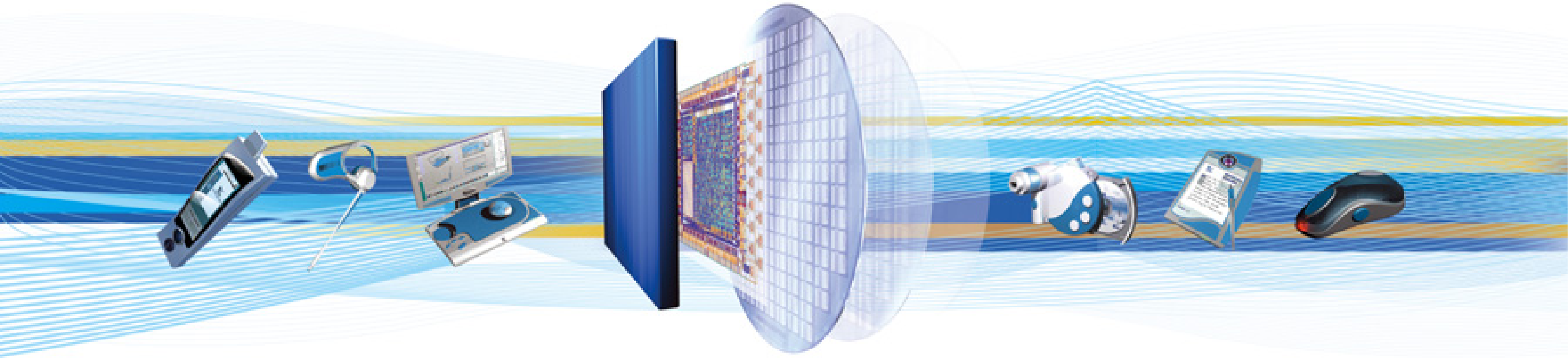
CSR Synergy Bluetooth 18.2.0

HF – Hands-Free

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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Contents

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

List of Figures

List of Tables

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the hands-free side of the Hands-Free Profile, henceforward called HF. This implementation also supports the headset side of the Headset Profile, henceforward called the HS. The HF requirements and functionality are specified in [HF], and the requirements and functionality for the HS are specified in [HS].

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- There is a secure and reliable transport between the profile part, i.e. the HF/HS profile and the application layer

Knowledge of the Headset profile, HF profile and Hands-Free Profile Application Guideline [CCAP] is assumed since part of the functionality is application layer specific and must be implemented in the application layer.

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

2 Description

2.1 Introduction

The HF/HS manager supplies the interface for applications that should provide the functionality and conform to the hands-free side of the Hands-Free Profile and the Headset side of the Headset Profile. The HF is implemented as specified in the Hands-Free Profile [HF] with the addition of the requirements as specified in the Bluetooth® Hands-Free profile Application Guidelines [CCAP]. The HS is implemented as specified in [HS].

The HS/HF layer provides functionality for:

- Establishing and maintaining a service level connection between the HF profile layer and a peer device. The peer device must conform to the Hands-Free Gateway (HFG) side as defined in [HF]
- Establishing a connection to a peer device that conforms to the Headset Audio Gateway (HAG) as defined [HS]
- Sending and receiving AT-commands and result codes between a HF and a HFG
- Sending and receiving AT-commands and result codes between a HS and a HAG
- The application layer, i.e. automatic and transparent handling of low power modes

2.2 Reference Model

The application can interface to the HF/HS profile, the Security Controller, and the Service Discovery module. The Service Discovery module has an interface that can be used for searching for remote devices. The Security Controller has an interface that is used for bonding and pairing.

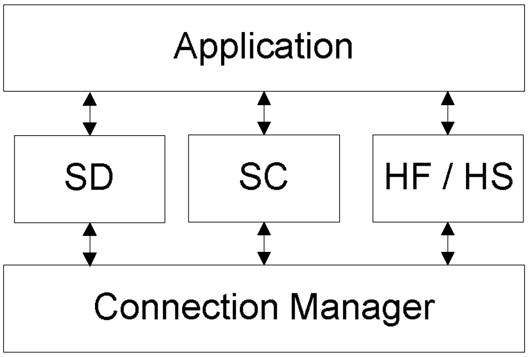


Figure 1: Reference model

Thus e.g. bonding, which is normally carried out before any other transaction, must be accomplished before setting up any transactions using the HF/HS interface as described below.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

2.3 Sequence Overview

Figure 2 outlines the basic functionality of the HF/HS profile.

A normal scenario is that a connection between a HFG or a HAG and the HF/HS profile side is established after pairing of the devices. The connection can be established from both sides. Pairing is described in [SC] and is outside the scope of this document.

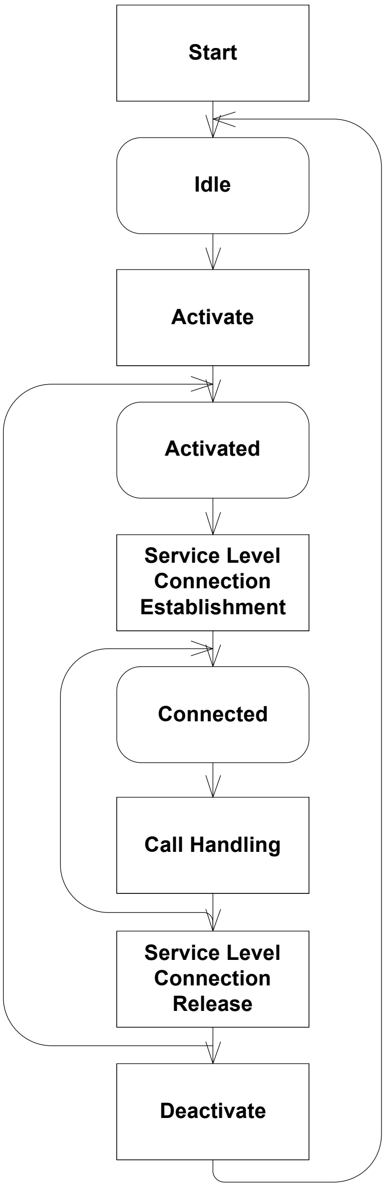

Start
Idle
Activate
Activated
Service Level Connection Establishment
Connected
Call Handling
Service Level Connection Release
Deactivate

Figure 2: Sequence overview

Upon completion of the service level connection to a HFG a call can be initiated. Call handling includes AT-command exchange and setting up a full duplex audio channel between the two devices. Once a call is terminated the service level connection may be maintained for another call setup or may be released. If connection is made to a HAG, call handling is limited to answering a call and requesting audio.

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

3 Interface Description

3.1 Activation

Before the profile can be used, the application must send the CSR_BT_HF_ACTIVATE_REQ message to register which supported features the HF part is supporting and indicate if the HS part should support remote audio control. The csr_bt_hf_prim.h file holds some defines that can be added together to describe the capabilities of the HF profile.

During activation, it is possible for the application to specify whether the HS profile shall be supported or not by the HF/HS manager, and whether only HF connections and functionality or both HF and HS shall be available in the device.

When the profile has been activated the CSR_BT_HF_ACTIVATE_CFM message will be sent to the application.

3.2 Connection Establishment

Once the profile layer has been activated with the CSR_BT_HF_ACTIVATE_REQ message the HF/HS profile manager is able to accept a connection from a remote device or connect to another device.

Please note that whether or not the Bluetooth device will be discoverable, i.e. can be found by other Bluetooth devices, it must be controlled by the application. For more information, please refer to [CM]. After initialization of CSR Synergy Bluetooth the Bluetooth® device is set up to be discoverable.

The HF/HS manager stays connectable until a remote device has connected or the profile is deactivated by the application layer. The CSR_BT_HF_SERVICE_CONNECT_IND is an indication that a connection is established.

Once a connection is established, the HF/HS profile will, transparently for the application layer, make use of low power modes according to the rules described in [CCAP] and [HS]; this implies use of sniff and park mode if supported by the HFG. The interface towards the application is identical whether or not the HFG supports low power modes.

Note: the application is not allowed to send any signals except CSR_BT_HF_DISCONNECT_REQ before it receives a CSR_BT_HF_SERVICE_CONNECT_IND.

3.2.1 Accept Connection from Hands-Free Gateway

If a HFG device connects to the HF/HS profile it will be indicated to the application with a CSR_BT_HF_SERVICE_CONENCT_IND signal. Figure 3 shows two of the parameters returned in this signal (all parameters can be found in section 4.4). The result parameter indicates if the connection establishment was a success and the connectionType parameter will indicate which type of connection that has been established. When a HFG is connected the connectionType parameter will be CSR_BT_HF_CONNECTION_HF, which is defined in csr_bt_hf_prim.h. Another important parameter contained in the CSR_BT_HF_SERVICE_CONNECT_IND is the connection ID, which shall be kept and used by the application for future communication through that specific connection.

When the application receives the CSR_BT_HF_SERVICE_CONNECT_IND message a remote device is connected to the HF/HS profile and the initial service discovery of remote supported features and network has been carried out. These parameters are returned in the CSR_BT_HF_SERVICE_CONNECT_IND.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

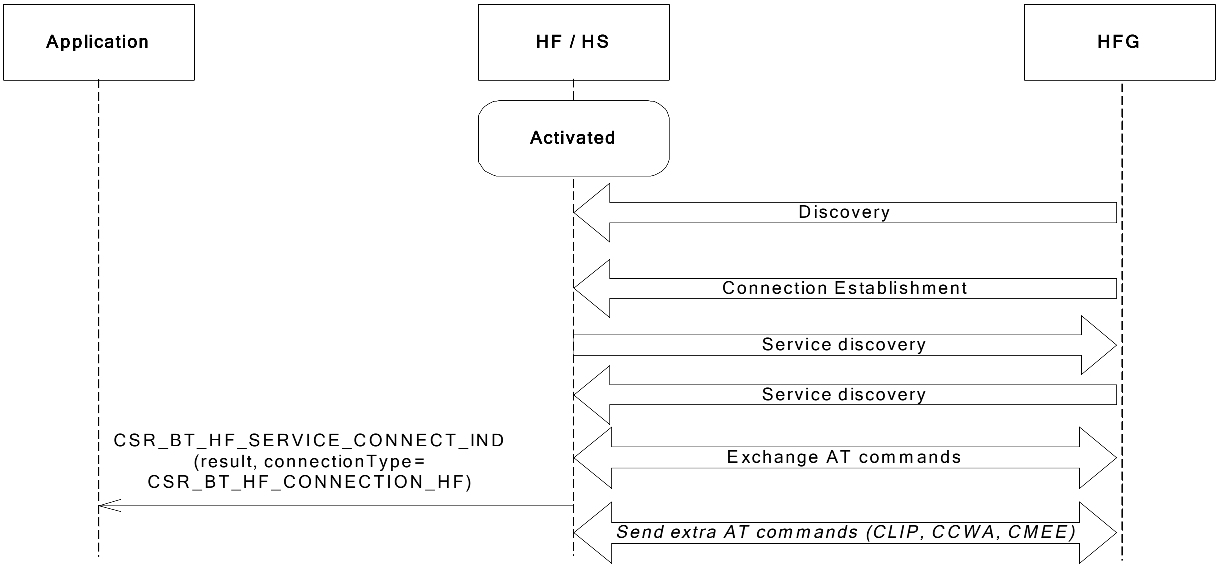
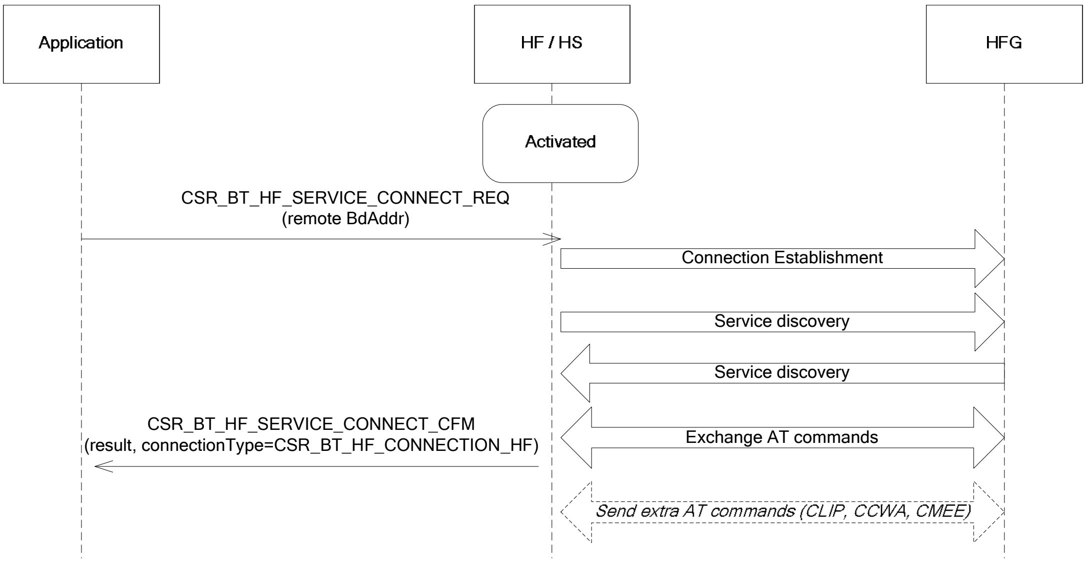


Figure 3: Accept connection from Hands-free Gateway

The rest of the AT sequence needed, before a complete service level connection is established, is sent by the profile layer after the CSR_BT_HF_SERVICE_CONNECT_IND message has been sent to the application (see section 4.4).

3.2.2 Establishing a Connection to a Hands-Free Gateway

In addition to the accept connection presented above, it is also possible to initiate a connection to a remote HAG or HFG device. Sending a CSR_BT_HF_SERVICE_CONNECT_REQ from the application will accomplish this. Upon receiving the CSR_BT_HF_SERVICE_CONNECT_REQ the profile layer will perform a service discovery search on the device with the Bluetooth address specified in the request. The profile will search for both the HFG and the HAG service on the remote device. If a HFG is found (and no hands-free device is connected) the profile will connect to this service, otherwise it will try and connect to the HAG if this service is available on the remote device. Completion of the connection is indicated with a CSR_BT_HF_SERVICE_CONNECT_CFM, the service that has been connected will be indicated.



Application
HF / HS
HFG
Activated
CSR_BT_HF_SERVICE_CONNECT_REQ
(remote BdAddr)
Connection Establishment
Service discovery
Service discovery
CSR_BT_HF_SERVICE_CONNECT_CFM
(result, connectionType=CSR_BT_HF_CONNECTION_HF)
Exchange AT commands
Send extra AT commands (CLIP, CCWA, CMEE)

Figure 4: Locally initiated connection establishment sequence

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Due to a race condition, it is possible that a CSR_BT_HF_SERVICE_CONNECT_IND may be received with a Bluetooth® device address that is different from the one issued in the connect request.

3.2.3 Accepting a Connection from a Headset Audio Gateway

When a HAG connects to the HF/HS profile it is indicated to the application with a CSR_BT_HF_SERVICE_CONNECT_IND. The connectionType parameter will be CSR_BT_HF_CONNECTION_HS (defined in csr_bt_hf_prim.h), and the result parameter will hold a result code (defined in csr_bt_profiles.h).

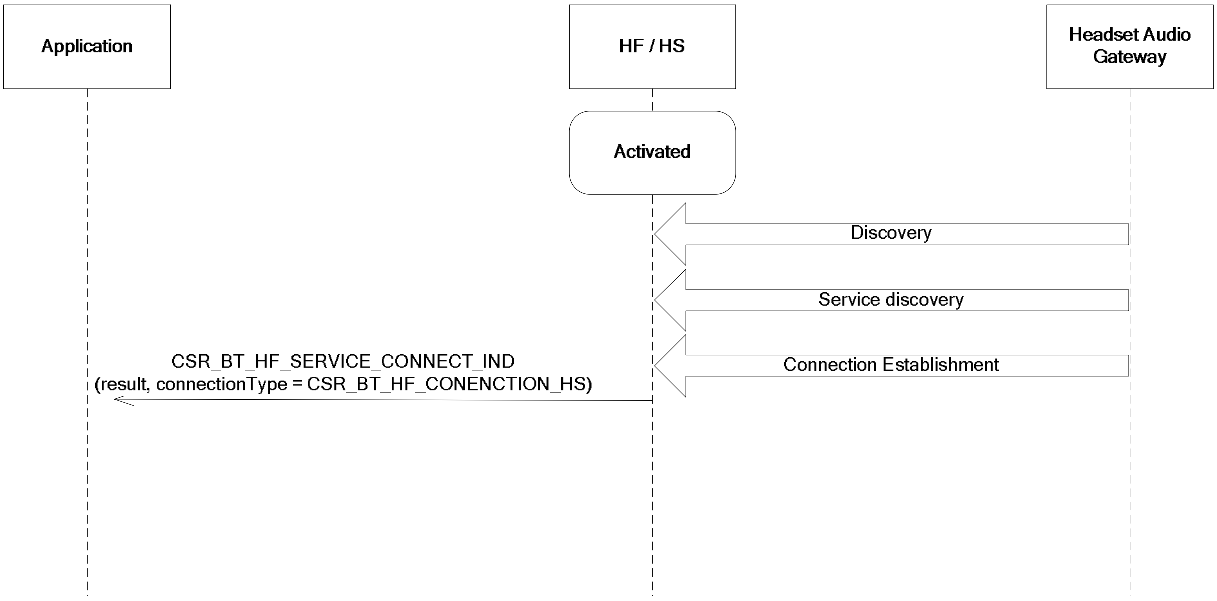


Figure 5: Accepting a connection from a Headset Audio Gateway

3.2.4 Connection Establishment to a Headset Audio Gateway

Connecting to a HAG device is done with the CSR_BT_HF_SERVICE_CONNECT_REQ message.

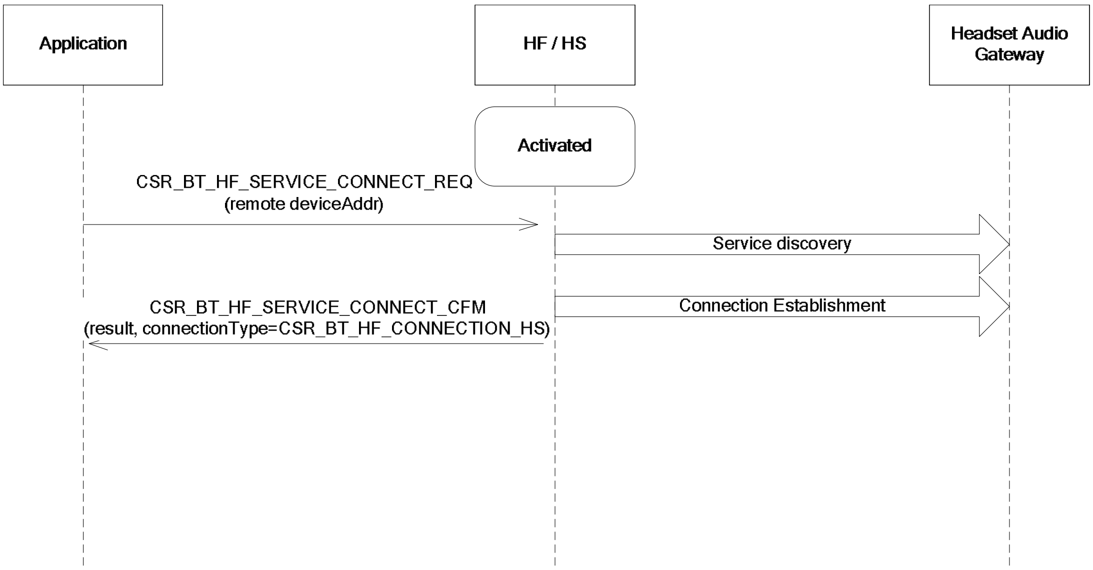


Figure 6: Connection to a Headset Audio Gateway

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

The remote device address in the connect request must be the address of a device with the Audio Gateway profile. If the remote device also supports HFG the HF/HS profile will connect to that service. In the scenario shown in Figure 6 the remote device only has the HAG service, and in this case the HF/HS profile will return a CSR_BT_HF_SERVICE_CONNECT_CFM where the connectionType parameter will be CSR_BT_HF_CONNECTION_HS.

3.2.5 Service Level Connection Completion

The [HF] specifies a series of AT commands that must be sent from the HF profile to the HFG before the Service Level Connection (SLC) has been established. The HF/HS profile sends these AT commands to the HFG and interprets the results. Figure 7 shows an example of establishing a SLC.

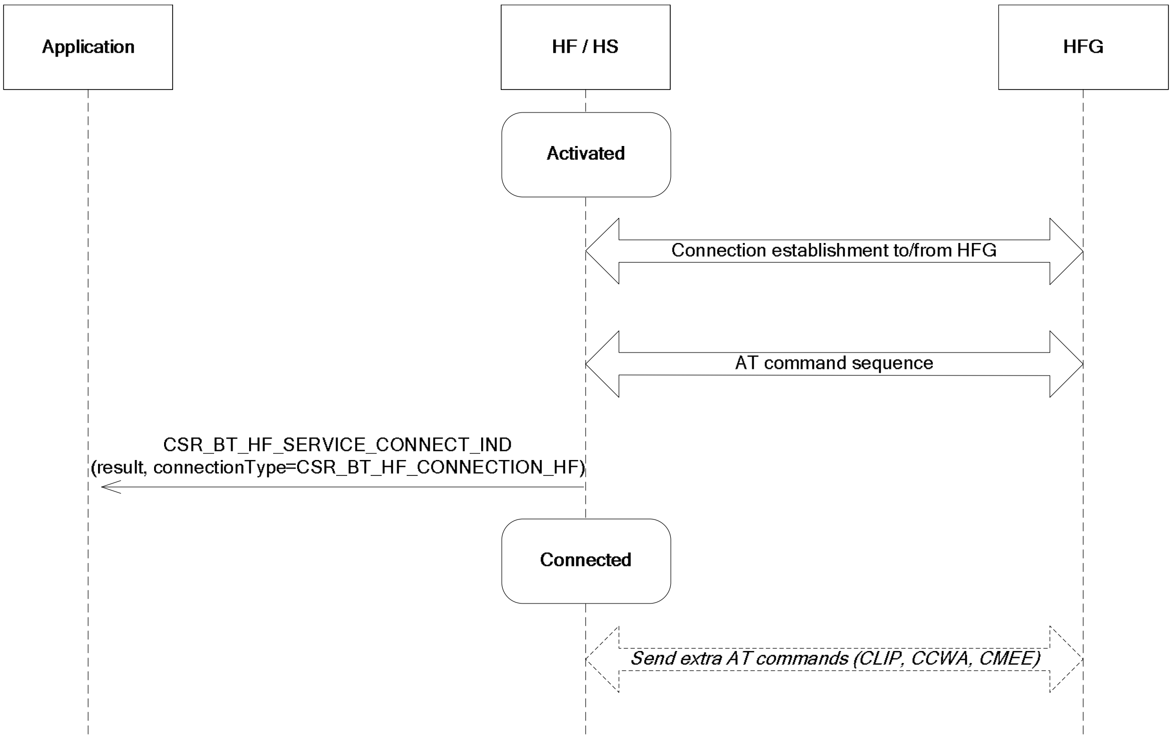

Application
HF / HS
HFG
Activated
Connection establishment to/from HFG
AT command sequence
CSR_BT_HF_SERVICE_CONNECT_IND
(result, connectionType=CSR_BT_HF_CONNECTION_HF)
Connected
Send extra AT commands (CLIP, CCWA, CMEE)

Figure 7: Example of establishing a Service Level Connection

During the establishment of SLC the HFG sends a list of supported Status Indicators and the current value of the indicators. The HF profile keeps the data received in these AT commands and delivers this information to the application in the CSR_BT_HF_SERVICE_CONNECT_IND (see chapter 4.4 for details).

Besides, if the HF has been activated without disabling automatic CLIP, CCWA and CMEE exchange and both parties claim support for these features, the commands needed to activate these features will be sent automatically to the HFG right after SLC establishment (see chapter 4.2 for details).

3.2.6 Using the Transparent AT feature in Connection Establishment

Normally the profile will handle the different AT-command sequences to complete a service level connection. However if the CSR_BT_HF_AT_MODE_TRANSPARENT_ENABLE bit is set in the config bitmask in Activation of the HF profile, then the application must perform the AT command exchange needed to establish an SLC.

When the basic serial communication is established, the HF Profile Specification requires that the HF issues a number of AT commands to inform about and request application layer specific information. The application layer must send any AT-commands using the CSR_BT_HF_AT_CMD_REQ command. Likewise the application layer will be receiving any responses from the HFG in a CSR_BT_HF_AT_CMD_IND. It is the responsibility of the application layer to interpret and validate the commands received. Figure 8 shows an example of how the

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Service Level connection is established when the CSR_BT_HF_AT_MODE_TRANPARENT_ENABLE config bit is set.

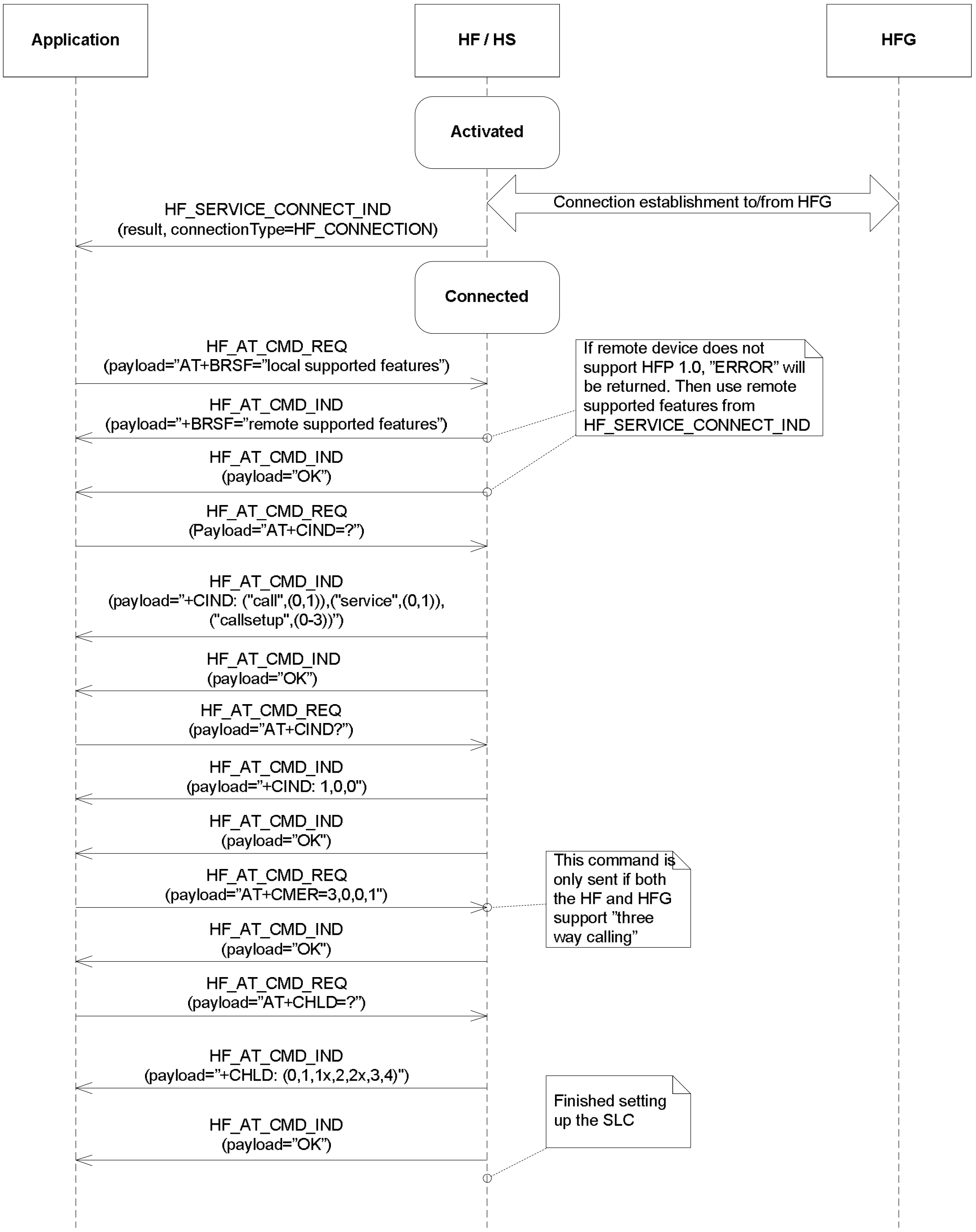

Application
HF / HS
HFG
Activated
Connection establishment to/from HFG
HF_SERVICE_CONNECT_IND
(result, connectionType=HF_CONNECTION)
Connected
HF_AT_CMD_REQ
(payload="AT+BRSF="local supported features")
If remote device does not support HFP 1.0, "ERROR" will be returned. Then use remote supported features from HF_SERVICE_CONNECT_IND
HF_AT_CMD_IND
(payload="+BRSF="remote supported features")
HF_AT_CMD_IND
(payload="OK")
HF_AT_CMD_REQ
(Payload="AT+CIND=?")
HF_AT_CMD_IND
(payload="+CIND: ("call",(0,1)),("service",(0,1)),
("callsetup",(0-3))")
HF_AT_CMD_IND
(payload="OK")
HF_AT_CMD_REQ
(payload="AT+CIND?")
HF_AT_CMD_IND
(payload="+CIND: 1,0,0")
HF_AT_CMD_IND
(payload="OK")
HF_AT_CMD_REQ
(payload="AT+CMER=3,0,0,1")
This command is only sent if both the HF and HFG support "three way calling"
HF_AT_CMD_IND
(payload="OK")
HF_AT_CMD_REQ
(payload="AT+CHLD=?")
HF_AT_CMD_IND
(payload="+CHLD: (0,1,1x,2,2x,3,4)")
Finished setting up the SLC
HF_AT_CMD_IND
(payload="OK")

Figure 8: Example of establishing a Service Level Connection in Tranparent AT mode

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

3.3 Call Handling

Once a connection is established, the application layer can initiate an outgoing call towards a HFG or HAG and a HFG or HAG can initiate incoming calls towards the HF/HS profile. The HF side of the HF/HS manager offers more advanced call handling features, since the HF profile offers more advanced features than the HS profile.

3.3.1 Incoming Call

When the application is receiving an incoming call, it receives a CSR_BT_HF_RING_IND. When the application receives this signal it will contain the connectionType parameter, which will indicate if the ring is from a HFG or a HAG. If the connection is from a HAG the application can accept the call with the CSR_BT_HF_ANSWER_REQ or choose to ignore the ring and thereby reject the call.

If the connection is from a HFG, the application can either accept or reject the call. If the application layer wants to answer the call it sends a CSR_BT_HF_ANSWER_REQ to the HF manager and the call will be established from the HFG with or without audio. If the application for any reason wants to reject the call, a CSR_BT_HF_REJECT_REQ is returned to the HF/HS manager. If the call is rejected the service level connection is maintained and the HFG will return a CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND to the HF.

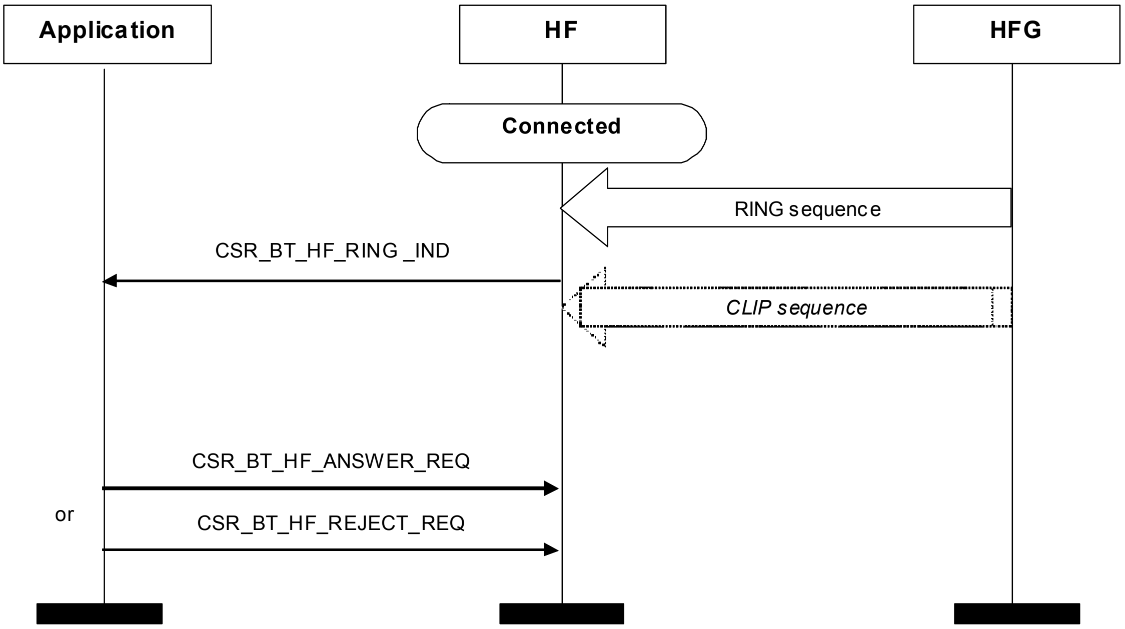


Figure 9: Incoming call received

If the HFG or HAG wants to make use of in-band ringing, audio can be connected before the ring signal is sent to the HF/HS manager. When audio is connected, the application will receive a CSR_BT_HF_AUDIO_IND command. If no in-band ring is used, audio can be connected upon answer from the application.

The application layer on the HFG side may also answer the call itself. The HFG application layer initiated answer is done by means of the +CIEV command, which will be received in a CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND.

3.3.2 Incoming Audio

Upon reception of an incoming audio request, the HF application is given the possibility to decide whether to accept or reject the audio connection. As a step in the procedure, the application layer must decide whether to accept the connection and if so what audio settings are acceptable and choose a PCM slot to map the audio to.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

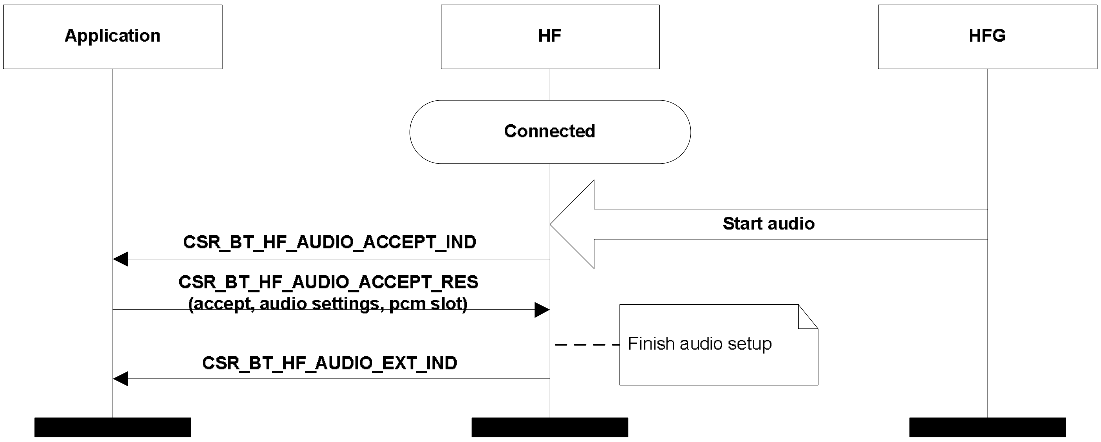

Application
HF
HFG
Connected
Start audio
CSR_BT_HF_AUDIO_ACCEPT_IND
CSR_BT_HF_AUDIO_ACCEPT_RES
(accept, audio settings, pcm slot)
Finish audio setup
CSR_BT_HF_AUDIO_EXT_IND

Figure 10: Incoming audio with dynamic PCM mapping

3.3.3 Outgoing Audio

The CSR_BT_HF_AUDIO signals will include valid PCM information.

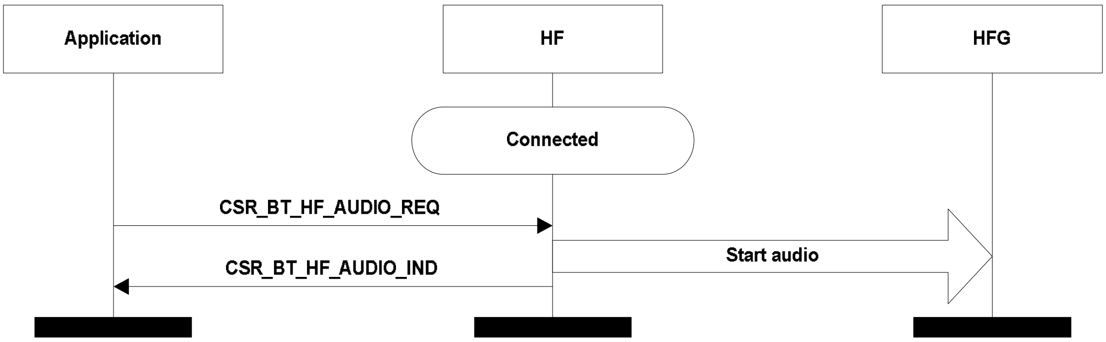


Figure 11: Outgoing audio initiation

3.3.4 Outgoing Call Set-Up

The HF can indicate the AG to start an outgoing call in three ways: commanding it to dial a specific number; to dial a number in a specific memory position; or to dial the last dialled number.

An application following the HF profile may start an outgoing call so by issuing the primitives CSR_BT_HF_DIAL_REQ, with the proper command as parameter as seen in Table 1.

Primitives	AT-command	Response
CSR_BT_HF_DIAL_REQ (CSR_BT_HF_DIAL_NUMBER)	ATDddd..dd	CSR_BT_HF_DIAL_CFM, +CIEV..., +CIEV...
CSR_BT_HF_DIAL_REQ (CSR_BT_HF_DIAL_MEMORY)	ATD>nnn	CSR_BT_HF_DIAL_CFM, +CIEV..., +CIEV...
CSR_BT_HF_DIAL_REQ (CSR_BT_HF_DIAL_REDIAL)	AT+BLDN	CSR_BT_HF_DIAL_CFM, +CIEV..., +CIEV...

Table 1: Outgoing call setup dial AT-commands

This interface will only be active during a hands free connection and not during a headset connection.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

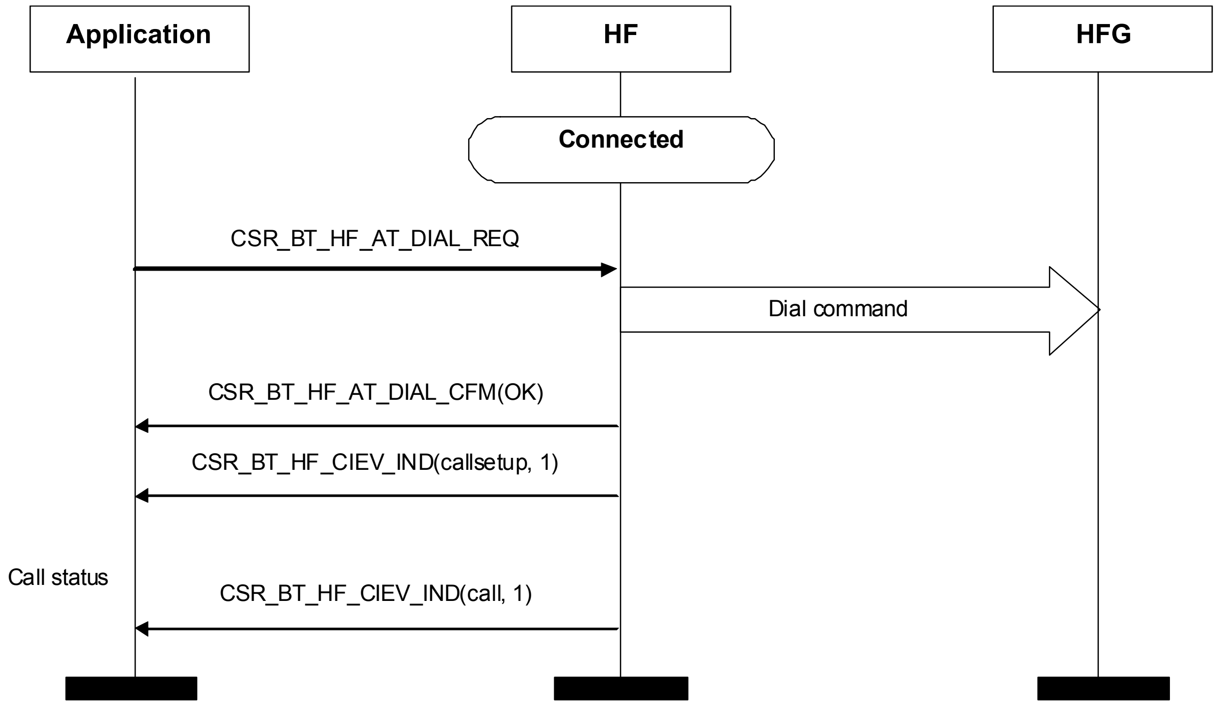

Application
HF
HFG
Connected
CSR_BT_HF_AT_DIAL_REQ
Dial command
CSR_BT_HF_AT_DIAL_CFM(OK)
CSR_BT_HF_CIEV_IND(callsetup, 1)
Call status
CSR_BT_HF_CIEV_IND(call, 1)

Figure 12: Outgoing call

During the call set-up process the application layer will continuously be updated from the HFG side with the current call status "+CIEV" command to the HF side. These commands will be forwarded to the application in the CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND signal. Further information can be found in [HF].

3.3.5 AT-Command Handling

The HF/HS is capable of interpreting the entire set of the AT-commands defined in the hands-free profile. It will parse the commands received and map them into different mail primitives as described in the next chapters. Besides the HF/HS profile allows the application to send and receive any AT command or response by using the CSR_BT_HF_AT_CMD_REQ/_IND primitives.

AT-command	HF signal primitive mapped to	Description
+BTRH	CSR_BT_HF_CALL_HANDLING_IND	Response & hold
+CHLD	NA (handled by the profile)	Hold capabilities information
+CIEV	CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND	Status change indication
+CLCC	CSR_BT_HF_GET_CURRENT_CALL_LIST_IND	Call list indication
+CME ERROR	This will be mapped to the corresponding signal depending on the operation being performed.	Error response from HFG
+CNUM	CSR_BT_HF_GET_SUBSCRIBER_NUMBER_INFORMATION_IND	Subscription number indication
+COPS	CSR_BT_HF_GET_CURRENT_OPERATOR_SELECTION_CFM	Operator name
RING	CSR_BT_HF_CALL_RINGING_IND	Incoming call
+VGM	CSR_BT_HF_MIC_GAIN_IND	Microphone gain

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

AT-command	HF signal primitive mapped to	Description
+VGS	CSR_BT_HF_SPEAKER_GAIN_IND	Speaker gain
+CLIP	CSR_BT_HF_CALL_NOTIFICATION_IND	CLIP information
+CCWA	CSR_BT_HF_CALL_WAITING_NOTIFICATION_IND	Call waiting
+BVRA	CSR_BT_HF_SET_VOICE_RECOGNITION_IND	Voice recognition
+BINP	CSR_BT_HF_BT_INPUT_CFM	Associate a phone number to a voice tag
+CIND	CSR_BT_HF_GET_ALL_STATUS_INDICATORS_CFM	Get the current status of all indicators supported
+BSIR	CSR_BT_HF_INBAND_RING_SETTING_CHANGED_IND	In-band ringing settings

Table 2: HF/HS upstream interpreted AT-commands

The HF/HS manager supports generic AT-command sending using the CSR_BT_HF_AT_CMD_REQ. The application layer must compile the AT string and include it in the signal. Besides the CSR_BT_HF_AT_CMD_REQ a number of specific commands are supplied in the HF interface; these are:

AT-command	Dedicated AT-command signal
ATA	CSR_BT_HF_CALL_ANSWER_REQ
AT+CHUP	CSR_BT_HF_CALL_END_REQ
AT+VGM	CSR_BT_HF_MIC_GAIN_STATUS_REQ
AT+VGS	CSR_BT_HF_SPEAKER_GAIN_STATUS_REQ
AT+CHLD or AT+BTRH	CSR_BT_HF_CALL_HANDLING_REQ
AT+COPS	CSR_BT_HF_GET_CURRENT_OPERATOR_SELECTION_REQ
ATD	CSR_BT_HF_DIAL_REQ
AT+CMEE	CSR_BT_HF_SET_EXTENDED_AG_ERROR_RESULT_CODE_REQ
AT+CLIP	CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION_REQ
AT+CCWA	CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION_REQ
AT+NREC	CSR_BT_HF_SET_ECHO_AND_NOISE_REQ
AT+BVRA	CSR_BT_HF_SET_VOICE_RECOGNITION_REQ
AT+BINP	CSR_BT_HF_BT_INPUT_REQ
AT+VTS	CSR_BT_HF_GENERATE_DTMF_REQ
AT+CIND?	CSR_BT_HF_GET_ALL_STATUS_INDICATORS_REQ

Table 3: HF/HS manager downstream specific AT- commands

The dedicated signals are converted directly to the corresponding AT-commands and sent to the HFG.

In order to comply with the HF/HS specifications, the HF profile maintains a queue of AT commands to be delivered to the HFG. This is because there must not be more than one AT command pending between the two devices at any given time. This may cause a certain delay in the AT command delivery process.

3.4 Connection Release

Connection release for both the HF and HS part of the HF/HS manager is handled in the same way. The application specifies which connection that should be released in the connectionType parameter. When the

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

application receives a disconnection indication the connectionType parameter indicates which connection that has been released.

3.4.1 Connection Release

A service level connection is released by sending a CSR_BT_HF_DISCONNECT_REQ to the HF/HS manager. A disconnect request is always confirmed with a CSR_BT_HF_DISCONNECT_CFM, with result codes defined in csr_bt_profiles.h.

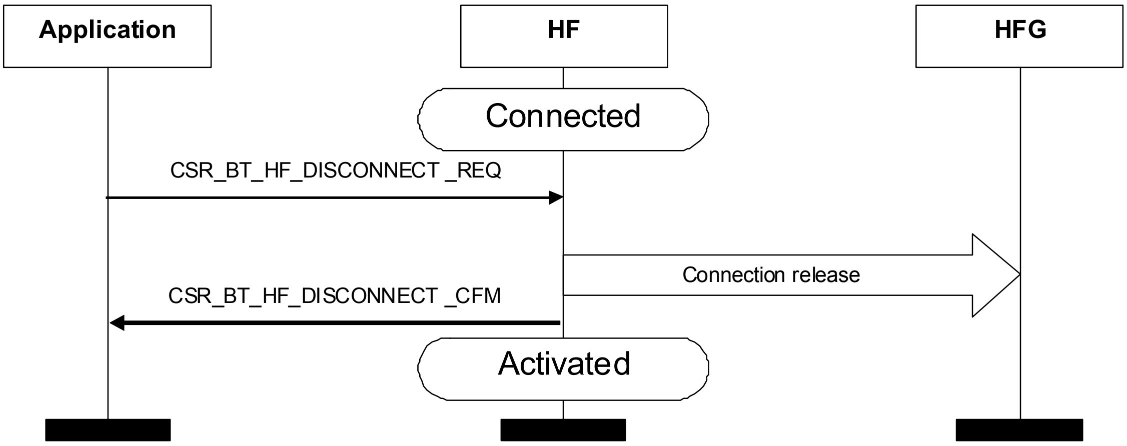


Figure 13: Connection release sequence

The application layer is not allowed to issue any new signals to the HF until the disconnect request is confirmed.

Please note that the disconnect command will release all resources and remove the Bluetooth® link between the HF and HFG or HS and HAG. To establish a new connection and setup a new call the procedures in 3.1, must be applied.

Due to race conditions it is possible that commands are received after the disconnect request has been sent. It is the responsibility of the application layer to handle these commands properly.

Note: The CSR_BT_HF_DISCONNECT_REQ signal will disconnect the RFCOMM link. According to the [HS] the Headset side should not disconnect the RFCOMM link. The HS should only use CSR_BT_HF_AUDIO_REQ(ON/OFF) to switch the audio on or off, and let the HAG disconnect the link if necessary. If all (e)SCO parameters are set to 0 in the audio on request, the best possible audio connection will be made. This is highly recommended. Only in extreme cases should tweaking the parameters from case to case manually be necessary. For changing the default settings globally, edit csr_bt_usr_config.h. For other recommendations regarding minimum parameter requirements, see [HF]

3.4.2 Peer Side Connection Release

The HFG or HAG at the remote end may at any time release the connection. A peer side connection release is indicated in the CSR_BT_HF_DISCONNECT_IND signal.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

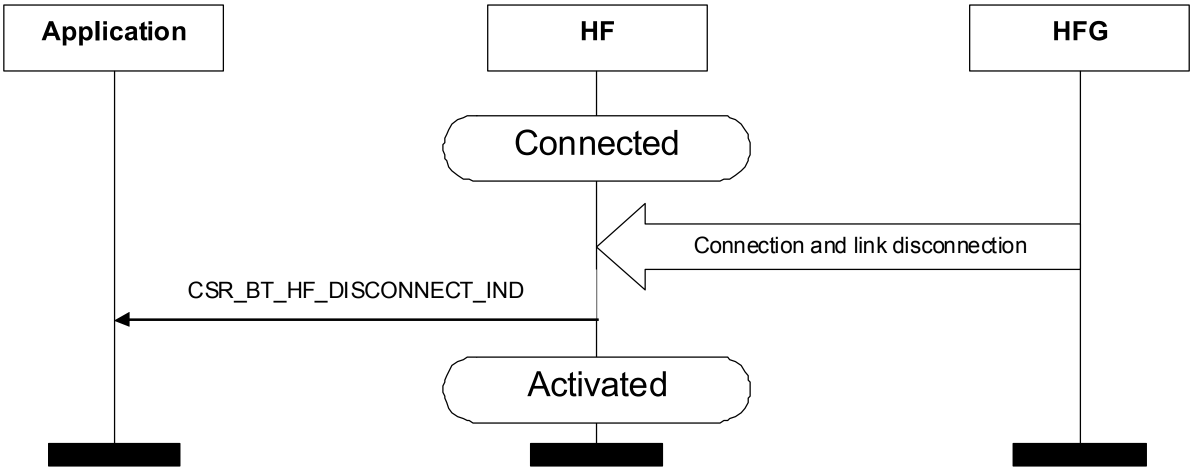


Figure 14: Peer side connection release sequence

When the CSR_BT_HF_DISCONNECT_IND is received no further signaling is possible until a new service level connection is established (see 3.1). After receiving the disconnect indication, the application layer cannot assume that the HFG part has maintained the state and history of the call.

3.4.3 Link Loss

It is possible that the physical link is closed due to an abnormal situation, e.g. radio interference or the devices getting out of coverage from each other. The HF manager will handle a link loss by informing the application layer via a CSR_BT_HF_DISCONNECT_IND with a result code indicating an abnormal situation. After receiving the disconnect indication, the application layer cannot assume that the HFG part has maintained the state and history of the call.

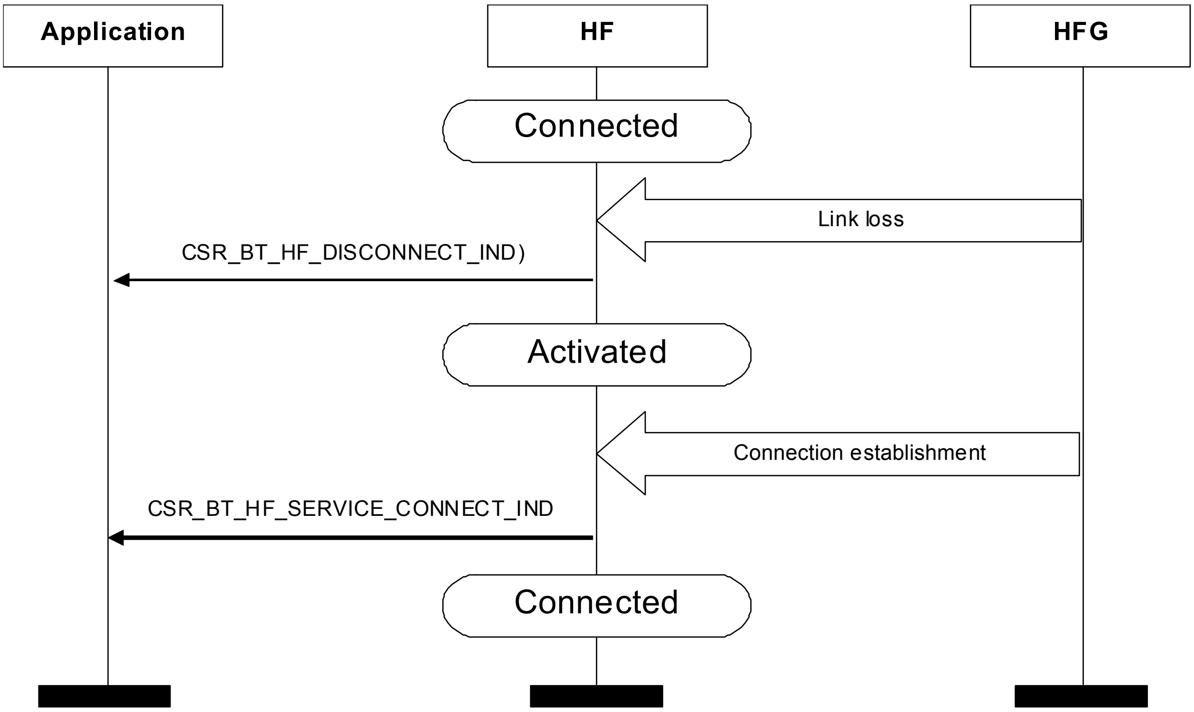


Figure 15: Abnormal connection release sequence due to link loss

The HF/HS profile manager automatically enters the activated state when disconnected, and will be ready to accept a reconnect from the HFG or the application may try and reconnect. If the abnormal disconnect occurs

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

during a call, it is the responsibility of the applications layer to reconnect audio and then the service level connection is established.

3.5 Deactivation

The application may deactivate the HF/HS manager at any time after the profile has been activated. This is done be sending the CSR_BT_HF_DEACTIVATE_REQ signal to the profile manager. This will make the HF/HS unregister both the HF and HS service record and cancel accepting connection on these server channels. After the profile has been deactivated, it will send the CSR_BT_HF_DEACTIVATE_CFM signal to the application. After this the application will not be able to send or receive any messages from the HF/HS profile until it has been activated again using the CSR_BT_HF_ACTIVATE_REQ..

3.6 Wide-Band Speech

The Wide-Band Speech (WBS) feature is an enhancement introduced in the HFP Bluetooth specification, version 1.6. To have access to this feature using the CSR Synergy BT implementation, the following conditions shall be met:

- The CSR chip used shall have DSP support
- The CSR Synergy BT code shall be compiled with the option “CSR_DSPM_ENABLE=1”
- The flag “CSR_BT_HF_SUPPORT_CODEC_NEGOTIATION” shall be included in the “supportedFeatures” field of the HF_ACTIVATE_REQ primitive.

Only if the remote device supports WBS too, will it be possible to establish a WBS audio connection between the two devices. The WBS feature mandates support for two audio formats: CVSD and mSBC, where the first is a narrow band format and the second is a wide band format.

At times, it may be necessary to disable the use of one of the formats, for example because there are not enough resources available. The CSR_BT_HF_UPDATE_SUPPORTED_CODEC_REQ primitive allows to enable or disable individual codec formats at run-time.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4 Hands-Free Primitives

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

To enable a fast integration process, library functions, which build each of the signals that are to be sent to the hands-free, are available. These are described in the library file csr_bt_hf_lib.h, which also defines the valid information elements that are to be included when sending a signal..

4.1 List of All Primitives

Primitives	Reference
CSR_BT_HF_ACTIVATE	See section 4.2
CSR_BT_HF_DEACTIVATE	See section 4.3
CSR_BT_HF_CONFIG_LOW_POWER	See section 4.4
CSR_BT_HF_CONFIG_AUDIO	See section 4.5
CSR_BT_HF_SERVICE_CONNECT	See section 4.6
CSR_BT_HF_CANCEL_CONNECT	See section 4.7
CSR_BT_HF_DISCONNECT_IND	See section 4.8
CSR_BT_HF_AUDIO_CONNECT	See section 4.9
CSR_BT_HF_AUDIO_ACCEPT_CONNECT	See section 4.10
CSR_BT_HF_AUDIO_DISCONNECT	See section 4.11
CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND	See section 4.12
CSR_BT_HF_GET_ALL_STATUS_INDICATORS	See section 4.13
CSR_BT_HF_GET_CURRENT_OPERATOR_SELECTION	See section 4.14
CSR_BT_HF_GET_SUBSCRIBER_NUMBER_INFORMATION	See section 4.15
CSR_BT_HF_GET_CURRENT_CALL_LIST	See section 4.16
CSR_BT_HF_SET_EXTENDED_AG_ERROR_RESULT_CODE	See section 4.17
CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION	See section 4.18
CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION	See section 4.19
CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND	See section 4.20
CSR_BT_HF_SET_STATUS_INDICATOR_UPDATE	See section 4.21
CSR_BT_HF_SET_ECHO_AND_NOISE	See section 4.22
CSR_BT_HF_SET_VOICE_RECOGNITION	See section 4.23
CSR_BT_HF_BT_INPUT	See section 4.24
CSR_BT_HF_GENERATE_DTMF	See section 4.25
CSR_BT_HF_SPEAKER_GAIN_STATUS	See section 4.26
CSR_BT_HF_MIC_GAIN_STATUS	See section 4.27
CSR_BT_HF_DIAL	See section 4.28
CSR_BT_HF_CALL_ANSWER	See section 4.29
CSR_BT_HF_CALL_END	See section 4.30
CSR_BT_HF_CALL_HANDLING	See section 4.31
CSR_BT_HF_AT_CMD	See section 4.32
CSR_BT_HF_SECURITY_IN	See section 4.33
CSR_BT_HF_SECURITY_OUT	See section 4.33

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Primitives	**Reference**
CSR_BT_HF_CALL_RINGING	See section 4.34
CSR_BT_HF_C2C_SF	See section 4.35
CSR_BT_HF_SET_C2C_AUDIO_CODEC	See section 4.36
CSR_BT_HF_GET_C2C_ADPCM_LOCAL_SUPPORTED	See section 4.37
CSR_BT_HF_SET_C2C_SAMPLE_RATE	See section 4.38
CSR_BT_HF_C2C_PWR	See section 4.39
CSR_BT_HF_C2C_BATT	See section 4.40
CSR_BT_HF_C2C_SMS	See section 4.41
CSR_BT_HF_C2C_TXT	See section 4.42
CSR_BT_HF_INBAND_RING_SETTING_CHANGED	See section 4.43
CSR_BT_HF_DEREGISTER_TIME	See section 4.44
CSR_BT_HF_INDICATION_ACTIVATION	See section 4.45
CSR_BT_HF_UPDATE_SUPPORTED_CODEC_REQ	See section 4.46
CSR_BT_HF_UPDATE_SUPPORTED_CODEC_CFM	See section 4.46

Table 4: List of all primitives

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.2 CSR_BT_HF_ACTIVATE

Parameters Primitives	type	phandle	maxHFConnections	maxHSConnections	maxSimultaneousConnections	supportedFeatures	hfConfig	atResponseTime	resultCode	resultSupplier
CSR_BT_HF_ACTIVATE_REQ	✓	✓	✓	✓	✓	✓	✓	✓		
CSR_BT_HF_ACTIVATE_CFM	✓								✓	✓

Table 5: CSR_BT_HF_ACTIVATE Primitives

Description

This signal is used for registering the service record with the supported features available in the HF profile. The CSR_BT_HF_ACTIVATE_REQ message can be issued at any time by the application. However some restrictions apply:

- If the activation request is issued while ongoing connections exist, the number of allowed simultaneous connections in the signal must be equal or greater than the actual number of ongoing connections.
- The maximum number of HF connection must be equal or greater than the number of existing HF connections
- The maximum number of HS connections must be equal or greater than the number of actual HS connections.

Parameters

type
The signal identity, CSR_BT_HF_ACTIVATE_REQ/ _CFM.

phandle
The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to phandle.

maxHFConnections
The maximum number of HFP simultaneous connections that shall be allowed. If set to 0, this means that the HF profile will not be active.

maxHSConnections
The maximum number of HSP simultaneous connections that shall be allowed. If set to 0, this means that the HS profile will not be active.

maxSimultaneousConnections
The maximum number of simultaneous connections allowed

supportedFeatures
This parameter is a bitmap representing the supported features in the Hands Free profile. The values are defined in the csr_bt_hf.h file.

hfConfig
This parameter is used for enabling or disabling some features in the HS/HF manager. This shall be a value consisting of a bitmask enabling and/or disabling different features:

- CSR_BT_HF_CNF_ENABLE_LOW_POWER_STATUS = 0x00000001
  Send CSR_BT_HF_STATUS_LOW_POWER_IND when changes in LP-mode happen
- CSR_BT_HF_CNF_DISABLE_LOW_POWER = 0x00000002
  If set to '0' the HF device will request low power mode instead of waiting

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

for the remote device to do it. This is default behavior.

- CSR_BT_HF_CNF_DISABLE_REMOTE_VOLUME_CONTROL = 0x00000004
If set to “1”, the HSP will not support remote volume control. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_CMER_UNDER_SLC_ESTABLISHMENT = 0x00000008
If “1” then the HF will not ask for indicators from the HFG. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_CHLD_UNDER_SLC_ESTABLISHMENT = 0x00000010
If “1” the HF will not ask for CHLD capabilities from the HFG. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_CLIP_UNDER_SLC_ESTABLISHMENT = 0x00000020
If “1” the calling line identification feature will not be automatically enabled. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_CCWA_UNDER_SLC_ESTABLISHMENT = 0x00000040
If “1” ” the call waiting feature will not be automatically enabled. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_CMEE_UNDER_SLC_ESTABLISHMENT = 0x00000080
If “1” ” the extended error feature will not be automatically enabled. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_OUT_SDP_SEARCH = 0x00000100
If ‘1’ the HF will only perform remote server channel search during SDP search for outgoing SLC: neither remote version number, nor service name, nor network nor supported features will be asked for. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_OUT_SERVICE_NAME_SEARCH =0x00000200
If ‘1’ and CSR_BT_HF_DISABLE_OUT_SDP_SEARCH is ‘0’, the remote service name will not be asked for during SDP for outgoing SLC. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_OUT_NETWORK_SEARCH =0x00000400
If ‘1’ and CSR_BT_HF_DISABLE_OUT_SDP_SEARCH is ‘0’, the remote “network” attribute will not be asked for during SDP for outgoing SLC. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_OUT_SUP_FEATURES_SEARCH =0x00000800
If ‘1’ and CSR_BT_HF_DISABLE_OUT_SDP_SEARCH is ‘0’, the remote supported features attribute will not be asked for during SDP for outgoing SLC. Default is ‘0’

- CSR_BT_HF_CNF_DISABLE_INC_SDP_SEARCH =0x00001000
If ‘1’ no service search operation will be performed at all for incoming SLC. Default is ‘0’. If ‘0’, at least the remote version number is retrieved.

- CSR_BT_HF_CNF_DISABLE_INC_SERVICE_NAME_SEARCH =0x00002000
If ‘1’ and CSR_BT_HF_DISABLE_INC_SDP_SEARCH is ‘0’ the remote service name attribute will not be retrieved during incoming SLC establishment.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

- CSR_BT_HF_CNF_DISABLE_INC_NETWORK_SEARCH =0x00004000
  If ‘1’ and CSR_BT_HF_DISABLE_INC_SDP_SEARCH is ‘0’ the remote network attribute will not be retrieved during incoming SLC establishment.

- CSR_BT_HF_CNF_DISABLE_INC_SUP_FEATURES_SEARCH =0x00008000
  If ‘1’ and CSR_BT_HF_DISABLE_INC_SDP_SEARCH is ‘0’ the remote supported features attribute will not be retrieved during incoming SLC establishment.

- CSR_BT_HF_AT_MODE_TRANSPARENT =0x00010000
  If ’1’ then the HF profile will not handle any recieved AT commands, instead they will be passed directly to the application.

atResponseTime — Time in seconds that the device shall wait for response on an AT command sent. Default value is 2; minimum value is 2 (i.e. values lower than 2 in the activate signal will be ignored and the default value will be used).

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

```
void CsrBtHfActivateReqSend(  CsrSchedQid thePhandle,
                              CsrUint8   maxNumberOfHfConnections,
                              CsrUint8   maxNumberOfHsConnections,
                              CsrUint8   maxSimultaneousConnections,
                              CsrUint32  theSupportedFeatures,
                              CsrUint32  theHsConfig)
```

Phandle — The same as in the table above.

`maxNumberOfHfConnections` — The same as “maxHFConnections” above

`maxNumberOfHsConnections` — The same as “maxHSConnections” above

maxSimultaneousConnections — The same as “maxSimultaneousConnections” above

supportedFeatures — The same as in the table above.

`theHsConfig` — The same as “hsConfig” in the table above.

Example

Here is an example of how to send the signal using the library function.

```
CsrBtHfActivateReqSend(APP_TASK, 2, 0, 2,
CSR_BT_HF_SUPPORT_CLI_PRESENTATION_CAPABILITY |
   CSR_BT_HF_SUPPORT_CALL_WAITING_THREE_WAY_CALLING|CSR_BT_HF_SUPPORT_REMOTE_VOLUME
_CONTROL| CSR_BT_HF_SUPPORT_CODEC_NEGOTIATION,
CSR_BT_HF_CNF_DISABLE_AUTOMATIC_CCWA_ACTIVATION, 0).
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.3 CSR_BT_HF_DEACTIVATE

Parameters Primitives	type	resultCode	resultSupplier
CSR_BT_HF_DEACTIVATE_REQ	✓		
CSR_BT_HF_DEACTIVATE_CFM	✓	✓	✓

Table 6: CSR_BT_HF_DEACTIVATE Primitives

Description

This signal will deactivate the HS/HF profile.

Parameters

Type	The signal identity, CSR_BT_HF_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

```
void CsrBtHfDeactivateReqSend(void)
```

Example

Here is an example of how to send the signal using the library function.

```
CsrBtHfDeactivateReqSend()
```

4.4 CSR_BT_HF_CONFIG_LOW_POWER

Primitives \ Parameters	type	connectionId	mask	currentMode	oldMode	wantedMode	remoteReq	resultCode	resultSupplier
CSR_BT_HF_CONFIG_LOW_POWER_REQ	✓	✓	✓						
CSR_BT_HF_STATUS_LOW_POWER_IND	✓	✓		✓	✓	✓	✓	✓	✓
CSR_BT_HF_CONFIG_LOW_POWER_CFM	✓								

Table 7: CSR_BT_HF_CONFIG_LOW_POWER Primitives

Description

The CSR_BT_HF_CONFIG_LOW_POWER_REQ signal will determine how the profile shall handle the low power mode. When the profile manager handles the request it will issue the CSR_BT_HF_CONFIG_LOW_POWER_CFM message to the application. Whenever there is a change in the low power status of the connection, the application will receive the CSR_BT_HF_STATUS_LOW_POWER_IND.

Parameters

Type The signal identity, CSR_BT_HF_CONFIG_LOW_POWER_REQ /IND/CFM.

connectionId Connection identifier

Mask Low power configuration bitmask. Can take the following values or a combination of them:

- CSR_BT_HF_PWR_DISCONNECT_ON_NO_LP, 0x00000001.
  If enabled, disconnect the link if LP mode could not be entered

- CSR_BT_HF_PWR_DISABLE_PARK_ACP, 0x00000002.
  If enabled, do not accept park mode from the remote device

- CSR_BT_HF_PWR_DISABLE_SNIFF_REQ, 0x00000004.
  If set, disable local sniff and sniff sub-rate requests

- CSR_BT_HF_PWR_DISABLE_SNIFF_ACP, 0x00000008.
  If set, disable remote sniff and sniff sub-rate request

`currentMode` current low power mode

`OldMode` old low power mode

`wantedMode` wanted low power mode

`remoteReq` If TRUE, the mode change was requested by remote peer

`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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

values can be found in csr_bt_result.h

Library Function

```
void CsrBtHfConfigLowPowerReqSend (CsrBtHfConnectionId   connectionId,
                                   HfConfigLowPowerMask_t  mask)
```

Example

Here is an example of how to send the signal using the library function.

```
CsrBtHfConfigLowPowerReqSend (conId, CSR_BT_HF_PWR_DISCONNECT_ON_NO_LP);
```

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.5 CSR_BT_HF_CONFIG_AUDIO

Primitives \ Parameters	type	connectionId	audioType	audioSettingLen	audioSetting
CSR_BT_HF_CONFIG_AUDIO_REQ	✓	✓	✓	✓	✓
CSR_BT_HF_CONFIG_AUDIO_CFM	✓				

Table 8: CSR_BT_HF_CONFIG_AUDIO Primitives

Description

The HF profile ensures compliance with the Bluetooth specification regarding audio connections. However, the application has the possibility to establish or accept incoming audio connections that use other settings than the ones specified in the Bluetooth spec by means of the primitive CSR_BT_HF_CONFIG_AUDIO_REQ. In the case of outgoing connections, if the connection establishment fails, the HF profile will fall back to use the settings specified in Ref. [HF].

Parameters

type	The signal identity, CSR_BT_HF_CONFIG_AUDIO_REQ/CFM.
connectionId	The identification number of the connection.
AudioType	The type of data contained in the pointer below. The different types available are defined in csr_bt_hf_prim.h.
audioSettingLen	Length of the data in the pointer below in bytes, in a 16-bit value
*audioSetting	Pointer to audio configuration data of the type specified in the field audioType above.

Library Function

```
    void CsrBtHfAudioConfigReqSend(CsrBtHfConnectionId  connectionId,
                                   CsrBtHfAudioType  audioType,
                                   CsrUint8          *audioSetting,
                                   CsrUint16      audioSettingLength)
```

connectionId	The same as in the table above.
`audioType`	The same as in the table above.
`*audioSetting`	The same as in the table above.
audioSettingLen	The same as in the table above.

Example

Here is an example of how to send the request signal using the library function.

```
   CsrBtHfAudioConfigReqSend(connectionId,
CSR_BT_HF_AUDIO_PREFERED_SAMPLE_RATE_SETUP,
                       *sampleRateConfig,
sizeof(CsrBtHfAudioPreferedSampleRateType));
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

```
If the Wide-Band speech feature is supported, the code needs to be compiled with
the "CSR_USE_DSPM" flag enabled. In that case, it is possible to configure how to
route the audio and to configure the audio routing itself before the audio is
connected. Some audio configuration parameters can also be set AFTER the audio
connection has been established. For instance, the audio gain. This is only
relevant if the chip uses DSP. For more information about this please refer to the
chip documentation.
```

4.6 CSR_BT_HF_SERVICE_CONNECT

Primitives \ Parameters	type	deviceAddr	supportedFeatures	network	serviceName	connectionType	connectionId	indicatorSupported	indicatorValue	chldString	remoteversion	resultCode	resultSupplier	btConnId
CSR_BT_HF_SERVICE_CONNECT_REQ	✓	✓				✓								
CSR_BT_HF_SERVICE_CONNECT_IND	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CSR_BT_HF_SERVICE_CONNECT_CFM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 9: CSR_BT_HF_SERVICE_CONNECT Primitives

Description

Setup and establish a connection between a HFG and a HF or a HS and a HAG specified by the Bluetooth® address. Please note that before a complete service level connection is established a number of AT-commands must be exchanged. An example of a string given in the "chldString" field is:

(0,1,1x,2,2x,3,4)

This means that the HFG that has been connected supports the following commands [GSM02.30]:

0 Releases all held calls or sets User Determined User Busy (UDUB) for a waiting call.
1 Releases all active calls (if any exist) and accepts the other (held or waiting) call.
1x Releases a specific active call X.
2 Places all active calls (if any exist) on hold and accepts the other (held or waiting) call.
2x Places all active calls on hold except call X with which communication shall be supported.
3 Adds a held call to the conversation.
4 Connects the two calls and disconnects the subscriber from both calls

The application sends a CSR_BT_HF_SERVICE_CONNECT_REQ in order to start the establishment of a connection. When it is established, the profile sends the CSR_BT_HF_SERVICE_CONNECT_CFM back to the application. In the case of an incoming connection, the profile manager will send a CSR_BT_HF_SERVICE_CONNECT_IND instead.

Parameters

type The signal identity, CSR_BT_HF_SERVICE_CONNECT_REQ/IND/CFM.

deviceAddr The Bluetooth® device address to which a connection must be established. The deviceAddr is of type BD_ADDR_T which again is defined as:

```
typedef struct
{
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

```
    CsrUint24 lap;   /* Lower Address Part 00..23 */
    CsrUint8  uap;   /* upper Address Part 24..31 */
    CsrUint16 nap;   /* Non-significant    32..47 */
} BD_ADDR_T;
```

supportedFeatures

If a connection was established to a HFG this parameter will describe the supported features of the remote device represented as a bitmap. The values are defined in the csr_bt_hf.h file (`CSR_BT_HFG_SUPPORT_xxx`). If a HAG is connected this parameter will be 0.

Network

If a connection is established to a HFG this parameter will indicate the supported network of the HFG. This parameter, which is of type CsrUint8, will be 0 if connected to a HAG.

serviceName

Text defined by the HF service record.
The contents will be a string that is 0-terminated.

connectionType

Indicates which type of connection that has been established. A connection to a HFG will be indicated with the value CSR_BT_HF_CONNECTION_HF and a connection to a HAG will be indicated with a CSR_BT_HF_CONNECTION_HS. If the connect request fails an error code will be returned in the result parameter, and the connectionType will be CSR_BT_HF_CONNECTION_UNKNOWN. The value CSR_BT_HF_CONNECTION_UNKNOWN shall be used if both HF and HS connections are allowed to be established.The values used in the connectionType parameter is defined in csr_bt_hf_prim.h

connectionId

Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF (CSR_BT_HF_CONNECTION_ALL) is used for denoting "ALL_CONNECTIONS"

indicatorSupported

Pointer to a 0-terminated comma separated string with the indicators supported by the HFG and their range.

indicatorValue

Pointer to a 0-terminated comma separated string with the current status of the indicators supported by the HFG.

ChldString

String that describes the call hold and multiparty services available in the HFG as shown above.

remoteVersion

remote device's supported version of the spec

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

btConnId

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

Library Function

```
    void CsrBtHfServiceConnectReqSend(deviceAddr_t deviceAddr, ConType_t
                                      connectionType)
```

deviceAddr

The same as in the table above.

connectionType

The same as in the table above

Example

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Here is an example of how to send the request signal using the library function.

```
CsrBtHfServiceConnectReqSend(deviceAddr, connectionType)
```

4.7 CSR_BT_HF_CANCEL_CONNECT

Parameters / Primitives	type	deviceAddr
CSR_BT_HF_CANCEL_CONNECT_REQ	✓	✓

Table 10: CSR_BT_HF_CANCEL_CONNECT Primitives

Description

The CSR_BT_HF_CANCEL_CONNECT_REQ signal is used for stopping a connection operation previously started from the application, before it has actually been established. When the connection establishment is successfully stopped, the profile will return the CSR_BT_HF_SERVICE_CONNECT_IND signal with a proper result code. Please note that in some situations where the CSR_BT_HF_CANCEL_CONNECT_REQ and the CSR_BT_HF_SERVICE_CONNECT_IND signals cross, the connection is actually established, and the application will have to disconnect it.

Parameters

type — The signal identity, CSR_BT_HF_CANCEL_CONNECT_REQ.

deviceAddr — The Bluetooth® device address to which a connection must be established. The deviceAddr is of type BD_ADDR_T which again is defined as:

```
typedef struct
{
    CsrUint24 lap;   /* Lower Address Part 00..23 */
    CsrUint8  uap;   /* upper Address Part 24..31 */
    CsrUint16 nap;   /* Non-significant    32..47 */
} BD_ADDR_T;
```

Library Function

```
void CsrBtHfCancelConnectReqSend(deviceAddr_t deviceAddr)
```

`deviceAddr` — The same as in the table above.

Example

Here is an example of how to send the request signal using the library function. This example requests disconnection of all existing connections.

```
CsrBtHfCancelConnectReqSend(devAddress);
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.8 CSR_BT_HF_DISCONNECT

Parameters / Primitives	type	connectionId	reasonCode	reasonSupplier
CSR_BT_HF_DISCONNECT_REQ	✓	✓		
CSR_BT_HF_DISCONNECT_IND	✓	✓	✓	✓
CSR_BT_HF_DISCONNECT_CFM	✓	✓	✓	✓

Table 11: CSR_BT_HF_DISCONNECT Primitives

Description

The CSR_BT_HF_DISCONNECT_REQ signal is used for disconnecting a HF or HS connection, which is indicated in the connectionId parameter. When the connection is closed the profile will return the CSR_BT_HF_DISCONNECT_CFM signal with a result code and a connection Id parameter. If the connection is disconnected without the application asking for it, the message used will be the CSR_BT_HF_DISCONNECT_IND instead.

Parameters

type The signal identity, CSR_BT_HF_DISCONNECT_REQ/IND/CFM.

connectionId Unique identifier for the connection that was disconnected. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF (CSR_BT_HF_CONNECTION_ALL) is used for denoting “ALL_CONNECTIONS”

reasonCode The reason code of the operation. Possible values depend on the value of reasonSupplier. If e.g. the reasonSupplier == CSR_BT_SUPPLIER_CM then the possible reason codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h files are regarded as reserved and the application should consider them as errors.

reasonSupplier This parameter specifies the supplier of the reason given in reasonCode. Possible values can be found in csr_bt_result.h

Library Function

```
    void CsrBtHfDisconnectReqSend(CsrBtHfConnectionId connectionId)
```

connectionId The same as in the table above.

Example

Here is an example of how to send the request signal using the library function. This example requests disconnection of all existing connections.

```
   CsrBtHfDisconnectReqSend(0xFFFFFFFF)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.9 CSR_BT_HF_AUDIO_CONNECT

Parameters Primitives	type	connectionId	audioParametersLength	*audioParameters	scoHandle	pcmSlot	pcmRealloc	linkType	txInterval	weSco	rxPacketLength	txPacketLength	airMode	codecUsed	sampleRate	resultCode	resultSupplier
CSR_BT_HF_AUDIO_CONNECT_REQ	✓	✓	✓	✓		✓	✓										
CSR_BT_HF_AUDIO_CONNECT_IND	✓	✓			✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
CSR_BT_HF_AUDIO_CONNECT_CFM	✓	✓			✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 12: CSR_BT_HF_AUDIO_CONNECT Primitives

Description

Request for audio establishment. The audio must be controlled by the application due to the interface for external audio, e.g. GSM.

If the application has issued the CSR_BT_HF_CONFIG_AUDIO_REQ primitive to set the audio quality prior to the CSR_BT_HF_AUDIO_CONNECT_REQ message, then the settings given in the configuration message will be used by the HF. Thus, the application decides the audio quality. Please note that a higher quality requires more bandwidth and more power consumption from the battery. If the CSR_BT_HF_CONFIG_AUDIO_REQ has not been issued, it is up to the profile to decide the best quality available at the given time.

Parameters

Type

The signal identity, CSR_BT_HF_AUDIO_CONNECT_REQ/IND.

connectionId

The connection to request audio on.

audioParametersLength

Number of entries in the audioParameters pointer. NB: This parameter is currently reserved for future usage and should hence be set as 0.

*audioParameters

Specifies which SCO/eSCO parameters to use in the connection establishment. If NULL the default Audio parameters from csr_bt_usr_config.h or CSR_BT_HF_CONFIG_AUDIO_REQ are used.

This pointer is of the type CsrBtHfgAudioLinkParameterListConfig and is described below.

NB: This parameter is currently reserved for future usage and should hence be set as NULL.

scoHandle

Sco handle if routed internally

pcmSlot

Pcm slot to use

pcmRealloc

If TRUE, then reallocate automatically if pcm-slot chosen is in use

LinkType

Link type used in the audio connection (0x00 = SCO, 0x02 = eSCO)

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

txInterval	Transmission interval in baseband slots (Set to zero for SCO links)
WeSco	Retransmission window in baseband slots (Set to 0 for SCO links)
rxPacketLength	RX packet length in bytes for eSCO connection (Set to 0 for SCO links)
txPacketLength	TX packet length in bytes for eSCO connection (Set to 0 for SCO links)
AirMode	The air mode: 0x00 = my-law; 0x01 = A-law; 0x02 = CVSD; 0x03 = Transparent data
codecUsed	Codec used for the audio connection established
sampleRate	Sample rate used for the audio connection established
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

```
    void CsrBtHfAudioConnectReqSend(CsrBtHfConnectionId connectionId, CsrUint8
                                  audioParametersLength,
                                  CsrBtHfAudioLinkParameterListConfig
                                  audioParameters, CsrUint8 pcmSlot,
                                    CsrBool pcmRealloc)
```

Example

Here is an example of how to send the request signal using the library function.

```
   CsrBtHfAudioConnectReqSend(conId, 0, NULL, 0x01, FALSE)
```

Examples for generic (e)SCO packet bit patterns:

The packet type support for negotiating the audio quality is specified in a 2 octet bit pattern as shown here.

						3EV5	**2EV5**	**3EV3**	**2EV3**	**EV5**	**EV4**	**EV3**	**HV3**	**HV2**	**HV1**

Note that the EDR eSCO packets (2EV3, 3EV3, 2EV5 and 3EV5) have their enable bits inverted. The grayed out section are bits reserved for future use and their setting is irrelevant for the application interfacing the hands-free profile manager. In the examples below the reserved bits are set to 1.

Enable all:
Should all packets be available for while negotiating the quality setting for the audio link between devices, the HV bits and EV bits should be set. The fact that the EDR eSCO packets have their enable bits inverted means that they should not be set to enable them.

Set HV1 + HV2 + HV3 + EV3 + EV4 + EV5 and not 2EV3 + 3EV3 + 2EV5 + 3EV5
1111110000111111 = 0xFC3F

For setting this bit pattern, the following packet definitions from hci.h should be combined:

```
HCI_ESCO_PKT_HV1 | HCI_ESCO_PKT_HV2 | HCI_ESCO_PKT_HV3 | HCI_ESCO_PKT_EV3 |
HCI_ESCO_PKT_EV4 | HCI_ESCO_PKT_EV5
```

Specifying only SCO:
Should only SCO packets be available, the HV and the EDR eSCO bits should be set.

Set HV1 + HV2 + HV3 and 2EV3 + 3EV3 + 2EV5 + 3EV5

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

1111111111000111   = 0xFFC7

For setting this bit pattern, the following packet definitions from hci.h should be combined:

```
HCI_ESCO_PKT_HV1 | HCI_ESCO_PKT_HV2 | HCI_ESCO_PKT_HV3 | HCI_ALL_MR_ESCO
```

Specifying only eSCO:
Should only the eSCO packet types be available, the EV bits should be set and the EDR eSCO bits should not.

Set EV3 + EV4 + EV5 and not 2EV3 + 3EV3 + 2EV5 + 3EV5
1111110000111000   = 0xFC38

For setting this bit pattern, the following packet definitions from hci.h should be combined:

HCI_ESCO_PKT_HV1 | HCI_ESCO_PKT_HV2 | HCI_ESCO_PKT_HV3

Description of the type CsrBtHfAudioLinkParameterListConfig;

packetType	Specifies which SCO/eSCo packet types to use
txBandwitdh	Specifies the maximum Transmission bandwidth to use
rxBandwitdth	Specifies the maximum Receive bandwidth to use
maxLatency	Specifies the maximum Latency to use.
voiceSettings	Specifies the voice coding used in the transmission. Eg. Csvd or Transparent.
reTxtEffort	Specifies the Retransmission setting to use.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.10 CSR_BT_HF_AUDIO_ACCEPT_CONNECT

Parameters / Primitives	type	connectionId	linkType	acceptResponse	*acceptParameters	acceptParamteresLength	pcmSlot	pcmReassign
CSR_BT_HF_AUDIO_ACCEPT_CONNECT_IND	✓	✓	✓					
CSR_BT_HF_AUDIO_ACCEPT_CONNECT_RES	✓			✓	✓	✓	✓	✓

Table 13: CSR_BT_HF_AUDIO_ACCEPT_CONNECT Primitives

Description

The CSR_BT_HF_AUDIO_ACCEPT_CONNECT_IND is sent to the application layer during the initiation of a remotely initiated audio connection. The application layer must then send a CSR_BT_HF_AUDIO_ACCEPT_CONNECT_RES to the HF profile in order to decide whether to accept an incoming audio connection, and if so what settings to accept and which PCM port the incoming connection must be mapped to.

Note that the dynamic PCM port selection will only work if PSKEY_HOSTIO_MAP_SCO_PCM is set to FALSE.

Note also that the contents of the "acceptParameters" may be overruled with default values in the HF profile in case an eSCO/SCO connection exists beforehand, as the settings already used for the existing connection might have an influence on what settings are possible in the new one.

Parameters

Type The signal identity, CSR_BT_HF_AUDIO_ACCEPT_CONNECT_IND/RES.

connectionId The identification number of the connection.

LinkType Specifies SCO/eSCO

acceptResponse The HCI response code from profile can be one of the following: HCI_SUCCESS,HCI_ERROR_REJ_BY_REMOTE_NO_RES, HCI_ERROR_REJ_BY_REMOTE_PERS Note: If this is not HCI_SUCCESS then the incoming SCO/eSCO connection will be rejected

acceptParameters Specifies which SCO/eSCO parameters to accept. If NULL the default ACCEPT SCO parameters from csr_bt_usr_config.h or CSR_BT_HF_CONFIG_AUDIO_REQ are used.

This pointer is of the type CsrBtHfgAudioIncomingAcceptParameters and is described below.

acceptParametersLength Shall be 1 if acceptParameters != NULL, otherwise 0

pcmSlot Map this audio connection to the specified PCM slot. Range is 0-4, where 0 means that audio will be routed through CSR Synergy Bluetooth, and 1-4 corresponds to PCM slot 1-4 on the BlueCore chip. If the parameter is set to

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

CSR_BT_PCM_DONT_CARE defined in csr_bt_profiles.h, the first available PCM slot will be assigned.

pcmRealloc — Automtically assign another pcm-slot if pcmSlot given in this response is already in use. The resulting pcmSlot will be informed in the CSR_BT_HF_AUDIO_IND.

Library Function

```
    void CsrBtHfAudioAcceptConnectResSend( CsrBtHfConnectionId    connectionId ,
                                 hci_error_t acceptResponse,
                                 CsrBtHfAudioIncomingAcceptParameters
                                 *acceptParameters, CsrUint8
                                 acceptParametersLength
                                 CsrUint8  pcmSlot,
                                 CsrBool  pcmRealloc)
```

Example

Here is an example of how to send the request signal using the library function.

```
   CsrBtHfAudioAcceptConnectResSend(indPrim->connectionId, HCI_SUCCESS, NULL,
1,TRUE)
```

Description of the type CsrBtHfgAudioIncomingAcceptParameters:

packetTypes — Specifies which SCO/eSCo packet types to accept

txBandwitdh — Specifies the maximum Transmission bandwidth to accept

rxBandwitdth — Specifies the maximum Receive bandwidth to accept

maxLatency — Specifies the maximum Latency to accept

contentFormat — Specifies which SCO/eSCO content format to accept

reTxtEffort — Specifies the Retransmission setting to accept.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.11 CSR_BT_HF_AUDIO_DISCONNECT

Primitives \ Parameters	type	connectionId	scoHandle	resultCode	resultSupplier	reasonCode	reasonSupplier
CSR_BT_HF_AUDIO_DISCONNECT_REQ	✓	✓	✓				
CSR_BT_HF_AUDIO_DISCONNECT_CFM	✓	✓	✓	✓	✓		
CSR_BT_HF_AUDIO_DISCONNECT_IND	✓	✓	✓			✓	✓

Table 14: CSR_BT_HF_AUDIO_DISCONNECT Primitives

Description

The application can release an audio connection by means of the CSR_BT_HF_AUDIO_DICONNECT_REQ primitive. When the connection is actually released, the profile will send a confirm message back to the application. However, if the audio connection is released by the remote device, or due to a link loss, the profile will send a CSR_BT_HF_DISCONNECT_IND to the application instead, to inform it about the release of the connection.

Parameters

Type The signal identity, CSR_BT_HF_AUDIO_DICONNECT_REQ /IND/CFM

connectionId The identification number of the connection.

scoHandle Sco handle if routed internally.NB: If this parameter is set as 0xFFFF in the CSR_BT_HF_AUDIO_DISCONNECT_REQ the profile will disconnect all established audio connections to the associated connectionId and it will send a CSR_BT_HF_AUDIO_DISCONNECT_CFM for each audio connection that is disconnected. Otherwise, it should be set to the right scoHandle as received in a respective CSR_BT_HF_AUDIO_CONNECT_IND/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

reasonCode The reason code of the operation. Possible values depend on the value of reasonSupplier. If e.g. the reasonSupplier == CSR_BT_SUPPLIER_CM then the possible reason codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h files are regarded as reserved and the application should consider them as errors.

reasonSupplier This parameter specifies the supplier of the reason given in reasonCode. Possible values can be found in csr_bt_result.h

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfAudioDisconnectReqSend(indPrim->connectionId, 0xFFFF)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

4.12 CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND

Parameters Primitives	type	connectionId	index	value	*name
CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND	✓	✓	✓	✓	✓

Table 15: CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND Primitives

Description

Whenever there is a change in any of the indicators supported by the HFG device, such as incoming call, outgoing call process, battery status, roaming or signal status, etc. the HFG will send the “+CIEV=” response code. The HF profile maps this response code into the signal CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND and sends it to the application.

Parameters

Type The signal identity, CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND

connectionId Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

indicatorSupported Pointer to a string with the list of indicators supported by the HFG

Index The index value into the CIND string

value Value of the status indicator

*name Name of the status indicator

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.13 CSR_BT_HF_GET_ALL_STATUS_INDICATORS

Parameters / Primitives	type	connectionId	indicatorSupported	indicatorValue	cmeeResultCode
CSR_BT_HF_GET_ALL_STATUS_INDICATORS_REQ	✓	✓			
CSR_BT_HF_GET_ALL_STATUS_INDICATORS_CFM	✓	✓	✓	✓	✓

Table 16: CSR_BT_HF_GET_ALL_STATUS_INDICATORS Primitives

Description

Used for requesting an update on all the status indicators supported by the remote device.

Parameters

type The signal identity, CSR_BT_HF_GET_ALL_STATUS_INDICATOR_REQ/CFM

connectionId Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

indicatorSupported Pointer to a string with the list of indicators supported by the HFG

indicatorValue Pointer to a string with the actual values of the indicators supported by the HFG

cmeeResultCode The result of the query, e.g. CSR_BT_CME_SUCCESS. The specific values are defined in the csr_bt_hf.h file.

Library Function

```
    void CsrBtHfGetAllStatusIndicatorsReqSend(CsrBtHfConnectionId   connectionId)
```

`connectionId` The same as in the table above.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfGetAllStatusIndicatorsReqSend(conId)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.14 CSR_BT_HF_GET_CURRENT_OPERATOR_SELECTION

Parameters / Primitives	type	connectionId	mode	format	forceResendingFormat	copsString	cmeeResultCode
CSR_BT_HF_GET_CURRENT_OPERATOR_SELECTION_REQ	✓	✓	✓	✓	✓		
CSR_BT_HF_GET_CURRENT_OPERATOR_SELECTION_CFM	✓	✓				✓	✓

Table 17: CSR_BT_HF_GET_CURRENT_OPERATOR_SELECTION Primitives

Description

This is the request for operator network name. Sending the CSR_BT_HF_COPS_REQ to the HF will cause the HF to set up the HFG to answer in to “long format alphanumeric” and afterwards it will ask for the network operator name. Setting up the format is only necessary once for each service level connection; however the application can force the profile to set the format by using the field “forceResendingFormat”. The completion of the process will be informed to the application by sending the CSR_BT_HF_COPS_CFM.

Parameters

Type — The signal identity, CSR_BT_HF_COPS_REQ/CFM.

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

Mode — The mode of the network operation. This parameter should always be set to ‘3’

Format — Specifies the format of the network operator string. This parameter, which is a single character, should always be set to ‘0’

forceResendingFormat — If TRUE, then the format of the answer will be set previous to the inquiry for the operator name.

copsString — Pointer to a 0-terminated string with the network operator name, e.g. “CSR Bluetooth”.

cmeeResultCode — The result of the query, e.g. CSR_BT_AT_COMMAND_OK. The parameter is of the type CsrUint8 and the specific values are defined in the csr_bt_hf.h file.

Library Function

```
    void CsrBtHfCopsReqSend( CsrUint8 mode,
                             CsrUint8 format,
                             CsrBool forceResendFormat,
                             CsrBtHfConnectionId connectionId)
```

`connectionId` — The same as in the table above.

Mode — The same as in the table above.

Format — The same as in the table above.

`forceResendFormat` — The same as in the table above

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfCopsReqSend('3','0',FALSE,0)
```

4.15 CSR_BT_HF_GET_SUBSCRIBER_NUMBER_INFORMATION

Parameters Primitives	type	connectionId	cnumString	cmeeResultCode
CSR_BT_HF_GET_SUBSCRIBER_NUMBER_INFORMATION_REQ	✓	✓		
CSR_BT_HF_GET_SUBSCRIBER_NUMBER_INFORMATION_CFM	✓	✓		✓
CSR_BT_HF_GET_SUBSCRIBER_NUMBER_INFORMATION_IND	✓	✓	✓	

Table 18: CSR_BT_HF_GET_SUBSCRIBER_NUMBER_INFORMATION Primitives

Description

The gateway response when inquired about the subscriber number information. When the gateway answers with the subscriber number, the profile issues the CSR_BT_HF_GET_SUBSCRIBER_NUMBER_IND to the application with the information received. For each subscriber identity received from the gateway, the profile will deliver it to the application in a separate CSR_BT_HF_GET_SUBSCRIBER_NUMBER_IND message. When the gateway send "OK" or "ERROR" either after having sent the subscriber information or without having sent it (i.e. if no information is available), the profile will issue the CSR_BT_HF_GET_SUBSCRIBER_NUMBER_CFM.

Parameters

type
The signal identity, CSR_BT_HF_GET_SUBSCRIBER_NUMBER_INFORMATION_REQ/CFM/IND.

connectionId
Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting "CSR_BT_HF_CONNECTION_ALL"

cnumString
Pointer to the 0-terminated alphanumeric string received from the gateway, which contains the number, e.g. "123456789" and type of number and service, as described in [GSM 07.07].

cmeeResultCode
The result of the query, e.g. CSR_BT_CME_SUCCESS. The specific values are defined in the csr_bt_hf.h file.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfGetSubscriberNumberInformationReqSend(0);
```

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.16 CSR_BT_HF_GET_CURRENT_CALL_LIST

Parameters / **Primitives**	type	connectionId	clccString	cmeeResultCode
CSR_BT_HF_GET_CURRENT_CALL_LIST_REQ	✓	✓		
CSR_BT_HF_GET_CURRENT_CALL_LIST_IND	✓	✓	✓	
CSR_BT_HF_GET_CURRENT_CALL_LIST_CFM	✓	✓		✓

Table 19: CSR_BT_HF_GET_CURRENT_CALL_LIST Primitives

Description

These primitives are used for getting information about the calls present at the gateway and their respective status, using the AT command AT+CLCC, as specified in [GSM 07.07]. The application sends the CSR_BT_HF_GET_CURRENT_CALL_LIST_REQ message and for each call present at the gateway, it will receive a CSR_BT_HF_GET_CURRENT_CALL_LIST_IND with information about the call. When the whole operation is completed and the application has received information about all existing calls, it will receive the CSR_BT_HF_GET_CURRENT_CALL_LIST_CFM signal.

Parameters

type — The signal identity, CSR_BT_HF_GET_CURRENT_CALL_LIST_REQ / CFM / IND.

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

ClccString — Pointer to a 0-terminated alphanumeric string which contains the call information as received from the gateway. For details about the contents, see [GSM 07.07].

cmeeResultCode — The result of the query, e.g. CSR_BT_CME_SUCCESS. The specific values are defined in the csr_bt_hf.h file.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfGetCurrentCallListReqSend(0);
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.17 CSR_BT_HF_SET_EXTENDED_AG_ERROR_RESULT_CODE

Primitives \ Parameters	type	connectionId	enable	cmeeResultCode
CSR_BT_HF_SET_EXTENDED_AG_ERROR_RESULT_CODE_REQ	✓	✓	✓	
CSR_BT_HF_SET_EXTENDED_AG_ERROR_RESULT_CODE_CFM	✓	✓		✓

Table 20: CSR_BT_HF_SET_EXTENDED_AG_ERROR_RESULT_CODE Primitives

Description

This signal is used for enabling or disabling the use of extended error codes by the gateway device. The HF device will send the command AT+CMEE=1 for enabling this feature, and the AT+CMEE=0 for disabling it. If enabled, the HFG will use extended error codes to notify the HF the outcome of an operation.

Parameters

type
The signal identity, CSR_BT_HF_SET_EXTENDED_AG_ERROR_RESULT_CODE_REQ / CFM

cmeeResultCode
Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results have not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

connectionId
Unique value used for identifing the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

Enable
Boolean to determine whether the feature shall be enabled or disabled.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfSetExtendedAgErrorResultCodeReqSend(0,TRUE);
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.18 CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION

Parameters Primitives	type	connectionId	enable	cmeeResultCode	clipString
CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION_REQ	✓	✓	✓		
CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION_CFM	✓	✓		✓	
CSR_BT_HF_CALL_NOTIFICATION_IND	✓	✓			✓

Table 21: CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION Primitives

Description

This CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION_REQ signal is used for enabling or disabling the calling line identification feature, depending on the value of the parameter “enable”. The HF will send the AT+CLIP command towards the HFG. When the enabling or disabling operation has been performed, the application will receive a CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION_CFM with the corresponding result code. Upon reception of an incoming call, if the feature is enabled, the application will receive a CSR_BT_HF_CALL_NOTIFICATION_IND message containing a string with the calling party’s information as specified in [GSM07.07].

Parameters

type — The signal identity, CSR_BT_HF_SET_CALL_NOTIFICATION_INDICATION_REQ / CFM or CSR_BT_HF_CALL_NOTIFICATION_IND

cmeeResultCode — Error code returned by the HFG. Values defined in csr_bt_hf.h and the type of the received error code is a CSRUint8.

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

Enable — Boolean to determine whether the feature shall be enabled or disabled.

ClipString — Pointer to a 0-terminated alphanumeric string which contains the CLIP information as received from the gateway. For details about the contents, see [GSM 07.07].

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfSetCallNotificationIndicationReqSend(0,TRUE);
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.19 CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION

Parameters Primitives	type	connectionId	enable	cmeeResultCode	ccwaString
CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION_REQ	✓	✓	✓		
CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION_CFM	✓	✓		✓	
CSR_BT_HF_CALL_WAITING_NOTIFICATION_IND	✓	✓			✓

Table 22: CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION Primitives

Description

This CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION_REQ signal is used for enabling or disabling the ability at the HFG to indicate that an incoming call is waiting while engaged in another call already. This feature will be enabled or disabled, depending on the value of the parameter “enable”. The HF will send the AT+CCWA command towards the HFG. When the enabling or disabling operation has been performed, the application will receive a CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION_CFM with the corresponding result code. Upon reception of a waiting call, if the feature is enabled, the application will receive a CSR_BT_HF_CALL_WAITING_NOTIFICATION_IND message containing a string with the calling party’s information as specified in [GSM07.07].

Parameters

Type
The signal identity, CSR_BT_HF_SET_CALL_WAITING_NOTIFICATION_REQ / CFM or CSR_BT_HF_CALL_WAITING_NOTIFICATION_IND

connectionId
Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

Enable
Boolean to determine whether the feature shall be enabled or disabled.

cmeeResultCode
Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results has not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

ccwaString
Pointer to a 0-terminated alphanumeric string which contains the call waiting information as received from the gateway. For details about the contents, see [GSM 07.07].

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfSetCallWaitingNotificationReqSend(0,TRUE);
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.20 CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND

Parameters / Primitives	type	connectionId	index	value	name
CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND	✓	✓	✓	✓	✓

Table 23: CSR_BT_HF_STATUS_INDICATOR_UPDATE Primitives

Description

The CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND signal is the HF/HS profile interpretation of the “+CIEV” AT command sent from the HFG to the HF. While establishing a SLC the HF/HS profile receives a list of status indicators that are supported by the HFG and a list with the current status of each indicator. During service level connection establishment the HF/HS profile will send a CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND for each supported status indicator. Furthermore the HF/HS profile will send a CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND each time it receives a “+CIEV” command from the HFG.

Parameters

type — The signal identity, CSR_BT_HF_STATUS_INDICATOR_UPDATE_IND.

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

Index — Index value in the CIND string

value — The value of the status indicator. The specific parameter values are present in the file csr_bt_hf.h and it is of the type CsrUint8.

name — Pointer to a string containing the name of the status indicator. E.g. “service”, “call”, “callsetup”, “signal”, “roam”, “battchg” or “callheld”.

NOTE: It is strongly recommended that the HF device issues the “CSR_BT_HF_GET_CURRENT_CALL_LIST_REQ” command whenever the “call”, “callsetup” and “callheld” indicators are received and several calls are involved. Due to limitations in the specification, this is the only way to precisely determine what particular call is in what particular state (i.e. active, incoming or held).

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.21 CSR_BT_HF_SET_STATUS_INDICATOR_UPDATE

Primitives \ Parameters	type	connectionId	enable	cmeeResultCode
CSR_BT_HF_SET_STATUS_INDICATOR_UPDATE_REQ	✓	✓	✓	
CSR_BT_HF_SET_STATUS_INDICATOR_UPDATE_CFM	✓	✓		✓

Table 24: CSR_BT_HF_SET_STATUS_INDICATOR_UPDATE Primitives

Description

It is possible to enable or disable the status indication feature in the remote HFG device at run time. To do that, the application needs to use the CSR_BT_HF_SET_STATUS_INDICATOR_UPDATE_REQ message. Per default, the status indication feature is enabled at SLC establishment. But if the HF application does not want to receive indications such as call setup status, call status, battery status, etc…for some reason, it shall use the request signal mentioned. The profile will send a confirmation message back when the operation has been performed.

Parameters

type — The signal identity, CSR_BT_HF_SET_STATUS_INDICATOR_UPDATE_REQ/CFM

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

Enable — Boolean to determine whether the feature shall be enabled or disabled.

cmeeResultcode — Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results have not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.22 CSR_BT_HF_SET_ECHO_AND_NOISE

Parameters Primitives	type	connectionId	enable	cmeeResultCode
CSR_BT_HF_SET_ECHO_AND_NOISE_REQ	✓	✓	✓	
CSR_BT_HF_SET_ECHO_AND_NOISE_CFM	✓	✓		✓

Table 25: CSR_BT_HF_SET_ECHO_AND_NOISE Primitives

Description

This CSR_BT_HF_SET_ECHO_AND_NOISE_REQ signal is used for enabling or disabling the echo and noise reduction functionalities in the HFG, depending on the value of the parameter "enable". The HF will send the AT+NREC command towards the HFG. When the enabling or disabling operation has been performed, the application will receive a CSR_BT_HF_SET_ECHO_AND_NOISE_CFM with the corresponding result code.

Parameters

Type — The signal identity, CSR_BT_HF_SET_ECHO_AND_NOISE_REQ / CFM

cmeeResultCode — Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results have not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting "CSR_BT_HF_CONNECTION_ALL"

enable — Boolean to determine whether the feature shall be enabled or disabled.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfSetEchoAndNoiseReqSend(0,TRUE);
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

4.23 CSR_BT_HF_SET_VOICE_RECOGNITION

Parameters Primitives	type	connectionId	start	cmeeResultCode	started
CSR_BT_HF_SET_VOICE_RECOGNITION_REQ	✓	✓	✓		
CSR_BT_HF_SET_VOICE_RECOGNITION_CFM	✓	✓		✓	
CSR_BT_HF_SET_VOICE_RECOGNITION_IND	✓	✓			✓

Table 26: CSR_BT_HF_SET_VOICE_RECOGNITION Primitives

Description

This CSR_BT_HF_SET_VOICE_RECOGNITION_REQ signal is used for starting or stopping a voice recognition operation, depending on the value of the parameter “start”. The HF will send the AT+BVRA command towards the HFG. When the start or stop operation has been performed, the application will receive a CSR_BT_HF_SET_VOICE_RECOGNITION_CFM with the corresponding result code.

Parameters

Type — The signal identity, CSR_BT_HF_SET_VOICE_RECOGNITION_REQ / CFM

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

start — Boolean to determine whether the operation shall be started or stopped.

cmeeResultCode — Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results have not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

started — Voice recognition input sequence in the AG is started/stopped.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfSetVoiceRecognitionReqSend(0,TRUE);
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.24 CSR_BT_HF_BT_INPUT

Primitives \ Parameters	type	connectionId	dataRequest	dataRespString	cmeeResultCode
CSR_BT_HF_BT_INPUT_REQ	✓	✓	✓		
CSR_BT_HF_BT_INPUT_CFM	✓	✓		✓	✓

Table 27: CSR_BT_HF_BT_INPUT Primitives

Description

This CSR_BT_HF_BT_I NPUT_REQ signal is used for requesting a phone number attached to a voice tag. The HF sends the AT+BINP=1 message to the HFG as specified in [HF]. When the operation is performed, the application will receive a CSR_BT_HF_BT_INPUT_CFM message with a string containing the number if any, and with a proper result code.

Parameters

Type	The signal identity, CSR_BT_HF_BT_INPUT_REQ / CFM
connectionId	Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”
dataRequest	According to [HF] this can only be 1 so far: request a phone number attached to a voice tag. Any other values are not specified for the hands-free profile, but this is open for future extensibility.
dataRespString	Pointer to a 0-terminated alphanumeric string which contains the information received from the gateway as specified in [HF]
cmeeResultCode	Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results have not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfBtInputReqSend(0,TRUE);
```

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.25 CSR_BT_HF_GENERATE_DTMF

Primitives \ Parameters	type	connectionId	dtmf	cmeeResultCode
CSR_BT_HF_GENERATE_DTMF_REQ	✓	✓	✓	
CSR_BT_HF_GENERATE_DTMF_CFM	✓	✓		✓

Table 28: CSR_BT_HF_GENERATE_DTMF Primitives

Description

This CSR_BT_HF_GENERATE_DTMF_REQ signal is used for requesting the HFG to generate a DTMF tone during a call. The HF sends the AT+VTS command to the HFG. When the operation is performed, the application will receive a CSR_BT_HF_GENERATE_DTMF_CFM message with a proper result code.

Parameters

type — The signal identity, CSR_BT_HF_GENERATE_DTMF_REQ / CFM

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

dtmf — The DTMF tone to generate. This is the ASCII representation of the values 0-9, * or #.

cmeeResultCode — Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results have not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

Example

Here is an example of how to send the request signal using the library function.

```
    CsrBtHfGenerateDTMFReqSend(0,’1’);
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.26 CSR_BT_HF_SPEAKER_GAIN_STATUS

Primitives \ Parameters	type	gain	connectionId	cmeeResultCode
CSR_BT_HF_SPEAKER_GAIN_STATUS_REQ	✓	✓	✓	
CSR_BT_HF_SPEAKER_GAIN_STATUS_CFM	✓		✓	✓
CSR_BT_HF_SPEAKER_GAIN_IND	✓	✓	✓	

Table 29: CSR_BT_HF_SPEAKER_GAIN_STATUS Primitives

Description

The CSR_BT_HF_SPEAKER_GAIN_IND indicates a change in speaker volume from the HFG or HAG. The CSR_BT_HF_SPEAKER_GAIN_STATUS_REQ may be used for controlling the speaker gain in the HF or HS and inform the HFG or HAG so it can synchronize its own gain values with the ones in the HF device. In this case, the HF will send the AT+VGS command to the gateway and when the gateway has handled it and responded to it, the application will receive the CSR_BT_HF_SPEAKER_GAIN_STATUS_CFM message with the appropriate result code.

Parameters

type — The signal identity, CSR_BT_HF_SPEAKER_GAIN_STATUS_REQ/CFM or CSR_BT_HF_SEAKER_GAIN_IND.

gain — Value for volume setting. The range must be in the interval 0 to 15 (both included). It is possible to define a maximum value. This can be set using the MAX_VGS_VALUE in the csr_bt_hf_prim.h file. The parameter is of the type CsrUint8.

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

cmeeResultCode — Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results have not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

Library Function

```
void CsrBtHfSpeakerGainStatusReqSend ( CsrUint8     gain,
                                       CsrBtHfConnectionId      theConnectionId)
```

gain — The same as in the table above.

theConnectionId — The same as “conenctionId” in the table above

Example

Here is an example of how to send the request signal using the library function.

```
CsrBtHfSpeakerGainStatusReqSend(8, 0)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.27 CSR_BT_HF_MIC_GAIN_STATUS

Primitives \ Parameters	type	gain	connectionId	cmeeResultCode
CSR_BT_HF_MIC_GAIN_STATUS_REQ	✓	✓	✓	
CSR_BT_HF_MIC_GAIN_STATUS_CFM	✓		✓	✓
CSR_BT_HF_MIC_GAIN_IND	✓	✓	✓	

Table 30: CSR_BT_HF_MIC_GAIN_STATUS Primitives

Description

The CSR_BT_HF_MIC_GAIN_IND indicates a change in microphone volume from the HFG or HAG. The CSR_BT_HF_MIC_GAIN_STATUS_REQ may be used for controlling the microphone gain in the HF or HS and inform the HFG or HAG so it can synchronize its own gain values with the ones in the HF device. In this case, the HF will send the AT+VGM command to the gateway and when the gateway has handled it and responded to it, the application will receive the CSR_BT_HF_MIC_GAIN_STATUS_CFM message with the appropriate result code.

Parameters

type — The signal identity, CSR_BT_HF_MIC_GAIN_STATUS_REQ/CFM or CSR_BT_HF_MIC_GAIN_IND.

gain — Value for volume setting. The range must be in the interval 0 to 15 (both included). It is possible to define a maximum value. This can be set using the MAX_VGM_VALUE in the csr_bt_hf_prim.h file. The parameter is of the type CsrUint8.

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

cmeeResultCode — Cme error result code of the operation if success (the AG responded OK) the result is CME_SUCCESS. Any other result code means the operation failed if CMEE results have not been enabled and ERROR response from the AG will be mapped to CME_AG_FAILURE.

Library Function

```
    void CsrBtHfMicGainStatusReqSend (CsrUint8 gain, CsrBtHfConnectionId
                                                          theConnectionId)
```

Gain — The same as in the table above.

theConnectionId — The same as “connenctionId” in the table above

Example

Here is an example of how to send the request signal using the library function.

```
   CsrBtHfMicGainStatusReqSend(8, 0)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.28 CSR_BT_HF_DIAL

Parameters / Primitives	type	connectionId	command	number	cmeeResultCode
CSR_BT_HF_DIAL_REQ	✓	✓	✓	✓	
CSR_BT_HF_DIAL_CFM	✓	✓			✓

Table 31: CSR_BT_HF_DIAL Primitives

Description

This signal is used by the HF/HS to request that the gateway shall make an outgoing call. The call can be to a specific number, to a number stored in the memory of the gateway or to the last dialled number. The type of call is specified with the field “command”. Once the message has been handled by the gateway and it has responded to it, the application will receive the CSR_BT_HF_DIAL_CFM message with an appropriate result code.

Parameters

type The signal identity, CSR_BT_HF_DIAL_REQ/_CFM.

connectionId Unique identifier for the connection where the call shall be rejected. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

Command This field can take three different values (all of them defined in csr_bt_hf_prim.h):

1. CSR_BT_HF_DIAL_NUMBER (0x00)
   The call shall be made to a given number, contained in the string “number”
2. CSR_BT_HF_DIAL_MEMORY(0x01)
   The call shall be made to a number stored in the gateway’s memory in the position given in the string “number”
3. CSR_BT_HF_DIAL_REDIAL(0x02)
   The call shall be made to the last dialled number and the “number” field will be a NULL pointer

number Pointer to a 0-terminated alphanumeric string with information as described above.

cmeeResultCode Error code returned by the HFG. Values defined in csr_bt_hf.h and the type of the received error code is a CSRUint8.

Library Function

```
    void CsrBtHfDialReqSend(CsrBtHfConnectionId connectionId,
                                           CsrBtHfDialCommand  command,
                                           CsrCharString     *number)
```

connectionId The same as in the table above.

command The same as in the table above.

Number The same as in the table above.

Example

Here is an example of how to send the request signal using the library function:

```
   CsrBtHfDialReqSend(0x00,CSR_BT_HF_DIAL_REDIAL,NULL)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.29 CSR_BT_HF_CALL_ANSWER

Parameters Primitives	type	connectionId	cmeeResultCode
CSR_BT_HF_CALL_ANSWER_REQ	✓	✓	
CSR_BT_HF_CALL_ANSWER_CFM	✓	✓	✓

Table 32: CSR_BT_HF_CALL_ANSWER Primitives

Description

This signal is used by the HF/HS for answering an incoming call.

Parameters

Type — The signal identity, CSR_BT_HF_CALL_ANSWER_REQ/_CFM.

connectionId — Unique identifier for the connection where the call shall be rejected. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

cmeeResultCode — Error code returned by the HFG. Values defined in csr_bt_hf.h and the type of the received error code is a CSRUint8.

Library Function

```
    void CsrBtHfAnswerReqSend(CsrBtHfConnectionId connectionId)
```

connectionId — The same as in the table above.

Example

Here is an example of how to send the request signal using the library function.

```
   CsrBtHfAnswerReqSend(0x00)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.30 CSR_BT_HF_CALL_END

Primitives \ Parameters	type	connectionId	cmeeResultCode
CSR_BT_HF_CALL_END_REQ	✓	✓	
CSR_BT_HF_CALL_END_CFM	✓	✓	✓

Table 33: CSR_BT_HF_CALL_END Primitives

Description

The CSR_BT_HF_CALL_END_REQ signal is used for rejecting incoming calls or interrupting an already ongoing call process. If this signal is used on an HF connection it will be mapped to the AT command AT+CHUP; if used on a HS connection, it will be mapped to the AT+CKPD=200 command. When the gateway responds to any of the commands mentioned, the application will receive the CSR_BT_HF_CALL_END_CFM message.

Parameters

Type	The signal identity, CSR_BT_HF_CALL_END_REQ / CFM.
connectionId	Unique identifier for the connection where the call shall be rejected. The type is `CsrBtHfConnectionId`. The value 0xFFFFFFFF is used for denoting "CSR_BT_HF_CONNECTION_ALL"
cmeeResultCode	Error code returned by the HFG. Values defined in csr_bt_hf.h and the type of the received error code is a CSRUint8.

Library Function

```
    void CsrBtHfCallEndReqSend(CsrBtHfConnectionId connectionId)
```

Example

Here is an example of how to send the signal using the library function.

```
   CsrBtHfCallEndReqSend(0xFFFFFFFF)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.31 CSR_BT_HF_CALL_HANDLING

Primitives \ Parameters	type	connecitonId	Command	Index	event	cmeeResultCode
CSR_BT_HF_CALL_HANDLING_IND	✓	✓	✓		✓	
CSR_BT_HF_CALL_HANDLING_REQ	✓	✓	✓	✓		
CSR_BT_HF_CALL_HANDLING_CFM	✓	✓				✓

Table 34: CSR_BT_HF_CALL_HANDLING Primitives

Description

The application may perform outgoing calls, answer or reject incoming calls, pot calls on hold and/or retrieve them, etc… The outcome of these operations will be communicated to the application too.

Parameters

type — The signal identity; CSR_BT_HF_CALL_HANDLING_REQ/__CFM/_IND

connectionId — Unique value used for identifying the connection. The type is `CsrBtHfConnectionId`. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

command — The value representing the call operation requested. The commands that can be used are defined in csr_bt_hf_prim.h:

- CSR_BT_RELEASE_ALL_HELD_CALL (0x00). Sends AT+CHLD=0
- CSR_BT_RELEASE_ACTIVE_ACCEPT (0x01). Sends AT+CHLD=1
- CSR_BT_RELEASE_SPECIFIED_CALL (0x02). Sends AT+CHLD=1x
- CSR_BT_HOLD_ACTIVE_ACCEPT (0x03). Sends AT+CHLD=2
- CSR_BT_REQUEST_PRIVATE_WITH_SPECIFIED (0x04). Sends AT+CHLD=2x
- CSR_BT_ADD_CALL (0x05). Sends AT+CHLD=3.
- CSR_BT_CONNECT_TWO_CALLS (0x06). Sends AT+CHLD=4.
- CSR_BT_BTRH_PUT_ON_HOLD (0x07). Sends AT+BTRH=0
- CSR_BT_BTRH_ACCEPT_INCOMING (0x08). Sends AT+BTRH=1
- CSR_BT_BTRH_REJECT_INCOMING (0x09). Sends AT+BTRH=2
- CSR_BT_BTRH_READ_STATUS (0x0A). Sends AT+BTRH?

The parameter is of the type callHandlingCommand_t. For more information about these AT commands, please see [HF].

index — Value (x) that determines what call the command aims to when issuing AT+CHLD=2x or AT+CHLD=1x

event — The value representing the response code received from the HFG about a call operation requested or a call operation performed at the HFG without being requested from the HF. The possible values that can be received are defined in csr_bt_hf_prim.h:

- CSR_BT_BTRH_INCOMING_ON_HOLD (0x00). Received +BTRH:0
- CSR_BT_BTRH_INCOMING_ACCEPTED (0x01). Received +BTRH:1
- CSR_BT_BTRH_INCOMING_REJECTED (0x02). Received +BTRH:2

The parameter is of the type callHandlingEvent_t. For more information about these AT commands, please see [HF].

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

cmeeResultCode	Error code returned by the HFG. Values defined in csr_bt_hf.h and the type of the received error code is a CSRUint8.

Library Function

```
void CsrBtHfCallHandlingReqSend(callHandlingCommand_t command,
                                CsrUint8 index,
                                CsrBtHfConnectionId connectionId)
```

Example

Here is an example of how to send the signal using the library function.

```
CsrBtHfCallHandlingReqSend(CSR_BT_RELEASE_ALL_HELD_CALL,0x00,0x00)
```

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.32 CSR_BT_HF_AT_CMD

Primitives \ Parameters	type	connectionId	atCmdString	cmeeResultCode
CSR_BT_HF_AT_CMD_IND	✓	✓	✓	
CSR_BT_HF_AT_CMD_CFM	✓	✓		✓
CSR_BT_HF_AT_CMD_REQ	✓	✓	✓	

Table 35: CSR_BT_HF_AT_CMD Primitives

Description

An AT-command is received or sent by the application layer. The data must be/is compiled as sent/received on the air interface, i.e. no interpretation takes place in the HF/HS manager. The application issues the CSR_BT_HF_AT_CMD_REQ in order to send an AT command towards the gateway device. When the remote device sends response codes to the local device, the application will receive a CSR_BT_HF_AT_CMD_IND containing a string with the response code received. However, when the remote device responds with a result code to a command issued by the local device, the HF manager will send the CSR_BT_HF_AT_CMD_CFM to the application.

The application layer is responsible for command validation and control.

Parameters

Type — The signal identity, CSR_BT_HF_AT_CMD_IND/CFM/REQ.

connectionId — Unique value used for identifying the connection. The type is `CsrBtHfConnectionId`. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

*atCmdString — AT-command bytes as received from the HF/HS manager. The pointer shall point to an allocated string that is 0-terminated. It is up to the sender of the AT-command to format the payload correctly, hence, the received payload cannot be expected to be AT-formatted according to specification.

It is the responsibility of the receiving layer to free the contents of the payload parameter. This means that the application has to free the contents of the CSR_BT_HF_AT_CMD_IND’s payload parameter.

cmeeResultCode — The result of the query, e.g. CSR_BT_CME_SUCCESS. The specific values are defined in the csr_bt_hf.h file.

Library Function

```
    void CsrBtHfAtCmdReqSend( CsrUint16       len,
                             CsrUint8     *payload,            CsrBtHfConnectionId
                             connectionId)
```

Len — The length of the string to send in bytes including the NULL terminator.

*payload — The same as “atCmdString” in the table above.

connectionId — The same as connectionId in the table above

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

Example

Here is an example of how to send the signal using the library function.

```
char *str;
str = CsrPmalloc(11);
CsrMemCpy(str, "AT+BTRH=2\r\0",11);
CsrBtHfAtCmdReqSend(10, (CsrCharString*) str, 0x00)
```

Note that the application must not send new AT commands (neither using the CSR_BT_HF_AT_CMD_REQ message nor via the rest of the API primitives) on a determined connection as long as the previous one has not been answered by the remote device.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.33 CSR_BT_HF_SECURITY_IN / OUT

Parameters Primitives	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_HF_SECURITY_IN_REQ	✓	✓	✓		
CSR_BT_HF_SECURITY_IN_CFM	✓			✓	✓
CSR_BT_HF_SECURITY_OUT_REQ	✓	✓	✓		
CSR_BT_HF_SECURITY_OUT_CFM	✓			✓	✓

Table 36: CSR_BT_HF_SECURITY_IN and CSR_BT_HF_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_HF_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

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

	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

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.34 CSR_BT_HF_CALL_RINGING

Primitives \ Parameters	type	connectionId
CSR_BT_HF_CALL_RINGING_IND	✓	✓

Table 37: CSR_BT_HF_CALL_RINGING Primitives

Description

Indicates an incoming call from the HFG or HAG side, the application can accept or reject the call.

Parameters

Type	The signal identity, CSR_BT_HF_CALL_RINGING_IND.
connectionId	Unique value used for identifying the connection. The type is `CsrBtHfConnectionId`. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.35 CSR_BT_HF_C2C_SF

Parameters / Primitives	type	connectionId	number	value	*indicators	indicatorsLength
CSR_BT_HF_C2C_SF_REQ	✓	✓	✓	✓		
CSR_BT_HF_C2C_SF_IND	✓	✓			✓	✓

Table 38: The HFG_C2C_SF primitives

Description

Note: The possibility to change the codec using CSR to CSR proprietary commands is deprecated and will be removed from the implementation. It is therefore recommended not to indicate support for the CSR_BT_C2C_ADPCM_IND **or the** CSR_BT_C2C_SAMPLE_RATE_IND **parameters.**

This signal is used for two things. The CSR_BT_HF_C2C_SF_IND signal reports what CSR2CSR features the gateway supports. The request is used for enabling/disabling a particular CSR2CSR feature locally and let the gateway know about it, such that it will be used when the gateway offers support for it.

Note that this signal can be sent as soon as the HF profile has been activated and before any service level connections are present. The HF will store the desired setting and automatically enable it if the application has previously requested a feature.

Note: In order for a feature to be usable both the HFG and the HF must enable it, so it may BE necessary for the application to keep track of the HFG indicators. If the application sends CSR2CSR requests that have not been agreed on by both the HF and HFG, the HF will ignore the messages.

Parameters

type — Signal identity CSR_BT_HF_C2C_SF_REQ/IND.

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

number — The CSR2CSR indicator index which should be enabled/disabled. The following compiler defines can be found in the header file *csr_bt_hf.h:*

- CSR_BT_C2C_NAME_IND
  Index 1, transfer names with call waiting and rings (clip). This is not needed anymore, as the HFG will send the name information automatically whenever available, regardless of whether this feature is enabled or not.

- CSR_BT_C2C_TXT_IND
  Index 2, support for unsolicited text

- CSR_BT_C2C_SMS_IND
  Index 3, support for SMS arrival notification and text transfer

- CSR_BT_C2C_BAT_IND
  Index 4, battery charge level notifications

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

- CSR_BT_C2C_PWR_IND
  Index 5, power status notifications

- CSR_BT_C2C_ADPCM_IND
  Index 6, audio codec settings. Can be set to a bitmask built with the values:

  - 0x01 CVSD
  - 0x02 2-bit ADPCM
  - 0x04 4-bit ADPCM

- CSR_BT_C2C_SAMPLE_RATE_IND
  Index 7, audio data sampling rate. Can be set to a bitmask built with the values:

  - 0x01 8 KHz
  - 0x02 16 KHz

value

The value of the given indicator. A value of “0” (zero) means off while a “1” (one) is on. In the case of the CSR_BT_C2C_ADPCM_IND and CSR_BT_C2C_SAMPLE_RATE_IND, the values can be any of the above mentioned, or 0 if none is supported. Note that the value is transferred as a real value, *not* an ASCII character.

*indicators

When the application receives the supported/enabled features from the HFG, they are packed into an array of characters. This is the pointer for that array, and must be CsrPfree()’ed after use by the application.

The indicator indexes are the same as explained above in the *number* parameter.

Note: It is critical that the application does not try to decode indexes that exceed the *indicatorsLength* or indicator indexes that are unknown for the application.

indicatorsLength

The length of the *indicators* array. There shall be made no assumptions on the length of the array, as new CSR2CSR features may be added in the future without warning.

Library Function

```
void CsrBtHfC2cSfReqSend( CsrBtHfConnectionId       connectionId,
                          CsrUint8       number,
                          CsrUint8       value     )
```

All parameters are as described above in the signal primitive details.

4.36 CSR_BT_HF_SET_C2C_AUDIO_CODEC

Parameters **Primitives**	type	connectionId	value
CSR_BT_HF_SET_C2C_AUDIO_CODEC_REQ	✓	✓	✓
CSR_BT_HF_SET_C2C_AUDIO_CODEC_IND	✓	✓	✓

Table 39: CSR_BT_HF_SET_C2C_AUDIO_CODEC Primitives

Description

Note: This signal and the possibility to change the codec using CSR to CSR proprietary commands is deprecated and will be removed from the implementation. It is recommended not to use it.

It is possible for the HF application to change the codec to be used for audio connections to a remote CSR based gateway, if it also supports this feature. The change will only take place if there is support for it in the HW. During audio connection establishment, the HF and HFG will then negotiate the codec to use: CVSD, 2-bit ADPCM or 4-bit ADPCM. The application may indicate support for more than one codec.

It is strongly recommended that application does not request change of codec settings unless they are absolutely positive that the new settings are supported by the HW/FW platform used. The application can find out by means of the primitive CSR_BT_HF_GET_C2C_ADPCM_LOCAL_SUPPORTED_REQ (see chapter 0)

Parameters

type Signal identity CSR_BT_HF_SET_C2C_AUDIO_CODEC_REQ/IND

connectionId Unique value used for identifying the connection. The type is CsrBtHfConnectionId.

value This field is a bitmask that consists of one of the following values, or a combination of them:

0x01	CVSD
0x02	2-bit ADPCM
0x04	4-bit ADPCM

Library Function

```
void CsrBtHfSetC2CAudioCodecReqSend (CsrBtHfConnectionId   connectionId,
                                      CsrUint8    value)
```

connectionId The same as in the table above

value The same as in the table above. The values mentioned above are also defined in csr_bt_hf_prim.h.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.37 CSR_BT_HF_GET_C2C_ADPCM_LOCAL_SUPPORTED

Parameters Primitives	type	result
CSR_BT_HF_GET_C2C_ADPCM_LOCAL_SUPPORTED_REQ	✓	
CSR_BT_HF_GET_C2C_ADPCM_LOCAL_SUPPORTED_IND	✓	✓

Table 40: CSR_BT_HF_GET_C2C_ADPCM_LOCAL_SUPPORTED Primitives

Description

Note: This signal and the possibility to change the codec using CSR to CSR proprietary commands is deprecated and will be removed from the implementation. It is recommended not to use it.

The HF application can find out whether the HW/FW platform supports ADPCM audio by means of this primitive. This operation should be performed prior to trying to change the codec settings.

Parameters

type	Signal identity CSR_BT_HF_GET_C2C_ ADPCM_LOCAL_SUPPORTED_REQ/IND
result	TRUE if ADPCM audio is supported; FALSE otherwise

Library Function

```
    void CsrBtHfGetC2CAdpcmLocalSupportedReqSend(void)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.38 CSR_BT_HF_SET_C2C_SAMPLE_RATE

Parameters Primitives	type	connectionId	value
CSR_BT_HF_SET_C2C_SAMPLE_RATE_REQ	✓	✓	✓
CSR_BT_HF_SET_C2C_SAMPLE_RATE_IND	✓	✓	✓

Table 41: CSR_BT_HF_SET_C2C_SAMPLE_RATE Primitives

Description

Note: This signal and the possibility to change the codec using CSR to CSR proprietary commands is deprecated and will be removed from the implementation. It is recommended not to use it.

The rate used for sampling audio data during an audio connection can be set from the application. This is done with the CSR_BT_HF_SET_C2C_SAMPLE_RATE_REQ primitive. This operation should only be performed if the HW/FW platform used supports it. Beware that this primitive can be used as soon as the HF profile has been activated and no connection is required before using it.

The primitive CSR_BT_HF_SET_C2C_SAMPLE_RATE_IND is sent to the application immediately after an audio connection has been established to indicate the sample rate actually used for that connection. This will only be the case if the application has used the CSR_BT_HF_SET_C2C_SAMPLE_RATE_REQ previously.

Parameters

type Signal identity CSR_BT_HF_SET_C2C_SAMPLE_RATE_REQ /IND

connectionId Unique value used for identifying the connection. The type is CsrBtHfConnectionId.

value This field is a bitmask that consists of one of the following values, or a combination of them:

0x01	8 KHz
0x02	16 KHz

Library Function

```
    void CsrBtHfSetC2CSampleRateReqSend (CsrBtHfConnectionId connectionId,
                                    CsrUint8    value)
```

connectionId The same as in the table above.

value The same as in the table above. The values mentioned above are also defined in csr_bt_hf.h.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.39 CSR_BT_HF_C2C_PWR

Parameters / Primitives	type	connectionId	value	cmeeResultCode
CSR_BT_HF_C2C_PWR_REQ	✓	✓	✓	
CSR_BT_HF_C2C_PWR_CFM	✓	✓		✓

Table 42: CSR_BT_HF_C2C_PWR Primitives

Description

If the CSR to CSR power source feature is enabled at both ends in a HF connection, then the HF device may inform the HFG about the power source used. The application can use the CSR_BT_HF_C2C_PWR_REQ message to do this. The primitive CSR_BT_HF_C2C_PWR_CFM is sent to the application when the gateway device has acknowledged the message.

Parameters

type Signal identity CSR_BT_HF_ C2C_PWR_REQ /CFM

connectionId Unique value used for identifying the connection. The type is CsrBtHfConnectionId.

value This field consists of one of the following values:

- 0x01 Battery poswered
- 0x02 External power source

cmeeResultCode The result of the query, e.g. CSR_BT_CME_SUCCESS. The specific values are defined in the csr_bt_hf.h file.

Library Function

```
    void CsrBtHfC2CPowerReqSend (CsrBtHfConnectionId connectionId,
                                CsrUint8    value)
```

connectionId The same as in the table above.

value The same as in the table above.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.40 CSR_BT_HF_C2C_BATT

Parameters Primitives	type	connectionId	value	cmeeResultCode
CSR_BT_HF_C2C_BATT_REQ	✓	✓	✓	
CSR_BT_HF_C2C_BATT_CFM	✓	✓		✓
CSR_BT_HF_C2C_BATT_IND	✓	✓		

Table 43: CSR_BT_HF_C2C_BATT Primitives

Description

If the CSR to CSR battery level in headset feature is enabled at both ends in a HF connection, then the HF device shall inform the HFG about the power source used. The application can use the CSR_BT_HF_C2C_BATT_REQ message to do this. The primitive CSR_BT_HF_C2C_BATT_CFM is sent to the application when the gateway device has acknowledged the message. Besides, the remote device may ask for the current battery level in the headset at any time during the connection. If it does, the application will receive a CSR_BT_HF_C2C_BATT_IND message, which must be answered with a CSR_BT_HF_C2C_BATT_REQ.

Parameters

type	Signal identity CSR_BT_HF_ C2C_BATT_REQ /CFM
connectionId	Unique value used for identifying the connection. The type is CsrBtHfConnectionId.
value	This field can take values in the range 0-9, 0 meaning empty battery and 9 meaning full battery.
cmeeResultCode	The result of the query, e.g. CSR_BT_CME_SUCCESS. The specific values are defined in the csr_bt_hf.h file.

Library Function

```
    void CsrBtHfC2CBattReqSend (CsrBtHfConnectionId connectionId,
                                CsrUint8    value)
```

connectionId	The same as in the table above.
value	The same as in the table above.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.41 CSR_BT_HF_C2C_SMS

Parameters / Primitives	type	connectionId	index	cmeeResultCode	smsString
CSR_BT_HF_C2C_GET_SMS_REQ	✓	✓	✓		
CSR_BT_HF_C2C_GET_SMS_CFM	✓	✓		✓	✓
CSR_BT_HF_C2C_SMS_IND	✓	✓	✓		

Table 44: CSR_BT_HF_C2C_SMS Primitives

Description

If the CSR to CSR SMS feature is enabled at both ends in a HF connection, then the HFG device may inform the HF about the reception of a new SMS. The application will receive a CSR_BT_HF_C2C_SMS_IND message in this case, which will contain the index of the SMS in the gateway's storage. The HF can use the primitive CSR_BT_HF_C2C_GET_SMS_REQ in order to retrieve an SMS from the remote device's storage. Once the HFG has answered to this request, the application will receive the message CSR_BT_HF_C2C_GET_SMS_CFM that contains a result code and, if the request has been successful, a pointer to a string containing the SMS requested.

Parameters

type — Signal identity CSR_BT_HF_ C2C_SMS_REQ /CFM /IND

connectionId — Unique value used for identifying the connection. The type is CsrBtHfConnectionId.

index — This is the index of the SMS in the HFG storage. The type is CsrUint8

cmeeResultCode — The result of the query, e.g. CSR_BT_CME_SUCCESS. The specific values are defined in the csr_bt_hf.h file.

smsString — Pointer to a 0-terminated string containing the SMS retrieved as sent from the HFG.

Library Function

```
    void CsrBtHfSetC2CGetSmsReqSend (CsrBtHfConnectionId connectionId,
                                     CsrUint8    index)
```

connectionId — The same as in the table above.

index — The same as in the table above.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.42 CSR_BT_HF_C2C_TXT

Parameters / Primitives	type	connectionId	txtString
CSR_BT_HF_C2C_TXT_IND	✓	✓	✓

Table 45: CSR_BT_HF_C2C_TXT Primitive

Description

If the CSR to CSR TXT feature is enabled at both ends in a HF connection, then the HFG device may send a text message to the HF. The application will receive a CSR_BT_HF_C2C_TXT_IND message with a NULL-terminated string containing the text received from the remote device.

Parameters

type	Signal identity CSR_BT_HF_C2C_TXT_IND
connectionId	Unique value used for identifying the connection. The type is CsrBtHfConnectionId.
`txtString`	Pointer to a 0-terminated string containing the SMS retrieved as sent from the HFG.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.43 CSR_BT_HF_INBAND_RING_SETTING_CHANGED

Parameters / Primitives	type	inbandRingingActivated	connectionId
CSR_BT_HF_INBAND_RING_SETTINGS_CHANGED_IND	✓	✓	✓

Table 46: CSR_BT_HF_INBAND_RING_SETTINGS_CHANGED Primitive

Description

The HF application is informed of any changes in the in-band ring tones setting at the HFG.

Parameters

type	Signal identity CSR_BT_HF_INBAND_RINGING_IND
inbandRingingActivated	TRUE if the HFG sends in-band ringtones; otherwise FALSE
connectionId	Unique value used for identifying the connection. The type is `CsrBtHfConnectionId`. The value 0xFFFFFFFF is used for denoting “CSR_BT_HF_CONNECTION_ALL”

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.44 CSR_BT_HF_DEREGISTER_TIME

Parameters / Primitives	type	waitSeconds	result
CSR_BT_HF_DEREGISTER_TIME_REQ	✓	✓	
CSR_BT_HF_DEREGISTER_TIME_CFM	✓		✓

Table 47: CSR_BT_HF_DEREGISTER_TIME Primitive

Description

Whenever a connection is established from a remote device, the HF profile will remove the service record used for that connection from the local service database. This is done to avoid other remote devices trying to connect to the local channel in that specific service record. However, some devices try to read some of the features stored in the service record even after the connection has been established. Therefore, the HF application may want to keep the service record for some time after connection for interoperability purposes. The profile provides an interface to do that: the CSR_BT_HF_DEREGISTER_TIME_REQ and CFM messages allow the application to determine the number of seconds that the profile must wait before removing the service record from the local service database. Per default, this time is 0, and the service record will be removed immediately after connection establishment.

BEWARE: As long as the service record exists in the service database, any other remote devices will be able to get information from it, and try to connect to the local device with the information in it. These attempts to connect will fail as long as there is an active connection on it!

Parameters

type	Signal identity CSR_BT_HF_DEREGISTER_TIME_REQ /CFM
waitSeconds	Number of seconds to wait.
result	Result of the operation: CSR_BT_CME_SUCCESS if the operation succeeds

Library Function

```
    void CsrBtHfSetDeregisterTimeReqSend (waitSeconds)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

4.45 CSR_BT_HF_INDICATION_ACTIVATION

Parameters Primitives	type	connectionId	indicatorBitMask	result
CSR_BT_HF_INDICATOR_ACTIVATION_REQ	✓	✓	✓	
CSR_BT_HF_INDICATOR_ACTIVATION_CFM	✓	✓		✓

Table 48: CSR_BT_HF_ACTIVATION_INDICATION Primitive

Description

When a Hands free connection exists between a remote gateway and the local HF device, the user may want to avoid receiving some indicators that it does not use. In this case, it can request that those indicators are not sent from the GW by issuing the CSR_BT_HF_INDICATOR_ACTIVATION_REQ. This operation will take effect for the active service level connection and only for as long as the connection is established. If the connection is released, and a new connection is established between the same two devices, the indicators will all be active again. Besides, the HF cannot disable the “call”, “call-setup” and “call-held” indicators.

Parameters

type — Signal identity CSR_BT_HF_INDICATOR_ACTIVATION_REQ /CFM

connectionId — Index of the connection that this primitive shall be executed on

indicatorBitMask — Bit mask with the indicators to enable/disable. If bit 0 has the value ‘1’, then the first indicator supported by the HFG in the “cindString” received in the CSR_BT_HF_SERVICE_CONNECT_IND/CFM message will be enabled. If it is 0, the indicator will be disabled. And so on for all the indicators supported by the remote device. This field is of type CsrUint16.

result — Result of the operation: CSR_BT_CME_SUCCESS if the operation succeeds

Library Function

```
    void CsrBtHfIndicatorActivationReqSend(indicatorBitMask, connectionId)
```

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

4.46 CSR_BT_HF_UPDATE_SUPPORTED_CODEC

Parameters Primitives	type	codecMask	enable	sendUpdate	resultSupplier	result
CSR_BT_HF_UPDATE_SUPPORTED_CODEC_REQ	✓	✓	✓	✓		
CSR_BT_HF_ UPDATE_SUPPORTED_CODEC_CFM	✓				✓	✓

Table 49: CSR_BT_HF_UPDATE_SUPPORTED_CODEC Primitive

Description

If the Hands-free device supports the codec negotiation feature (enabled at activation time), it is possible to enable and/or disable specific CODECS at run-time. When the application enables a particular codec, it will be available for use for audio connections with remote gateway devices. The hands-free profile will indicate this to the remote devices. Likewise, when the application disables a CODEC, it will become unavailable.

Parameters

type Signal identity CSR_BT_HF_UPDATE_SUPPORTED_CODEC_REQ /CFM

codecMask A bit mask specifing which codecs are to be affected. See definitions in csr_bt_hf.h.

enable Boolean to determine whether the codecs specified shall be enabled (TRUE) or disabled (FALSE).

sendUpdate Boolean to specify whether the change in codec support should be send over to the AG, if TRUE the update will be send. It is recommended to always send the update at least on codec upgrades.

result Result of the operation: CSR_BT_RESULT_CODE_HF_SUCCESS if the operation succeeds

resultSupplier This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Library Function

```
void CsrBtHfUpdateSupportedCodecReqSend(codec, enable)
```

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

5 Document References

Document	Reference
Bluetooth® Core Specification v.1.1, v.1.2 and v.2.0, profile section N/A	[BT]
CSR Synergy Bluetooth, SC – Security Controller API Description, Document no. api-0102-sc, profile section N/A	[SC]
The Bluetooth® Specification, Hands-free Profile, ver. 1.5 2005-08-01	[HF]
The Bluetooth Specification, Headset profile, version 1.2 2008-Dec-18	[HS]
Bluetooth® Hands-Free Profile Application Guidelines, profile section N/A	[CCAP]
CSR Synergy Bluetooth. CM – Connection Manager API Description, doc. no. api-0101-cm, profile section N/A	[CM]
Digital cellular telecommunications system (Phase 2+); Man-Machine Interface (MMI) of the Mobile Station (MS) (GSM 02.30 version 7.1.0 Release 1998), profile section N/A	[GSM02.30]
AT command set for GSM Mobile Equipment (ME) (GSM 07.07), profile section N/A	[GSM07.07]

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

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
HF	Hands-Free
HS	Headset
HFG	Hands-free Gateway
HAG	Headset Audio Gateway
SLC	Service Level Connection
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0

CSR Synergy Bluetooth 18.2.0 HF – Hands-Free API

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR’s non-disclosure agreement.

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.

© Cambridge Silicon Radio Limited 2011
This material is subject to CSR's non-disclosure agreement.