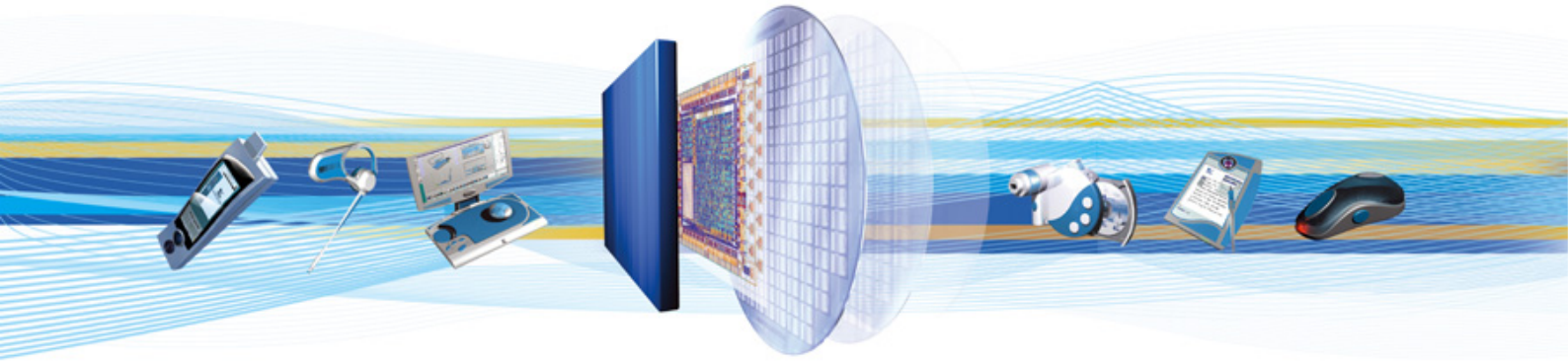
CSR Synergy Bluetooth 18.2.0

HIDD – Human Interface Device Client 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 API provided by the device side of the Human Interface Device (HID) Profile, i.e. the HID Device (HIDD). The HIDD conforms to the device side of the Bluetooth specification for the Human Interface Device Profile [HIDSPEC].

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- One HIDD instance only allows one connection to a HID host device
- Multiple HIDD instances can be created on physical device
- Bonding (pairing) is not handled by the HIDD profile

It is assumed that the reader has basic knowledge about Bluetooth® and Human Interface Device Profile [HIDSPEC]. Furthermore, it is assumed that the reader has basic understanding of the syntax and interpretation of Message Sequence Charts and State Diagrams, as these diagrams will be used throughout the description of the HIDD interface.

2 Description

This section gives an overview of the functionality and architecture of the HIDD.

2.1 Introduction

The Human Interface Device profile allows attaching input/output devices 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 HIDD supplies functionality for:

- Establishing connections with a HID host
- Registering the service records for the HID device
- Re-establishing connection if connection is dropped
- Unplugging HID devices
- Transport of HID data
- Support for multiple instances of the profile

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

2.2 Reference Model

The HIDD interfaces to the Connection Manager (CM) and to the Service Discovery Server (SDS) through the CM. The application must interface to the HIDD profile to communicate with a remote HID host device and interface to the Security Controller (SC) in order to handle encryption initiation.

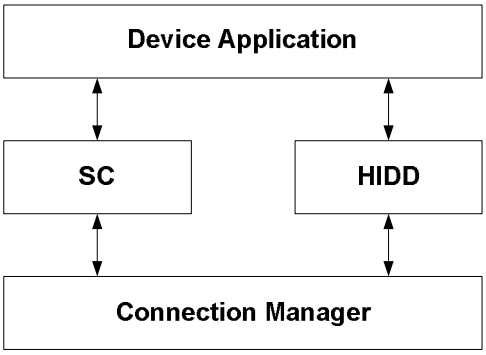


Figure 1: Reference model

The HIDD profile can have multiple instances, i.e. a physical device can include several HID devices, e.g. a mouse and a remote control.

2.3 Sequence Diagram

This section introduces a high level MSC for the HIDD profile. When the HIDD starts up it initialises and registers before entering the idle state where it can provide service to the application. The device application then activates the profile by supplying a service record, which determines connection responsibilities and device capabilities. The service record will remain registered until the profile is deactivated. After the profile has been activated, the device can enter page scan mode to be connectable. The very first connection between the device and a host must always be established by the host.

When a device is connected, HID data can be exchanged between the device and host, i.e. report and control data . 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. Furthermore, the device application has the responsibility of controlling the low power mode of the connection.

The device application is able to deactivate the HIDD in any state. The HIDD will disconnect if connected and return to idle without re-establishing the connection. If a virtual cable relation is available between the device and the host this will not be removed. The virtual cable relation between a device and a host can be removed by the unplug procedure, which also will ensure that both the host and the device disconnect without attempting to re-establish the connection.

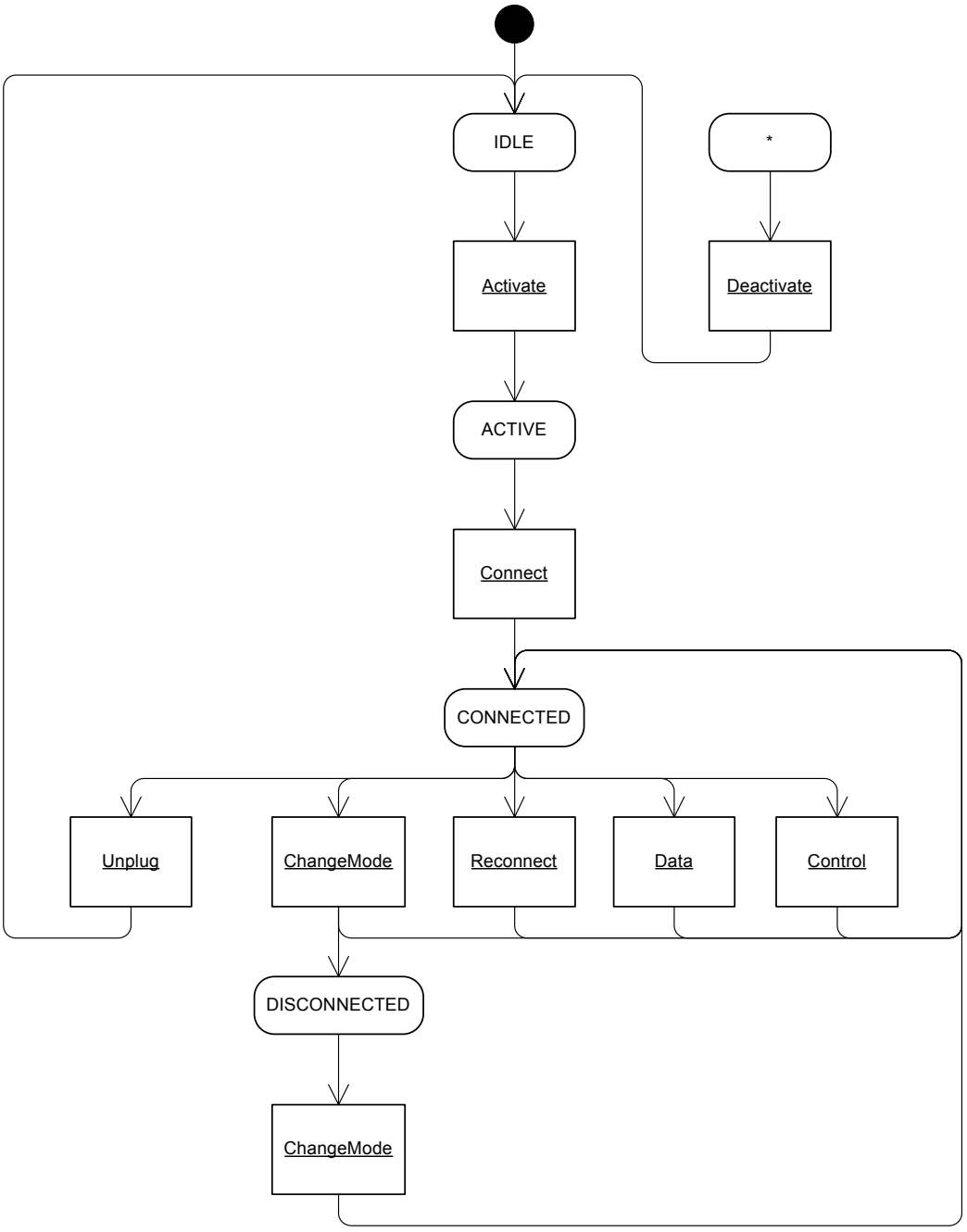


Figure 2: Sequence diagram

3 Interface Description

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

3.1 Activate Device

The HIDD service is activated using the CSR_BT_HIDD_ACTIVATE_REQ supplying the service record. The first time the profile registers the service record and puts the device into page scan mode to be able to accept a connection from a HID host. When in page scan mode the CSR_BT_HIDD_ACTIVATE_CFM is sent to the application. The HIDD is activated until deactivated or page scan timeout occurs.

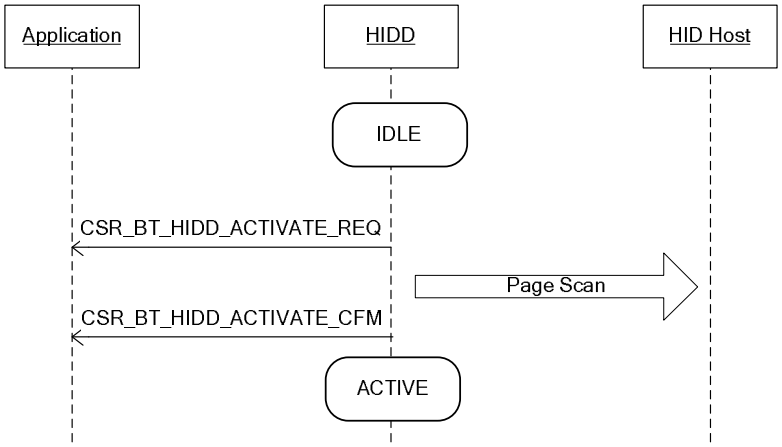


Figure 3: Application activates HIDD for the first time entering page scan

After first time activation the application makes a CSR_BT_HIDD_ACTIVATE_REQ with a known host device address. The action will dependent on whether the HIDReconnectInitiate attribute is set or not, i.e. if not set the HIDD registers and sets the device in page scan mode as described above or if set the HIDD initiates a connection with the known host. In this case the CSR_BT_HIDD_ACTIVATE_CFM is sent right after the connect request is sent to the lower layer.

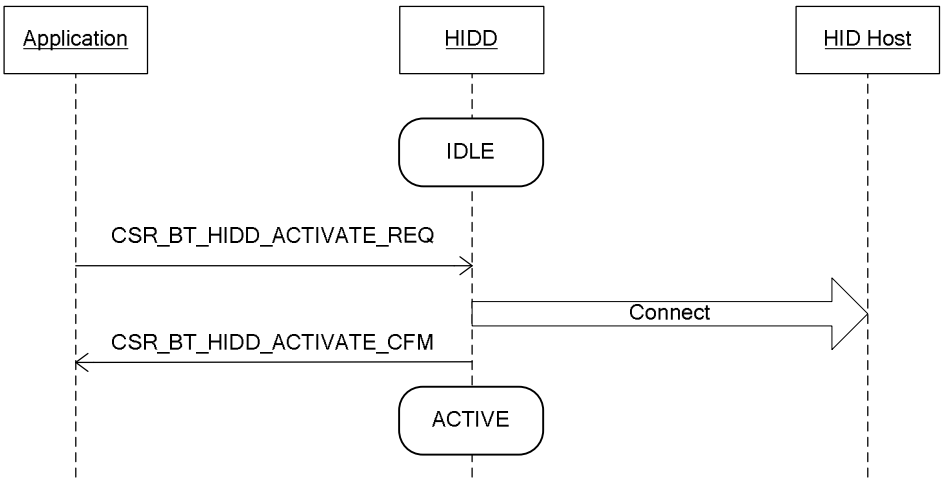


Figure 4: Application activates the HIDD profile initiating a connection

3.2 Connect

The connection is established in active state. Dependent on action taken to activate the HIDD, see Figure 3 and Figure 4, the connection is established from a HID host connect indication or from a connect confirm reply to the connect request from the HIDD. The application receives a CSR_BT_HIDD_STATUS_IND regarding the connection status.

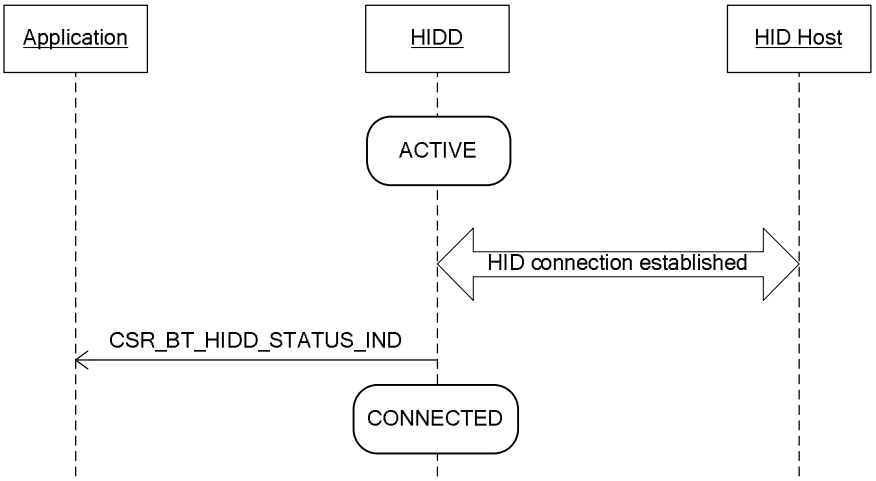


Figure 5: Connection either established by initiative from the HID host or HID device

3.3 Data Transfer

While the HIDD is connected, the application can request data to be sent to the HID host with the CSR_BT_HIDD_DATA_REQ. Once the data is sent to the lower layers a CSR_BT_HIDD_DATA_CFM is sent to the application. The data confirm does not imply that the data has been received on the HID host just the application is allowed to send a new data request to the HIDD.

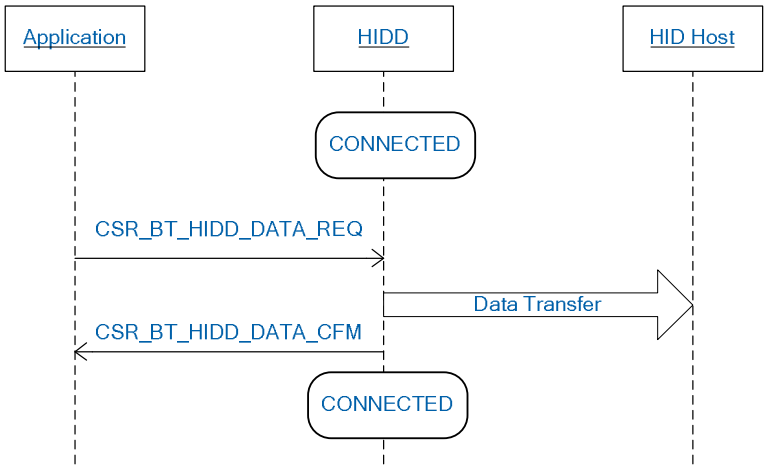


Figure 6: HID device initiated data transfer

Data can also be sent from the HID host. The HIDD forwards the data to the application with a CSR_BT_HIDD_DATA_IND.

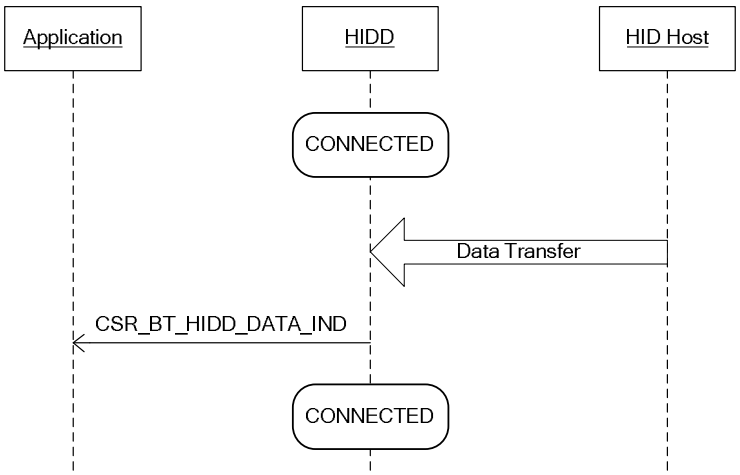


Figure 7: HID host initiated data transfer

3.4 Device Control

The HIDD can receive data requests for device control and these are sent to the application in a CSR_BT_HIDD_CONTROL_IND. The response from the application is sent with a CSR_BT_HIDD_CONTROL_RES.

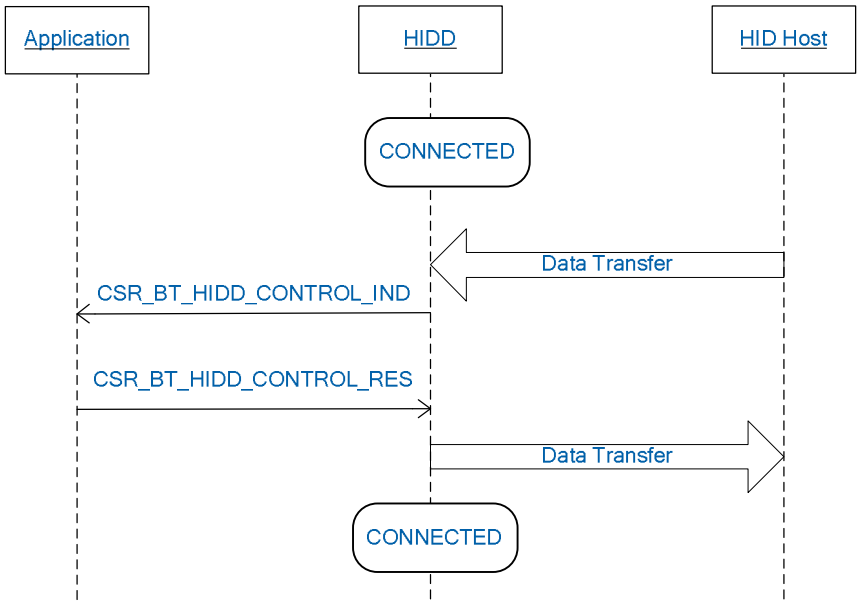


Figure 8: Control indication to HID device from the HID host

3.5 Mode Change

It is the application's own responsibility to control the low power mode of the HID device, i.e. the application can change the mode between Active, Sniff and Disconnect, where the Active mode and Sniff mode correspond to the modes offered by the Bluetooth Link Manager. Furthermore, it is possible to request a Disconnect mode to be able to disconnect the device entirely from the host connected.

The application changes the low power mode by sending CSR_BT_HIDD_MODE_CHANGE_REQ to the HIDD. In connected state requesting to change to either active or sniff mode, the mode is changed and CSR_BT_HIDD_MODE_CHANGE_CFM is sent to the application.

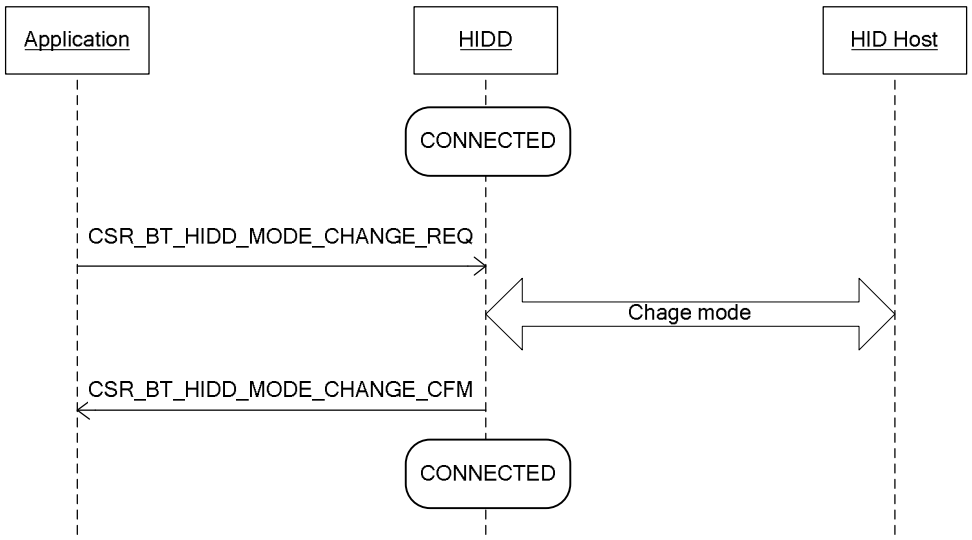


Figure 9: The application sets the HIDD device low power mode into active or sniff mode

If the CSR_BT_HIDD_MODE_CHANGE_REQ is sent with Disconnect mode request, the HIDD disconnects the connection to the host and replies with the CSR_BT_HIDD_MODE_CHANGE_CFM after disconnect completes.

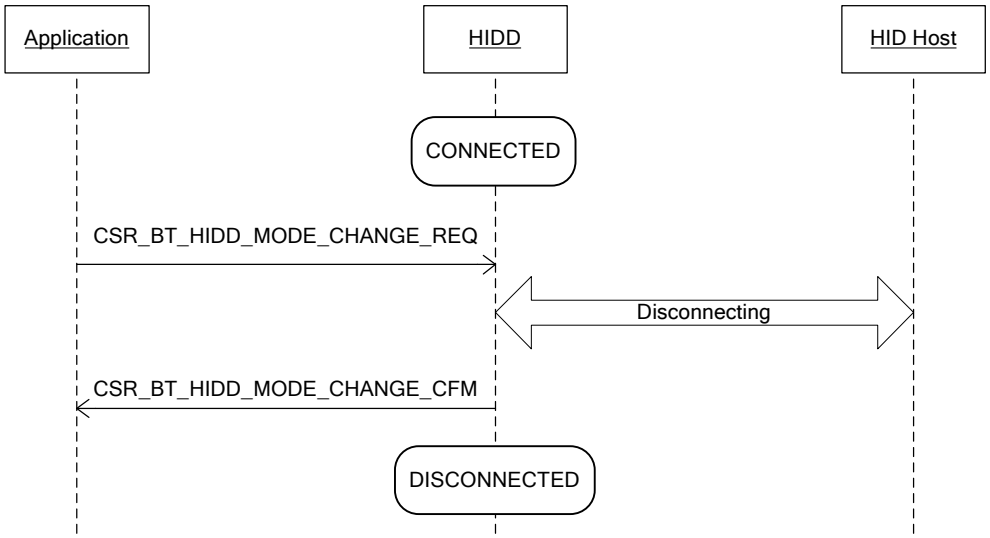


Figure 10: The application sets the HIDD device low power mode into disconnected mode

In disconnected state the application must send CSR_BT_HIDD_MODE_CHANGE_REQ to either Sniff or Active mode to get the device back into connected state to be able to send further requests. When the HIDD has established the connection the CSR_BT_HIDD_MODE_CHANGE_CFM is sent the application.

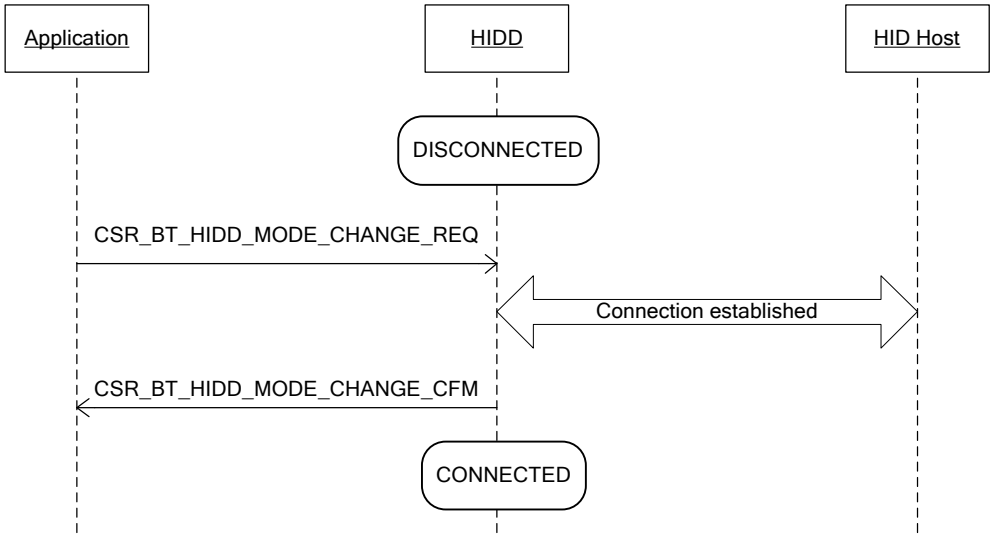


Figure 11: The application sets the HIDD device back into active or sniff mode

3.6 Reconnect

If a connection to a HID host is terminated unexpectedly the HIDD will send a CSR_BT_HIDD_STATUS_IND to inform the application that the HIDD is reconnecting. When the connection is re-established the HIDD sends a new CSR_BT_HIDD_STATUS_IND with information that the HIDD is connected.

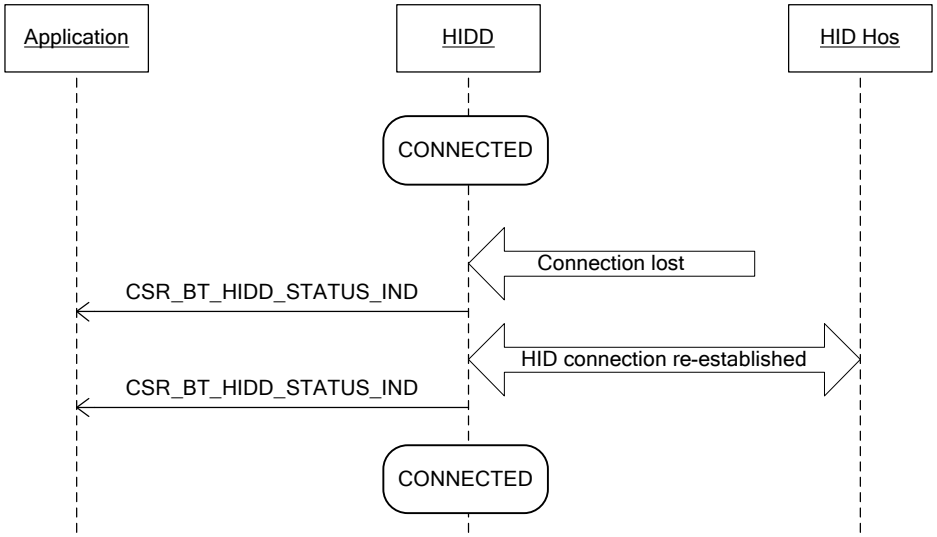


Figure 12: Re-establish lost connection

3.7 Unplug Virtual Cable

If the HIDVirtualCable attribute of the HID profile is set, both the device and the host are able to request unplug. The application sends the CSR_BT_HIDD_UNPLUG_REQ to initiate an unplug from the HIDD and after the host has disconnected the CSR_BT_HIDD_UNPLUG_CFM is returned to the application.

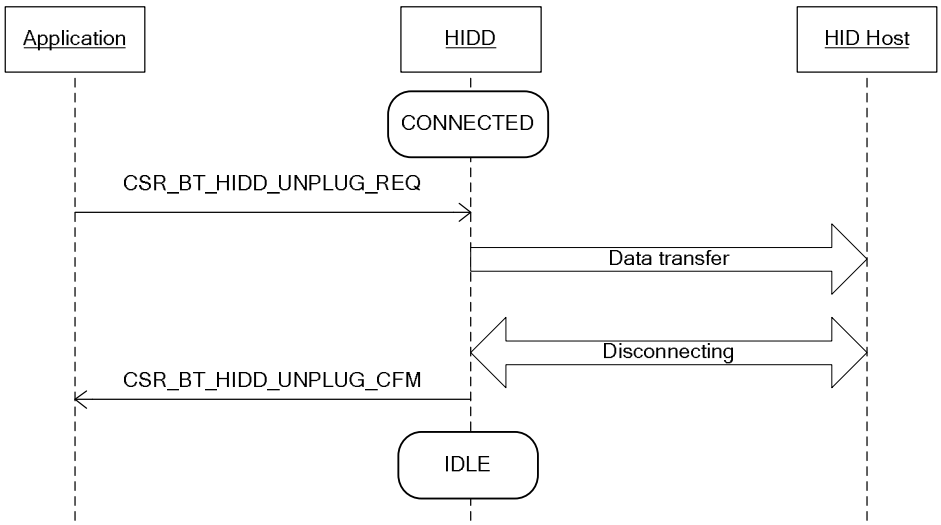


Figure 13: HID device initiated unplug of virtual cable

When unplug is requested by the host, the HIDD disconnects and sends the CSR_BT_HIDD_UNPLUG_IND to the application.

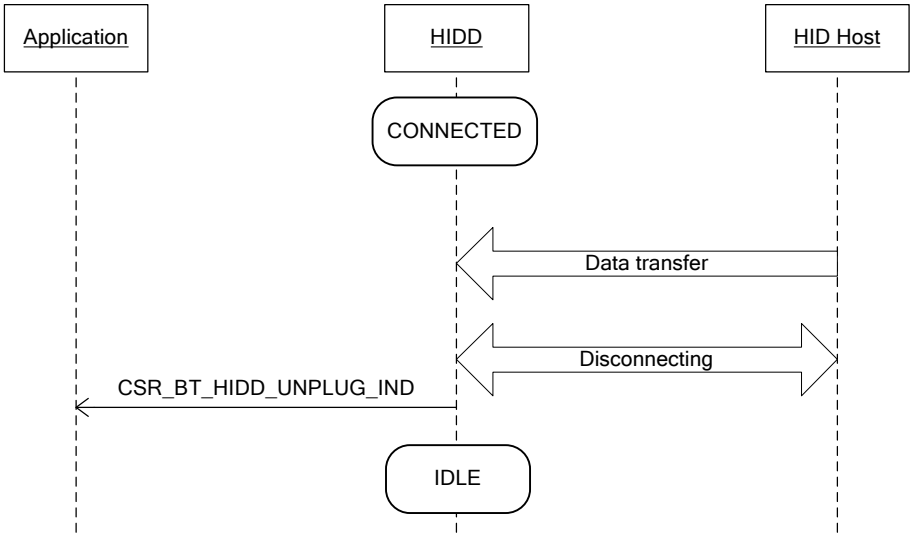


Figure 14: HID host initiated unplug of virtual cable

3.8 Deactivate Device

The application can send CSR_BT_HIDD_DEACTIVATE_REQ any time after activate request is confirmed to deactivate the service and unregister the service record. The HIDD disconnects potential connection and HIDD does not reconnect, then the CSR_BT_HIDD_DEACTIVATE_CFM is sent to the application. Note that the deactivate request does not unplug the virtual cable.

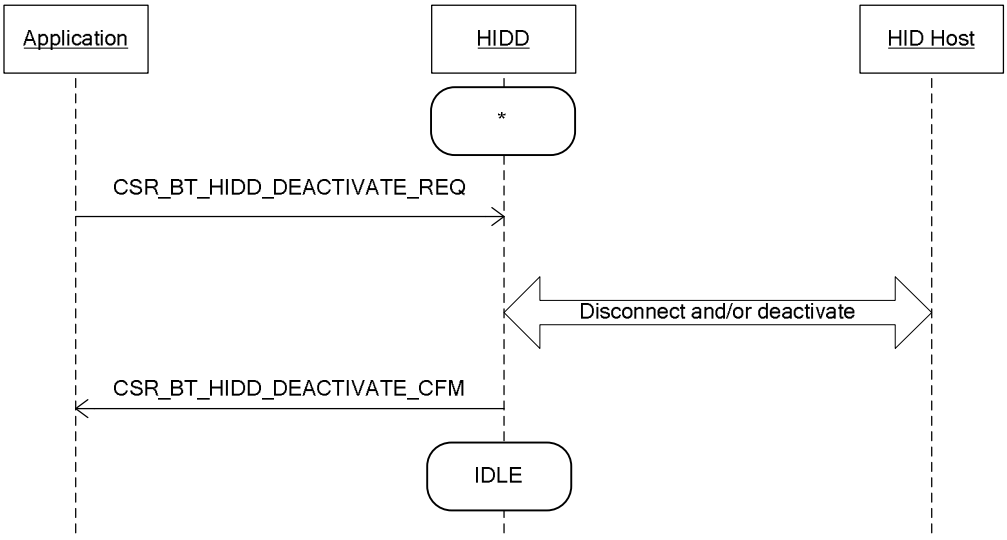


Figure 15: The HIDD profile disconnect if connected and deactivates

4 Human Interface Device Primitives

This section gives an overview of the primitives and parameters in the interface. Source code version can be found in the corresponding csr_bt_hidd_prim.h file.

4.1 List of All Primitives

Primitives:	Reference:
CSR_BT_HIDD_ACTIVATE_REQ	See section 4.2
CSR_BT_HIDD_ACTIVATE_CFM	See section 4.2
CSR_BT_HIDD_DEACTIVATE_REQ	See section 4.3
CSR_BT_HIDD_DEACTIVATE_CFM	See section 4.3
CSR_BT_HIDD_STATUS_IND	See section 4.4
CSR_BT_HIDD_CONTROL_IND	See section 4.5
CSR_BT_HIDD_CONTROL_RES	See section 4.5
CSR_BT_HIDD_DATA_REQ	See section 4.6
CSR_BT_HIDD_DATA_IND	See section 4.6
CSR_BT_HIDD_DATA_CFM	See section 4.6
CSR_BT_HIDD_UNPLUG_REQ	See section 4.7
CSR_BT_HIDD_UNPLUG_CFM	See section 4.7
CSR_BT_HIDD_UNPLUG_IND	See section 4.7
CSR_BT_HIDD_MODE_CHANGE_REQ	See section 4.8
CSR_BT_HIDD_MODE_CHANGE_CFM	See section 4.8
CSR_BT_HIDD_SECURITY_IN_REQ	See section 4.9
CSR_BT_HIDD_SECURITY_IN_CFM	See section 4.9
CSR_BT_HIDD_SECURITY_OUT_REQ	See section 4.9
CSR_BT_HIDD_SECURITY_OUT_CFM	See section 4.9

Table 1: List of all primitives

4.2 CSR_BT_HIDD_ACTIVATE

Primitives \ Parameters	type	appHandle	*qosCtrl	qosCtrlCount	*qosIntr	qosIntrCount	flushTimeout	deviceAddr	deviceIdSdpLen	*deviceIdSdp	hidSdpLen	*hidSdp	result Code	resultSupplier
CSR_BT_HIDD_ACTIVATE_REQ	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
CSR_BT_HIDD_ACTIVATE_CFM	✓												✓	✓

Table 2: CSR_BT_HIDD_ACTIVATE Primitives

Description

These signals are used for activating the HID Device profile. The same signal is used for re-activating after a connect attempt fail.

The profile can be activated either by the following process:

- Registration of the HID service record in the service discovery database
- Enabling page scan

or when the host device address is known and the HIDReconnectInitiate attribute is set true by:

- Requesting a connection to the known HID host

PLEASE NOTE!

The CSR_BT_HIDD_ACTIVATE_REQ primitive will change in a future release of CSR Synergy Bluetooth. The registration of the DI service record has been removed from the HIDD interface, and gotten its own interface.

Please see the api-0136-di documentation for further details.

It is recommended to start using the new interface now, to avoid backward compatibility issues in the future!

Parameters

type	Signal identity CSR_BT_HIDD_ACTIVATE_REQ/CFM.
appHandle	Identity of the device application
*qosCtrl	Wanted quality of service for the HID control channel signaling. If NULL, default values will be used.
qosCtrlCount	Number of elements in *qosCtrl*. This value is either 0 if *qosCtrl* is NULL, 1 otherwise.
*qosIntr	Wanted quality of service for the HID interrupt channel signaling. If NULL, default values will be used.
qosIntrCount	Number of elements in *qosIntr*. This value is either 0 if *qosIntr* is NULL, 1 otherwise.
flushTimeout	Wanted L2CAP flush timeout value. Recommended value: 0xFFFF (infinite).
deviceAddr	Bluetooth address of the remote device. Only used if host is already known.

deviceIdSdpLen	**PLEASE NOTE: THIS FIELD WILL BE REMOVED IN FUTURE VERSIONS OF CSR SYNERGY BLUETOOTH**. Please set the value to 0 (zero).
* deviceIdSdp	**PLEASE NOTE: THIS FIELD WILL BE REMOVED IN FUTURE VERSIONS OF CSR SYNERGY BLUETOOTH.** Please set the value to NULL.
hidSdpLen	Length of device hid service record
* hidSdp	HID service record representing the HID profile. May be Null in a re-activate request with a known host address, i.e. the application did not deactivate the service but the device lost the connection.
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

Library Function

The following library function is provided by CSR Synergy Bluetooth for HIDD, and can be used for creating the proper message and sending it to the HID device profile. The library functions are defined in csr_bt_hidd_lib.h.

```
void CsrBtHiddActivateReqSend(   CsrSchedQid            hiddInstanceId,
                                 CsrSchedQid              appHandle,
                                 qos_t                 *qosCtrl,
                                 qos_t                 *qosIntr,
                                 CsrUint16              flushTimeout,
                                 deviceAddr_t          deviceAddr,
                                 CsrUint16                deviceIdSdpLen,
                                 CsrUint8               *deviceIdSdp,
                                 CsrUint16                hidSdpLen,
                                 CsrUint8               *hidSdp );
```

hiddInstanceId	The queue of the HIDD instance to send the message to.
appHandle	The same as in parameters above.
* qosCtrl	The same as in parameters above.
qosCtrlCount	The same as in parameters above.
* qosIntr	The same as in parameters above.
qosIntrCount	The same as in parameters above.
flushTimeout	The same as in parameters above.
deviceAddr	The same as in parameters above.
deviceIdSdpLen	The same as in parameters above
* deviceIdSdp	The same as in parameters above.
hidSdpLen	The same as in parameters above
* hidSdp	The same as in parameters above.

4.3 CSR_BT_HIDD_DEACTIVATE

Primitives \ Parameters	type	resultCode	resultSupplier
CSR_BT_HIDD_DEACTIVATE_REQ	✓		
CSR_BT_HIDD_DEACTIVATE_CFM	✓	✓	✓

Table 3: CSR_BT_HIDD_DEACTIVATE Primitives

Description

Deactivate a service that has been activated previously. The service will no longer be connectable and returned to idle.

Parameters

type	Signal identity CSR_BT_HIDD_DEACTIVATE_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

Library Function

The following library function is provided by CSR Synergy Bluetooth for HIDD, and can be used for creating the proper message and sending it to the HID device profile. The library functions are defined in csr_bt_hidd_lib.h.

```
void CsrBtHiddDeactivateReqSend( CsrSchedQid            hiddInstanceId);
```

hiddInstanceId	The queue of the HIDD instance to send the message to.

4.4 CSR_BT_HIDD_STATUS

Primitives \ Parameters	type	deviceAddr	status	btConnId
CSR_BT_HIDD_STATUS_IND	✓	✓	✓	✓

Table 4: CSR_BT_HIDD_DEACTIVATE Primitives

Description

The status signal is used for informing the application of a change in the connection status.

Parameters

type — Signal identity CSR_BT_HIDD_DEACTIVATE_REQ/CFM.

deviceAddr — Bluetooth address of the remote device

status — Status of the connection,

CSR_BT_HIDD_DISCONNECTED: Connection disconnected returned to idle

CSR_BT_HIDD_CONNECTED: Connection established

CSR_BT_HIDD_CONNECT_FAILED: Connection establishment failed returned to idle

CSR_BT_HIDD_UNREGISTER_FAILED: Unregistration of service record failed returned to idle

CSR_BT_HIDD_RECONNECTING: The connection was lost for an unknown reason has returned to idle and re-establishing the connection (depending on service settings).

btConnId — Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend`-function.

4.5 CSR_BT_HIDD_CONTROL

Primitives \ Parameters	type	transactionType	parameter	dataLen	*data
CSR_BT_HIDD_CONTROL_IND	✓	✓	✓	✓	✓
CSR_BT_HIDD_CONTROL_RES	✓	✓	✓	✓	✓

Table 5: CSR_BT_HIDD_CONTROL Primitives

Description

These signals are use to send a control message from the host and respond.

Parameters

type	Signal identity CSR_BT_HIDD_CONTROL_IND/RES.
transactionType	The transaction type request. The possible values are defined in [HIDSPEC] on page 57. CSR Synergy Bluetooth profiles predefined macros in the file csr_bt_hidd_prim.h.
parameter	The parameter is dependent of transaction type. The parameter value is described in [HIDSPEC] on page 57. CSR Synergy Bluetooth provides pre-defined macros in csr_bt_hidd_prim.h.
dataLen	Length of data, including a one byte header. Only use if transaction type includes data.
*data	Data. The first byte of the data is reserved for header and should be left unused. Only included if transaction type includes data.

Library Function

The following library function is provided by CSR Synergy Bluetooth for HIDD, and can be used for creating the proper message and send it to the HID device profile. The library functions are defined in csr_bt_hidd_lib.h.

```
void CsrBtHiddControlResSend( CsrSchedQid            hiddInstanceId,
                              HiddTransactionType    transactionType,
                              HiddParameterType      parameter,
                              CsrUint16               dataLen,
                              CsrUint8                *data);
```

hiddInstanceId	The queue of the HIDD instance to send the message to.
transactionType	The same as in parameters above.
parameter	The same as in parameters above.
dataLen	The same as in parameters above.
*data	The same as in parameters above.

4.6 CSR_BT_HIDD_DATA

Parameters / Primitives	type	reportType	reportLen	*report	resultCode	resultSupplier
CSR_BT_HIDD_DATA_REQ	✓	✓	✓	✓		
CSR_BT_HIDD_DATA_IND	✓	✓	✓	✓		
CSR_BT_HIDD_DATA_CFM	✓				✓	✓

Table 6: CSR_BT_HIDD_DATA Primitives

Description

These signals are use to send and receive data reports. Please note that confirm is used for flow control, meaning that the application must wait for a confirm before it can send a new request.

Parameters

type	Signal identity CSR_BT_HIDD_DATA_REQ/IND/CFM.
reportType	ReportType must be input for requests and output for indications. The report type is described in [HIDSPEC] on page 70. CSR Synergy Bluetooth provides predefined macros for the possible values in the file csr_bt_hidd_prim.h.
reportLen	Length of report, including a one byte header.
*report	Report. The first byte of the data is reserved for header and should be left unused. Input report for data request and output report for data indication.
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

Library Function

The following library function is provided by CSR Synergy Bluetooth for HIDD, and can be used for creating the proper message and sending it to the HID device profile. The reportType is left out because data requests always are input reports sent to the host. The library functions are defined in csr_bt_hidd_lib.h.

```
void CsrBtHiddDataReqSend (   CsrSchedQid      hiddInstanceId,
                              CsrUint16         reportLen,
                              CsrUint8          *report);
```

hiddInstanceId	The queue of the HIDD instance to send the message to.
reportLen	The same as in parameters above.
*report	The same as in parameters above.

4.7 CSR_BT_HIDD_UNPLUG

Primitives \ Parameters	type	deviceAddr	result Code	resultSupplier
CSR_BT_HIDD_UNPLUG_REQ	✓	✓		
CSR_BT_HIDD_UNPLUG_CFM	✓		✓	✓
CSR_BT_HIDD_UNPLUG_IND	✓	✓	✓	✓

Table 7: CSR_BT_HIDD_UNPLUG Primitives

Description

These signals are used, when the HIDVirtualCable is set true, to request an unplug of the virtual cable that implies a disconnect.

Parameters

type	Signal identity CSR_BT_HIDD_UNPLUG_REQ/CFM/IND.
deviceAddr	Bluetooth address of the remote device.
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

Library Function

The following library function is provided by CSR Synergy Bluetooth for HIDD, and can be used for creating the proper message and sending it to the HID device profile. The library functions are defined in csr_bt_hidd_lib.h.

```
void CsrBtHiddUnplugReqSend ( CsrSchedQid      hiddInstanceId,
                              deviceAddr_t     deviceAddr);
```

hiddInstanceId	The queue of the HIDD instance to send the message to.
deviceAddr	The same as in parameters above.

4.8 CSR_BT_HIDD_MODE_CHANGE

Primitives \ Parameters	type	mode	resultCode	resultSupplier
CSR_BT_HIDD_MODE_CHANGE_REQ	✓	✓		
CSR_BT_HIDD_MODE_CHANGE_IND	✓	✓	✓	✓

Table 8: CSR_BT_HIDD_MODE_CHANGE Primitives

Description

These signals are used for changing the low power mode of the HIDD.

Parameters

type	Signal identity CSR_BT_HIDD_MODE_CHANGE_REQ/IND.
mode	Mode requested or confirmed, 0: active, 2: sniff, 4: disconnect
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

Library Function

The following library function is provided by CSR Synergy Bluetooth for HIDD, and can be used for creating the proper message and sending it to the HID device profile. The library functions are defined in csr_bt_hidd_lib.h.

```
void CsrBtHiddUnplugReqSend ( CsrSchedQid              hiddInstanceId,
                              HiddPowerModeType      mode);
```

hiddInstanceId	The queue of the HIDD instance to send the message to.
mode	The same as in parameters above.

4.9 CSR_BT_HIDD_SECURITY_IN / OUT

Primitives \ Parameters	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_HIDD_SECURITY_IN_REQ	✓	✓	✓		
CSR_BT_HIDD_SECURITY_IN_CFM	✓			✓	✓
CSR_BT_HIDD_SECURITY_OUT_REQ	✓	✓	✓		
CSR_BT_HIDD_SECURITY_OUT_CFM	✓			✓	✓

Table 9: CSR_BT_HIDD_SECURITY_IN and CSR_BT_HIDD_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_HIDD_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, Document no. api-0102-sc	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
HIDD	HID Device
HIDH	HID Host
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.