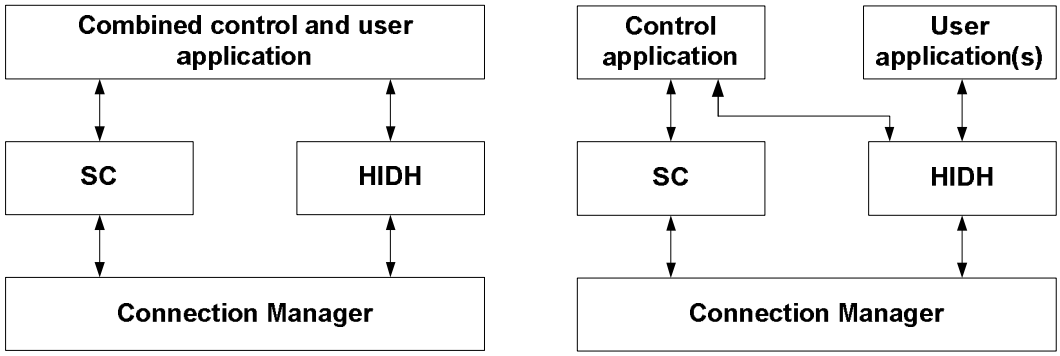


Figure 1: Reference models

In order for the control application to have as low coupling as possible to the user application(s), the HIDH contains a *device slot table*. The table key is a *slotId*, and the contents are *user application handles*. User applications register themselves in this table, and the control application connects devices to slots. This means that the user and control application only has to share the static slotIds, and not e.g. taskIds, which can be dynamic.

2.3 Sequence Overview

When the HIDH starts up it initialises and registers before entering the active state where it can provide service to the application. The very first connection between the host and a device must always be established by the host. As soon as the first connection is established, a relation between the host and device is created and the host application must store the device information. The HID device information is given by the service record of the device and determines the responsibility for future connection establishments/re-establishments as well as

describes the type of device with its input and output capabilities. HIDH can handle multiple HID devices simultaneously; however, connecting devices must be done sequentially.

When a device is connected, HID data can be exchanged between the host and device. If a HID connection is lost, attempts will be made to re-establish the connection in a 30 second period. If the connection is not re-established within this period, the connection is considered permanently lost and it is up to the application to decide on the proper action. A relation between host and a device can be removed by the unplug procedure which will ensure that both the host and device disconnect without attempting to re-establish the connection.

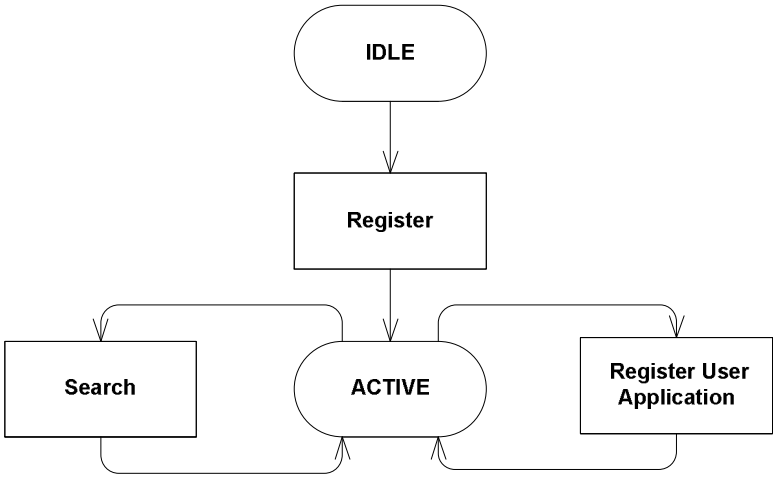


Figure 2: HIDH main state diagram

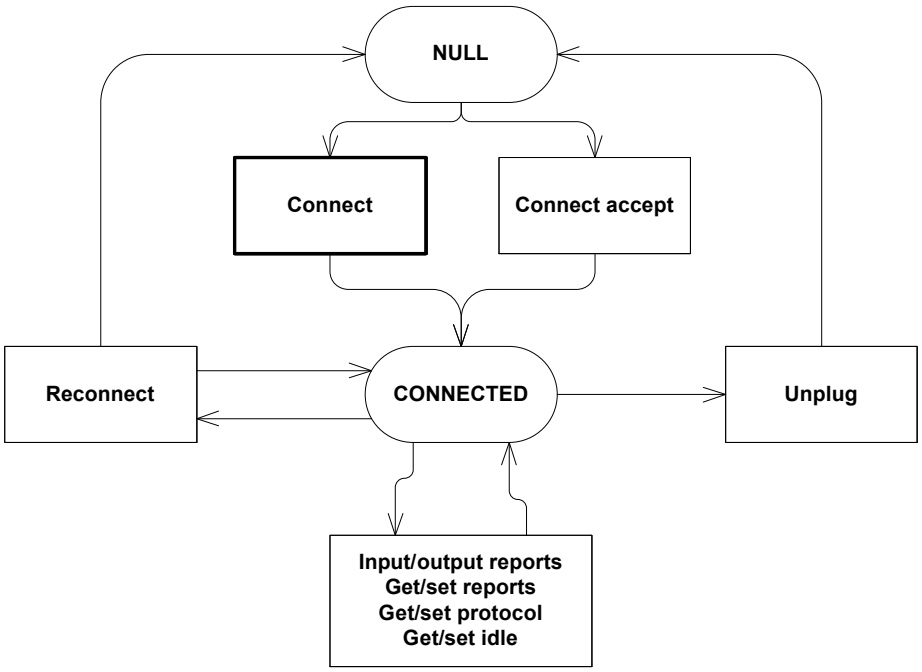


Figure 3: HIDH sub-state (per connected device) diagram, main state ACTIVE
(Note: Not all states shown)

3 Interface Description

In this section a series of MSCs will be presented to explain the usage of the primitives of HIDH. The primitives presented in this section will be described further in section 4, giving details of the parameters in each primitive.

3.1 Registering user Applications

To help the decoupled design approach with separated control and user applications, each user application must register itself at the HIDH before it can gain access to device data. When an user application register itself, it sends a CSR_BT_HIDH_REGISTER_USER_REQ primitive which contains the user application handle and the deviceId.

The deviceId is a number by which the HIDH makes distinction between the connected devices. The HIDH supports up to 7 simultaneous connections, whereas the deviceId number can be from 0 to 6.

The assignment of actual HID devices to deviceIds are carried out in the connection phase, see section 3.3.

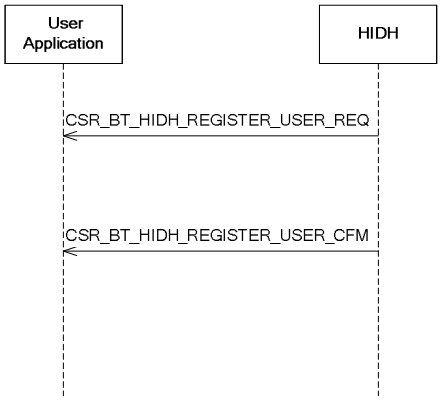


Figure 4: Registering user Applications

3.2 Host Initiated Connection Establishment

A connection to a HID device can be initiated by the host by sending a CSR_BT_HIDH_CONNECT_REQ primitive. HIDH will perform service discovery if the application does not supply the needed information. When the HIDH has successfully established the connection it will confirm it by sending a CSR_BT_HIDH_CONNECT_CFM to the control application and a CSR_BT_HIDH_STATUS_IND to the user application. Both of these primitives contain the device information (SdpInfo) that should be stored for later connections that might be established from the device.

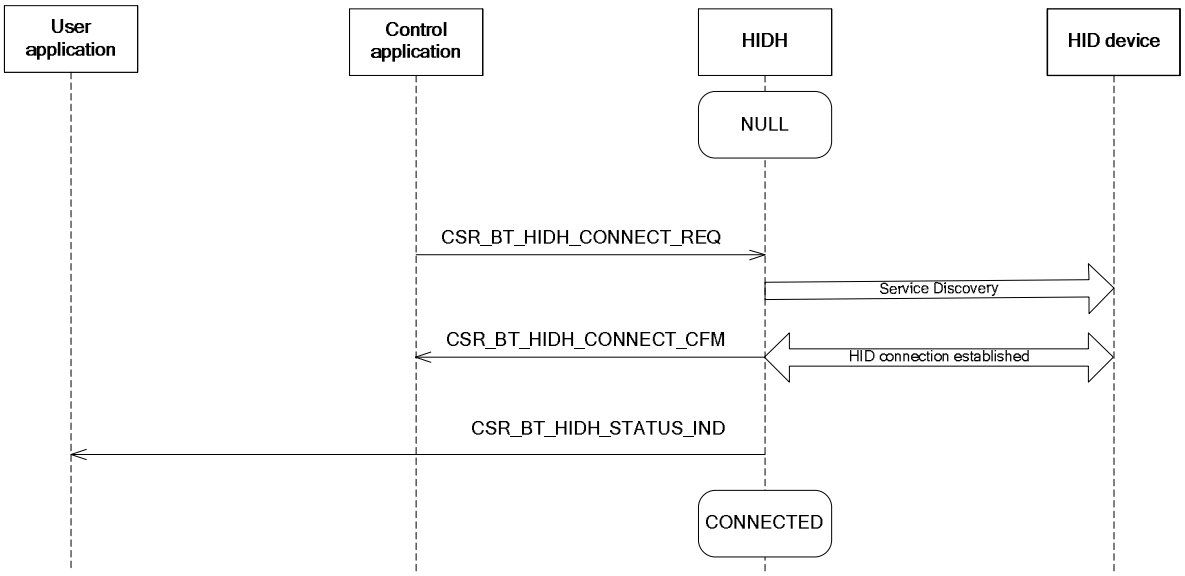


Figure 5: Host initiated connection establishment

3.3 Accepting Connections Established from Device

When a host application wants to allow a known device to establish a connection it must send a CSR_BT_HIDH_CONNECT_ACCEPT_REQ primitive to HIDH specifying the device information. HIDH will confirm the request by sending a CSR_BT_HIDH_CONNECT_ACCEPT_REQ, stating the result of the request. The confirm with successful result does not imply that a connection has been established but only that the request was accepted. When a connection is established, HIDH will inform the control application by sending a CSR_BT_HIDH_CONNECT_ACCEPT_IND primitive and the user application by sending a CSR_BT_HIDH_STATUS_IND.

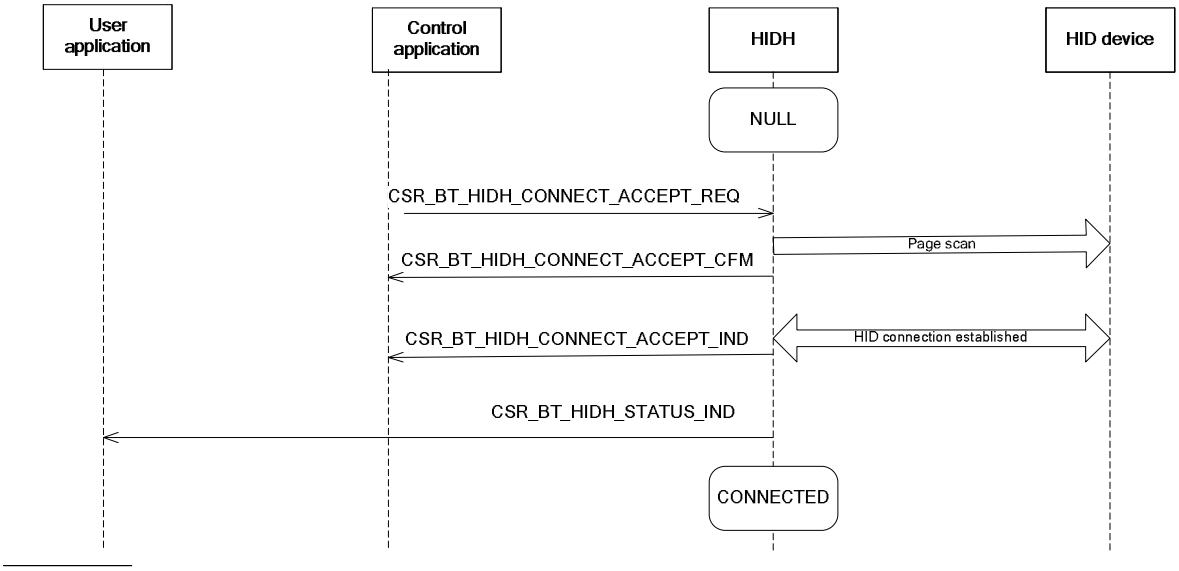


Figure 6: HID device initiated connection establishment

3.4 Connection Re-establishment

If a connection to a HID device is terminated unexpectedly, HIDH will inform the control application by sending a CSR_BT_HIDH_STATUS_IND and take the necessary action to re-establish the connection. If the connection is re-established, another CSR_BT_HIDH_STATUS_IND is sent to the control application. If the connection has not

been re-established within 30 seconds the control application will receive a CSR_BT_HIDH_DISCONNECT_IND primitive and the user application will receive a CSR_BT_HIDH_STATUS_IND indicating that the connection is lost.

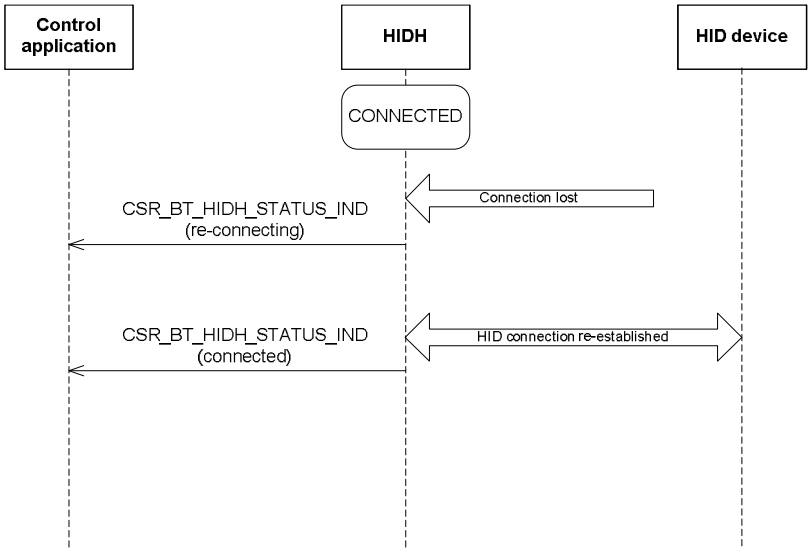


Figure 7: Connection re-establishment

If the connection has not been re-established within 30 seconds the control application will instead receive a CSR_BT_HIDH_DISCONNECT_IND primitive and the user application will receive a CSR_BT_HIDH_STATUS_IND indicating that the connection is lost. The status indication is always sent before the actual disconnect indication.

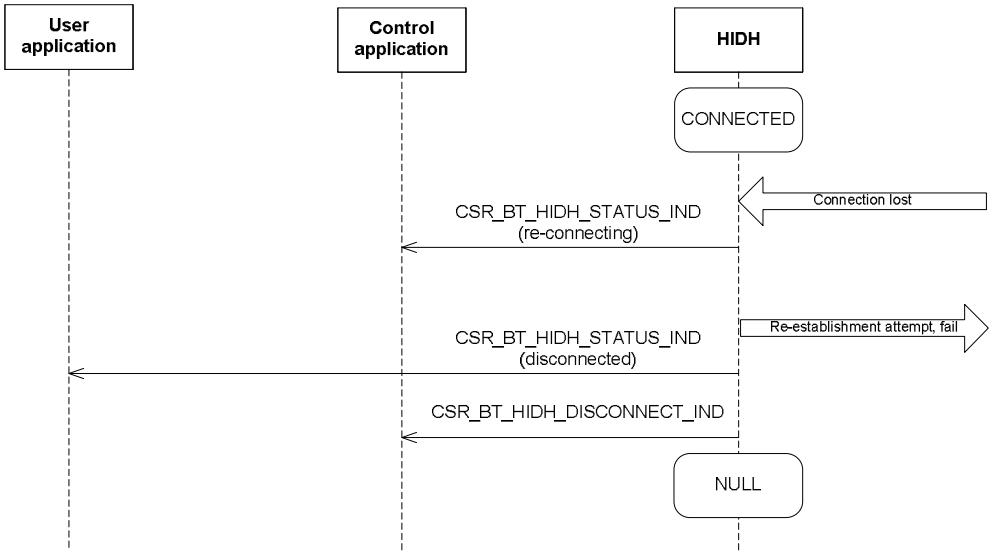


Figure 8: Failing connection re-establishment

3.5 Unplug/Disconnect

The unplug procedure allows the application to remove the relation between host and HID device. The application on host triggers the procedure by sending a CSR_BT_HIDH_CONTROL_REQ primitive with an operation parameter set to ‘unplug’. When the connection is subsequently released by the device, this is reported to the control application with a CSR_BT_HIDH_DISCONNECT_IND primitive and to the user application with a CSR_BT_HIDH_STATUS_IND primitive. No connection re-establishment attempts will be made by HIDH.

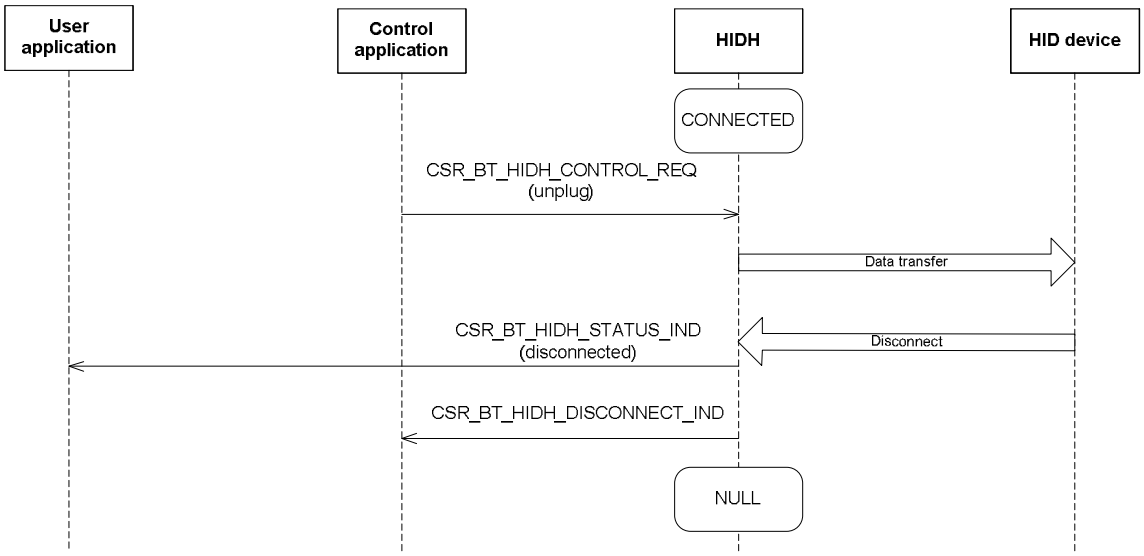


Figure 9: Host initiated unplug

The HID device may also initiate unplugging. This is reported to the control application with a CSR_BT_HIDH_CONTROL_IND primitive with an operation parameter set to 'unplug'. The control application should upon receiving this primitive erase the device information and disconnect the device by sending a CSR_BT_HIDH_DISCONNECT_REQ. HIDH will confirm the disconnect with a CSR_BT_HIDH_DISCONNECT_CFM to the control application and inform the user application with a CSR_BT_HIDH_STATUS_IND. In both cases, the status indication is always sent before the actual disconnect indication.

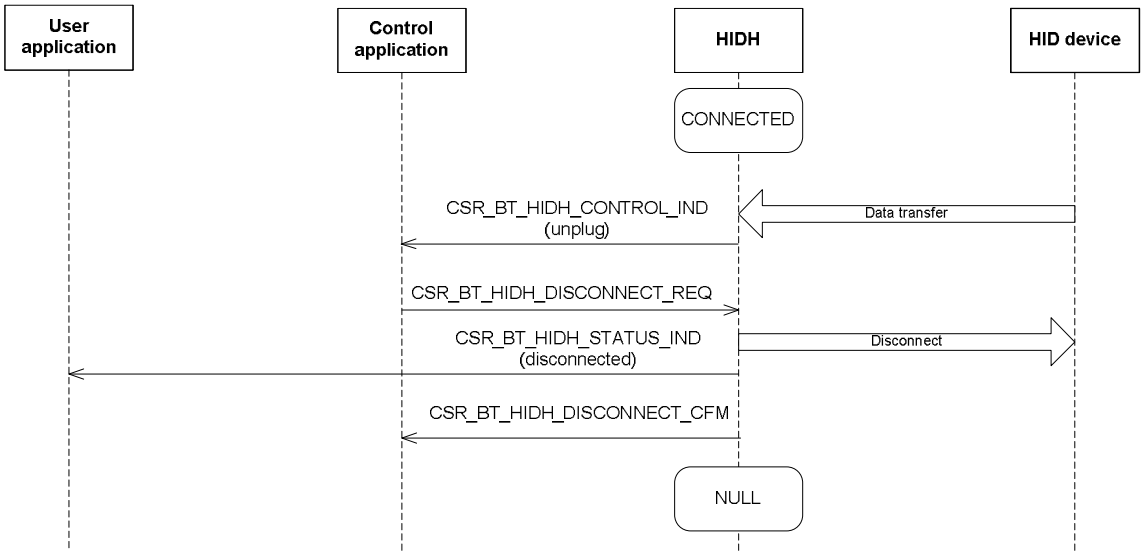


Figure 10: HID device initiated unplug

3.6 Device Control

The unplug procedure described in the previous section is a special case of device control, sending control messages between host and device. The general device control allows the host to send control messages to the device; various control operations can be requested for resetting, suspending and un-suspending the device. The host application can send a control message by sending a CSR_BT_HIDH_CONTROL_REQ primitive to HIDH.

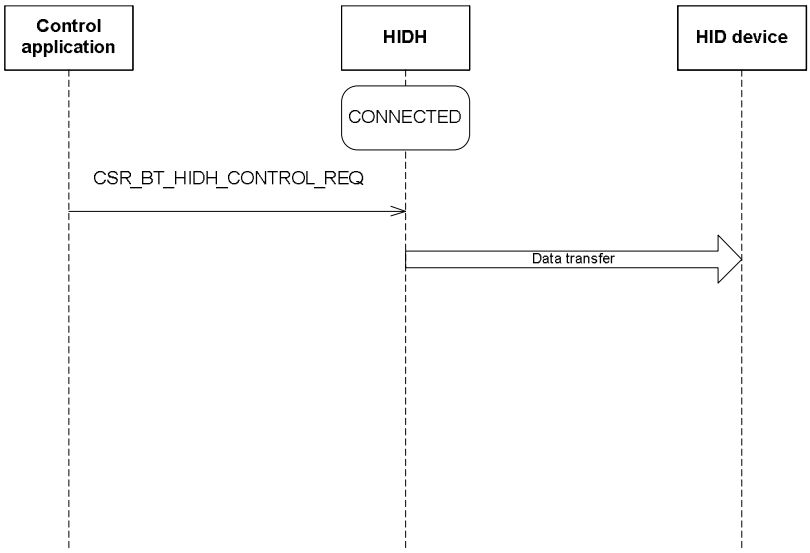


Figure 11: Control message transfer

Note that with the exception of the 'unplug' control message, it is only allowed to send control messages in the direction from host to device.

3.7 Get/Set Report

The user application can retrieve a report from the device by sending a CSR_BT_HIDH_GET_REPORT_REQ primitive. The report will be returned in a CSR_BT_HIDH_DATA_IND primitive.

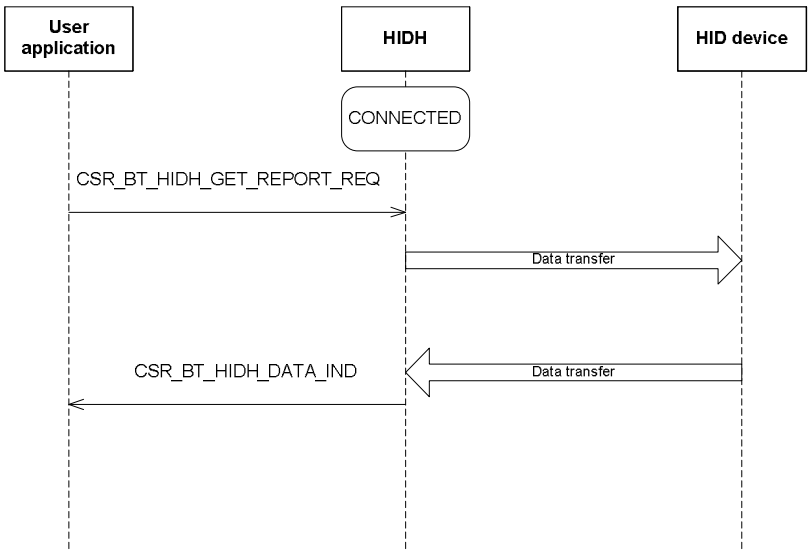


Figure 12: Get report

To send a report to the device, the application sends a CSR_BT_HIDH_SET_REPORT_REQ primitive. HIDH will transfer the report as data and a response from the device is indicated with a CSR_BT_HIDH_HANDSHAKE_IND primitive to the user application.

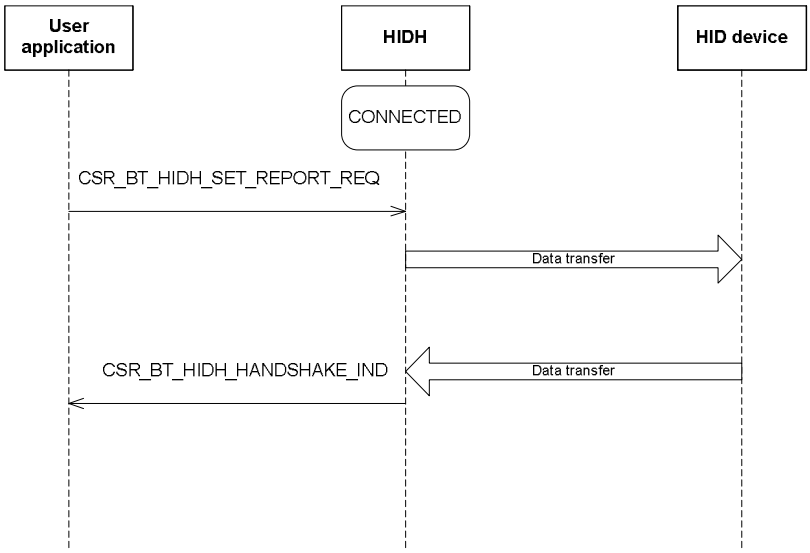


Figure 13: Set report

3.8 Get/Set Protocol

The user application can retrieve the type of HID protocol from the device by sending a CSR_BT_HIDH_GET_PROTOCOL_REQ primitive. If the request is supported by the device, the response is returned to the application in a CSR_BT_HIDH_DATA_IND primitive. If not supported by the device a CSR_BT_HIDH_HANDSHAKE_IND is returned.

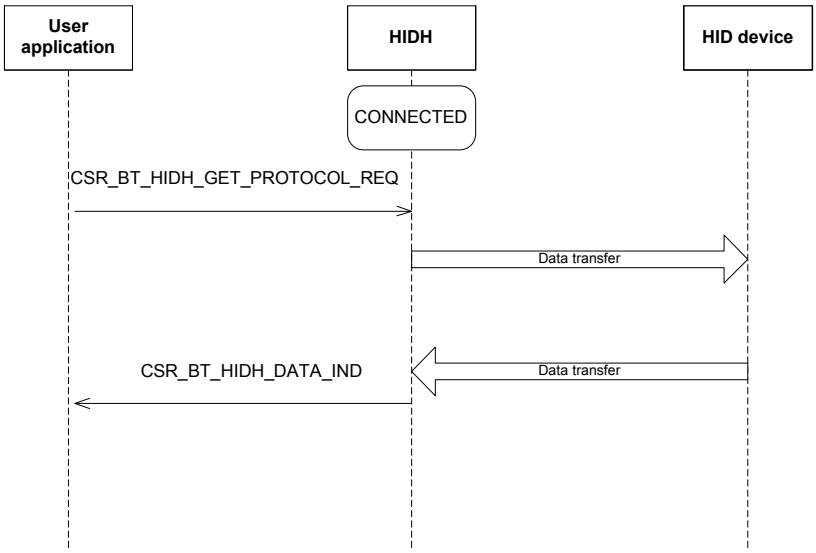


Figure 14: Get protocol

To specify the protocol to the device, the application sends a CSR_BT_HIDH_SET_PROTOCOL_REQ primitive. A response from the device is indicated with a CSR_BT_HIDH_HANDSHAKE_IND primitive to the user application.

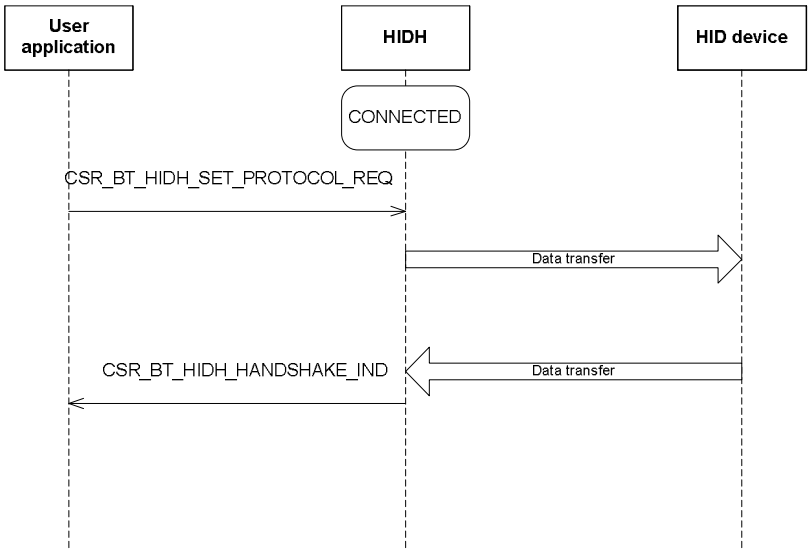


Figure 15: Set protocol

3.9 Get/Set Idle

The user application can retrieve the idle setting of the device by sending a CSR_BT_HIDH_GET_IDLE_REQ primitive. If the request is supported by the device, the response is returned to the application in a CSR_BT_HIDH_DATA_IND primitive. If not supported by the device a CSR_BT_HIDH_HANDSHAKE_IND is returned.

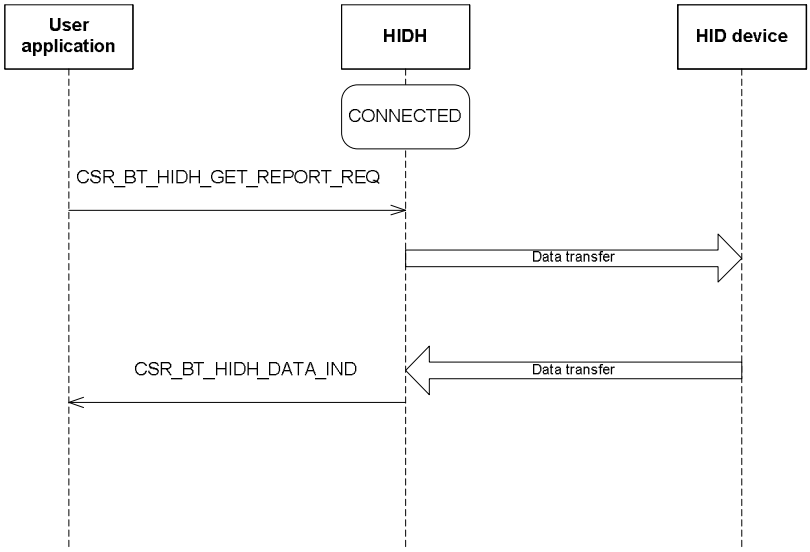


Figure 16: Get idle

To send a report to the device, the application sends a CSR_BT_HIDH_SET_IDLE_REQ primitive. A response from the device is indicated with a CSR_BT_HIDH_HANDSHAKE_IND primitive to the user application.

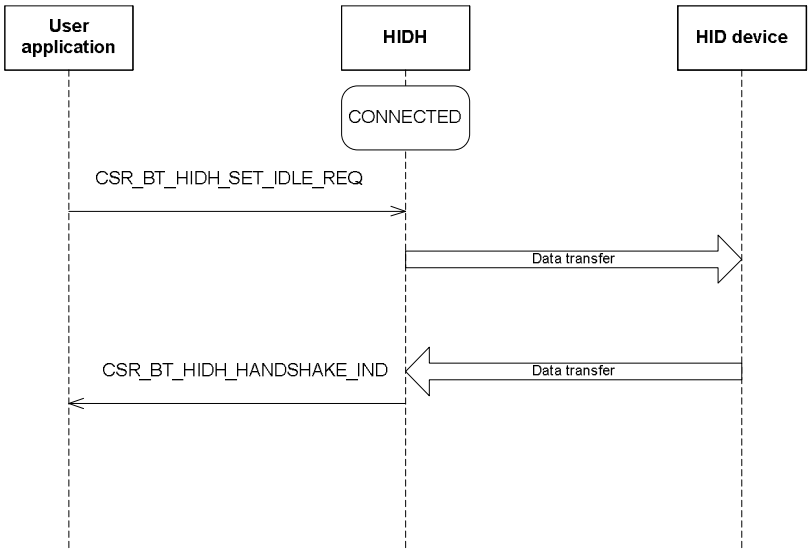


Figure 17: Set idle

3.10 Data Transfer

The user application can send data to the device using the CSR_BT_HIDH_DATA_REQ primitive. Data arriving from the device is transferred to the user application in a CSR_BT_HIDH_DATA_IND primitive.

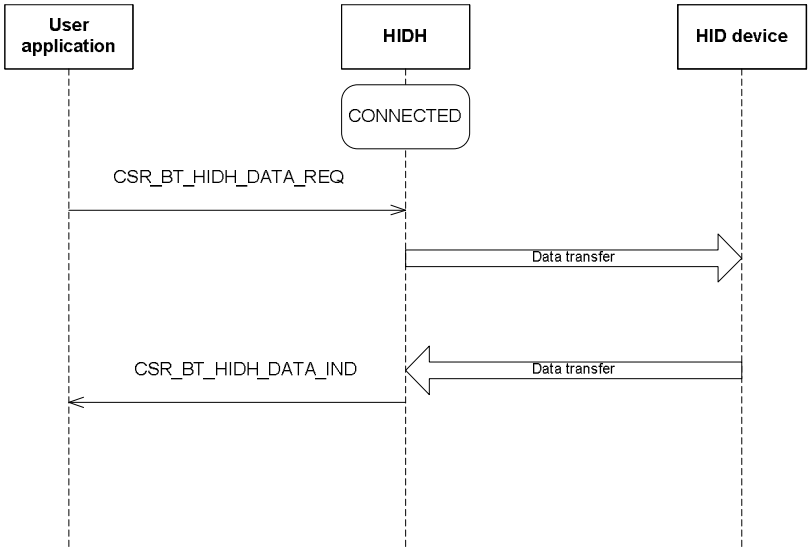


Figure 18: Data transport

4 Human Interface Device Profile Host Primitives

This section gives an overview of the primitives and parameters in the interface. Detailed information can be found in the corresponding csr_bt_hidh_prim.h file.

4.1 List of All Primitives

Primitives:	Reference:
CSR_BT_HIDH_REGISTER_USER_REQ	Section 4.2
CSR_BT_HIDH_REGISTER_USER_CFM	Section 4.2
CSR_BT_HIDH_CONNECT_CFM	Section 4.3
CSR_BT_HIDH_CONNECT_ACCEPT_REQ	Section 4.4
CSR_BT_HIDH_CONNECT_ACCEPT_CFM	Section 4.4
CSR_BT_HIDH_CONNECT_ACCEPT_IND	Section 4.4
CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT_REQ	Section 4.5
CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT_CFM	Section 4.5
CSR_BT_HIDH_DISCONNECT_REQ	Section 4.6
CSR_BT_HIDH_DISCONNECT_CFM	Section 4.6
CSR_BT_HIDH_DISCONNECT_IND	Section 4.6
CSR_BT_HIDH_STATUS_IND	Section 4.7
CSR_BT_HIDH_HANDSHAKE_IND	Section 4.8
CSR_BT_HIDH_CONTROL_REQ	Section 4.9
CSR_BT_HIDH_CONTROL_IND	Section 4.9
CSR_BT_HIDH_GET_REPORT_REQ	Section 4.10
CSR_BT_HIDH_SET_REPORT_REQ	Section 4.11
CSR_BT_HIDH_GET_PROTOCOL_REQ	Section 4.12
CSR_BT_HIDH_SET_PROTOCOL_REQ	Section 4.13
CSR_BT_HIDH_GET_IDLE_REQ	Section 4.14
CSR_BT_HIDH_SET_IDLE_REQ	Section 4.15
CSR_BT_HIDH_DATA_REQ	Section 4.16
CSR_BT_HIDH_DATA_IND	Section 4.16
CSR_BT_HIDH_SECURITY_IN_REQ	Section 4.17
CSR_BT_HIDH_SECURITY_IN_CFM	Section 4.17
CSR_BT_HIDH_SECURITY_OUT_REQ	Section 4.17
CSR_BT_HIDH_SECURITY_OUT_CFM	Section 4.17

Table 1: List of all primitives

4.2 CSR_BT_HIDH_REGISTER_USER

Parameters / Primitives	type	slotId	userHandle	resultCode	resultSupplier
CSR_BT_HIDH_REGISTER_USER_REQ	✓	✓	✓		
CSR_BT_HIDH_REGISTER_USER_CFM	✓	✓		✓	✓

Table 2: CSR_BT_HIDH_REGISTER_USER Primitives

Description

The CSR_BT_HIDH_REGISTER_USER primitives are used for registering user application handles in the HIDH. This must be done before attempting any connection. The slotId is then later used in connection signals to route HID data to user applications.

Parameters

type	Signal identity CSR_BT_HIDH_REGISTER_USER_REQ/CFM.
slotId	Register the user application for this slot.
userHandle	Application handle to receive device data
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.3 CSR_BT_HIDH_CONNECT

Primitives \ Parameters	type	resultCode	resultSupplier	ctrlHandle	slotId	deviceAddr	deviceId	flushTimeout	*qosCtrl	qosCtrlCount	*qosIntr	qosIntrCount	*sdpInfo	sdpInfoCount	*serviceName	descriptorLength	*descriptor	btConnId
CSR_BT_HIDH_CONNECT_REQ	✓			✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
CSR_BT_HIDH_CONNECT_CFM	✓	✓	✓			✓	✓						✓	✓	✓	✓	✓	✓

Table 3: CSR_BT_HIDH_CONNECT Primitives

Description

The CSR_BT_HIDH_CONNECT_REQ primitive is used for establishing a HID connection with a device. The outcome of the request is reported back by a CSR_BT_HIDH_CONNECT_CFM primitive. There must never be more than one outstanding request at any time.

Parameters

type	Signal identity CSR_BT_HIDH_CONNECT_REQ/CFM.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
ctrlHandle	Identity of the control application process (the initiator of the connection request).
slotId	Slot id from which the user application handle is used.
deviceAddr	Bluetooth address of the remote device.
deviceId	Identity assigned to distinguish devices that are connected or being connected. The value is only valid if the result field indicates success.
flushTimeout	Wanted L2CAP flush timeout value. Recommended value: 0xFFFF (infinite).
*qosCtrl	Wanted quality of service for the HID control channel signaling. If NULL, default values will be used.
qosCtrlCount	Number of elements pointed to by *qosCtrl*. This must be either 0 (when *qosCtrl* is NULL) or 1
*qosIntr	Wanted quality of service for the HID interrupt channel signaling. If NULL, default values will be used.
qosIntrCount	Number of elements pointed to by *qosIntr*. This must be either 0 (when *qosIntr* is NULL) or 1
*sdpInfo	Device information retrieved from the HID devices SDP record. Only included in a confirm if the result is success and only included in a request if the device is known.
sdpInfoCount	Number of elements pointed to by *sdpInfo*. The value is either 0 when *sdpInfo* is

	NULL, otherwise 1.
*serviceName	The null-terminated ASCII-string for the HID device service name. Can be NULL.
descriptorLength	Number of bytes in the *descriptor* pointer.
*descriptor	The raw HID descriptor table.
btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend`-function.

4.4 CSR_BT_HIDH_CONNECT_ACCEPT

Primitives \ Parameters	type	resultCode	resultSupplier	ctrlHandle	slotId	deviceAddr	deviceId	flushTimeout	*qosCtrl	qosCtrlCount	*qosIntr	qosIntrCount	*sdpInfo	sdpInfoCount	*serviceName	descriptorLength	*descriptor
CSR_BT_HIDH_CONNECT_ACCEPT_REQ	✓			✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CSR_BT_HIDH_CONNECT_ACCEPT_CFM	✓	✓	✓			✓	✓										
CSR_BT_HIDH_CONNECT_ACCEPT_IND	✓	✓	✓				✓										

Table 4: CSR_BT_HIDH_CONNECT_ACCEPT Primitives

Description

The CSR_BT_HIDH_CONNECT_ACCEPT_REQ primitive is used for allowing a specific HID device to establish a HID connection with the host. HIDH will immediately confirm with a CSR_BT_HIDH_CONNECT_ACCEPT_CFM primitive to indicate if the request is accepted or not. There must never be more than one outstanding request at any time. When the connection is established, a CSR_BT_HIDH_CONNECT_ACCEPT_IND is sent to the control application.

Parameters

type	Signal identity CSR_BT_HIDH_CONNECT_ACCEPT_REQ/CFM/IND
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
ctrlHandle	Identity of the control application process (the initiator of the connection request).
slotId	Slot id from which the user application handle is used.
deviceAddr	Bluetooth address of the remote device
deviceId	Identity assigned to distinguish devices that are connected or being connected. The value is only valid if the result field indicates success.
flushTimeout	Wanted L2CAP flush timeout value. Recommended value: 0xFFFF (infinite).
*qosCtrl	Wanted quality of service for the HID control channel signaling. If NULL, default values will be used.
qosCtrlCount	Number of elements pointed to by *qosCtrl*. This must be either 0 (when *qosCtrl* is NULL) or 1
*qosIntr	Wanted quality of service for the HID interrupt channel signaling. If NULL, default values will be used.
qosIntrCount	Number of elements pointed to by *qosIntr*. This must be either 0 (when *qosIntr* is

	NULL) or 1
*sdpInfo	Device information retrieved from the HID devices SDP record. Only included in a confirm if the result is success and only included in a request if the device is known.
sdpInfoCount	Number of elements pointed to by *sdpInfo*. The value is either 0 when *sdpInfo* is NULL, otherwise 1.
*serviceName	The null-terminated ASCII-string for the HID device service name. Can be NULL.
descriptorLength	Number of bytes in the *descriptor* pointer.
*descriptor	The raw HID descriptor table.

4.5 CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT

Primitives \ Parameters	type	resultCode	resultSupplier	deviceId
CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT_REQ	✓			✓
CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT_CFM	✓	✓	✓	✓

Table 5: CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT Primitives

Description

The CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT_REQ primitive is used for cancelling, waiting for a device to connect. The result of the request is reported to the application in a CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT_CFM.

Parameters

type	Signal identity CSR_BT_HIDH_CANCEL_CONNECT_ACCEPT_REQ/CFM.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
deviceId	Identity assigned to distinguish devices that are connected or being connected. When successfully cancelled, the deviceId is not valid afterwards.

4.6 CSR_BT_HIDH_DISCONNECT

Parameters / Primitives	type	resultCode	resultSupplier	deviceId
CSR_BT_HIDH_DISCONNECT_REQ	✓			✓
CSR_BT_HIDH_DISCONNECT_CFM	✓	✓	✓	✓
CSR_BT_HIDH_DISCONNECT_IND	✓	✓	✓	✓

Table 6: CSR_BT_HIDH_DISCONNECT Primitives

Description

The CSR_BT_HIDH_DISCONNECT_REQ primitive is used for disconnecting a device. The result of the request is reported to the application in a CSR_BT_HIDH_DISCONNECT_CFM. If the connection is dropped or disconnected by the HID device this is indicated to the control application with a CSR_BT_HIDH_DISCONNECT_IND primitive.

Parameters

type	Signal identity CSR_BT_HIDH_DISCONNECT_REQ/CFM/IND.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
deviceId	Identity assigned to distinguish connected devices. When successfully disconnected, the deviceId is not valid afterwards.

4.7 CSR_BT_HIDH_STATUS

Primitives	type	deviceId	status	*sdpInfo	*sdpInfo	sdpInfoCount	*serviceName	descriptorLength	*descriptor
CSR_BT_HIDH_STATUS_IND	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 7: CSR_BT_HIDH_STATUS Primitives

Description

The CSR_BT_HIDH_STATUS_IND primitive is used for informing the application(s) of a change in the connection status. The control application will receive this message when the connection is dropped and re-establishment is started. If the connection is re-established another primitive is sent to the control application. The user application will receive a CSR_BT_HIDH_STATUS_IND primitive when a connection is established and again when disconnected, i.e. the user application is not notified of re-establishment.

Parameters

type	Signal identity CSR_BT_HIDH_STATUS_IND.
deviceId	Identity assigned to distinguish devices.
status	The connection status. 0: Disconnected, 1: Connected, 2: Re-connecting
*sdpInfo	Device information retrieved from the HID devices SDP record. Only included in a confirm if the result is success and only included in a request if the device is known.
sdpInfoCount	Number of elements pointed to by *sdpInfo*. The value is either 0 when *sdpInfo* is NULL, otherwise 1.
*serviceName	The null-terminated ASCII-string for the HID device service name. Can be NULL.
descriptorLength	Number of bytes in the *descriptor* pointer.
*descriptor	The raw HID descriptor table.

4.8 CSR_BT_HIDH_HANDSHAKE

Parameters / Primitives	type	deviceId	resultCode
CSR_BT_HIDH_HANDSHAKE_IND	✓	✓	✓

Table 8: CSR_BT_HIDH_HANDSHAKE Primitives

Description

A HID handshake message from the device is indicated to the user application with a CSR_BT_HIDH_HANDSHAKE_IND primitive.

Parameters

type	Signal identity CSR_BT_HIDH_HANDSHAKE_IND.
deviceId	Identity assigned to distinguish devices.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.

4.9 CSR_BT_HIDH_CONTROL

Primitives \ Parameters	type	deviceId	operation
CSR_BT_HIDH_CONTROL_REQ	✓	✓	✓
CSR_BT_HIDH_CONTROL_IND	✓	✓	✓

Table 9: CSR_BT_HIDH_CONTROL Primitives

Description

The CSR_BT_HIDH_CONTROL_REQ primitive is used for sending a control message to the device. A control message from the device is indicated to the user application with a CSR_BT_HIDH_CONTROL_IND primitive.

Parameters

type	Signal identity CSR_BT_HIDH_CONTROL_REQ/IND.
deviceId	Identity assigned to distinguish devices.
operation	The control operation requested. The allowed values are defined in [HIDSPEC] on page 60. The file csr_bt_hidh_prim.h provides pre-defined macros for all control operation values.

4.10 CSR_BT_HIDH_GET_REPORT

Primitives \ Parameters	type	deviceId	reportType	reportId	bufferSize
CSR_BT_HIDH_GET_REPORT_REQ	✓	✓	✓	✓	✓

Table 10: CSR_BT_HIDH_GET_REPORT Primitives

Description

The CSR_BT_HIDH_GET_REPORT_REQ primitive is used for sending a ‘get report’ message to the device.

Parameters

type	Signal identity CSR_BT_HIDH_GET_REPORT_REQ.
deviceId	Identity assigned to distinguish devices.
reportType	The type of report requested. The possible values for the report type can be found in [HIDSPEC] on page 62. CSR Synergy Bluetooth provides pre-defined macros for these in the file csr_bt_hidh_prim.h.
reportId	Identity of requested report, Set to zero (0), if report ids are not used.
bufferSize	Maximum number of bytes to be returned. Set to zero (0) to return complete report.

4.11 CSR_BT_HIDH_SET_REPORT

Primitives \ Parameters	type	deviceId	reportType	reportLen	*report
CSR_BT_HIDH_SET_REPORT_REQ	✓	✓	✓	✓	✓

Table 11: HIDH_SET_REPORT Primitives

Description

The CSR_BT_HIDH_SET_REPORT_REQ primitive is used for sending a ‘set report’ message to the device.

Parameters

type	Signal identity CSR_BT_HIDH_SET_REPORT_REQ.
deviceId	Identity assigned to distinguish devices.
reportType	The type of report. The possible values for the report type can be found in [HIDSPEC] on page 64. CSR Synergy Bluetooth provides pre-defined macros for these in the file csr_bt_hidh_prim.h.
reportLen	Length of the report, including a one byte header.
*report	The report data. The first byte of the report data is reserved for header and should be left unused.

4.12 CSR_BT_HIDH_GET_PROTOCOL

Primitives \ Parameters	type	deviceId
CSR_BT_HIDH_GET_PROTOCOL_REQ	✓	✓

Table 12: CSR_BT_HIDH_GET_PROTOCOL Primitives

Description

The CSR_BT_HIDH_GET_PROTOCOL_REQ primitive is used for sending a ‘get protocol’ message to the device..

Parameters

type	Signal identity CSR_BT_HIDH_GET_PROTOCOL_REQ.
deviceId	Identity assigned to distinguish devices.

4.13 CSR_BT_HIDH_SET_PROTOCOL

Parameters / Primitives	type	deviceId	protocol
CSR_BT_HIDH_SET_PROTOCOL_REQ	✓	✓	✓

Table 13: CSR_BT_HIDH_SET_PROTOCOL Primitives

Description

The CSR_BT_HIDH_SET_PROTOCOL_REQ primitive is used for sending a ‘set protocol’ message to the device.

Parameters

type	Signal identity CSR_BT_HIDH_SET_PROTOCOL_REQ.
deviceId	Identity assigned to distinguish devices.
protocol	Protocol to be set. The possible values are defined in [HIDSPEC] on page 66, and CSR Synergy Bluetooth provides pre-defined macros in the file csr_bt_hidh_prim.h.

4.14 CSR_BT_HIDH_GET_IDLE

Primitives \ Parameters	type	deviceId
CSR_BT_HIDH_GET_IDLE_REQ	✓	✓

Table 14: CSR_BT_HIDH_GET_IDLE Primitives

Description

The CSR_BT_HIDH_GET_IDLE_REQ primitive is used for sending a ‘get idle’ message to the device.

Parameters

type	Signal identity CSR_BT_HIDH_GET_IDLE_REQ.
deviceId	Identity assigned to distinguish devices.

4.15 CSR_BT_HIDH_SET_IDLE

Parameters / Primitives	type	deviceId	idleRate
CSR_BT_HIDH_SET_IDLE_REQ	✓	✓	✓

Table 15: CSR_BT_HIDH_SET_IDLE Primitives

Description

The CSR_BT_HIDH_SET_IDLE_REQ primitive is used for sending a ‘set idle’ message to the device.

Parameters

type	Signal identity CSR_BT_HIDH_SET_IDLE_REQ.
deviceId	Identity assigned to distinguish devices.
idleRate	Wanted idle rate. 0: infinite, 1 – 255: [4ms – 1.020s]

4.16 CSR_BT_HIDH_DATA

Primitives \ Parameters	type	deviceId	reportType	dataLen	*data
CSR_BT_HIDH_DATA_REQ	✓	✓	✓	✓	✓
CSR_BT_HIDH_DATA_IND	✓	✓	✓	✓	✓

Table 16: CSR_BT_HIDH_DATA Primitives

Description

The CSR_BT_HIDH_DATA_REQ primitive is used for sending a data message to the device. A data message from the device is indicated to the user application with a CSR_BT_HIDH_DATA_IND primitive.

Parameters

type	Signal identity CSR_BT_HIDH_DATA_REQ/IND.
deviceId	Identity assigned to distinguish devices.
reportType	The type of report carried in the data. The possible values are defined in [HIDSPEC] on page 70. CSR Synergy Bluetooth provides pre-defined macros in the file csr_bt_hidh_prim.h.
dataLen	Length of data, including a one byte header.
*data	Data. The first byte of the data is reserved for header and should be left unused.

4.17 CSR_BT_HIDH_SECURITY_IN / _OUT

Parameters / Primitives	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_HIDH_SECURITY_IN_REQ	✓	✓	✓		
CSR_BT_HIDH_SECURITY_IN_CFM	✓			✓	✓
CSR_BT_HIDH_SECURITY_OUT_REQ	✓	✓	✓		
CSR_BT_HIDH_SECURITY_OUT_CFM	✓			✓	✓

Table 17: CSR_BT_HIDH_SECURITY_IN and CSR_BT_HIDH_SECURITY_OUT Primitives

Description

Applications that wish to change the enforcement to a specific profile security level, i.e. authentication, encryption and/or authorisation, can use this API to set up the security level for *new* connections. Note that this API is for the local device only and can be used from within any state.

The *CSR_BT_SECURITY_IN_REQ* signal sets up the security level for new incoming connections. Already established or pending connections are not altered.

The *CSR_BT_SECURITY_OUT_REQ* signal sets up the security level for new outgoing connections. Already established and pending connections are not altered. Note that *authorisation* should not be used for outgoing connections as that may be confusing for the user – there is really no point in requesting an outgoing connection and afterwards having to authorise as they are both locally-only decided procedures.

Note, that any attempts to set security to a less secure level than the mandatory security level will be rejected. See csr_bt_profiles.h for mandatory security settings. The default settings used by CSR Synergy Bluetooth are set to require authentication and encryption.

Note that if MITM protection is requested and the remote device does not have the required IO capabilities, pairing/bonding will fail and connections to the remote device *cannot* be made. See [SC] for further details.

Parameters

type — Signal identity CSR_BT_HIDH_SECURITY_IN/OUT_REQ/CFM

appHandle — Application handle to which the confirm message is sent.

secLevel — The application must specify one of the following values:

- CSR_BT_SEC_DEFAULT : Use default security settings
- CSR_BT_SEC_MANDATORY : Use mandatory security settings
- CSR_BT_SEC_SPECIFY : Specify new security settings

If CSR_BT_SEC_SPECIFY is set the following values can be OR'ed additionally:

- CSR_BT_SEC_AUTHORISATION: Require authorisation
- CSR_BT_SEC_AUTHENTICATION: Require authentication
- CSR_BT_SEC_ SEC_ENCRYPTION: Require encryption (implies

authentication)

- CSR_BT_SEC_MITM: Require MITM protection (implies encryption)

resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

5 Document References

Document	Reference
Human Interface Device (HID) Profile Version: 1.0 Date: 22-05-2003	[HIDSPEC]
CSR Synergy Bluetooth, SC – Security Controller API Description	[SC]

Terms and Definitions

BlueCore™	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
HID	Human Interface Device
HIDH	HID Host profile
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0

TradeMarks, Patents and Licences

Unless otherwise stated, words and logos marked with ™ or ® are trademarks registered or owned by CSR plc or its affiliates. Bluetooth® and the Bluetooth logos are trademarks owned by Bluetooth SIG, Inc. and licensed to CSR. Other products, services and names used in this document may have been trademarked by their respective owners.

The publication of this information does not imply that any licence is granted under any patent or other rights owned by CSR plc.

CSR reserves the right to make technical changes to its products as part of its development programme.

While every care has been taken to ensure the accuracy of the contents of this document, CSR cannot accept responsibility for any errors.

Life Support Policy and Use in Safety-critical Compliance

CSR's products are not authorised for use in life-support or safety-critical applications. Use in such applications is done at the sole discretion of the customer. CSR will not warrant the use of its devices in such applications.

Performance and Conformance

Refer to www.csrsupport.com for compliance and conformance to standards information.