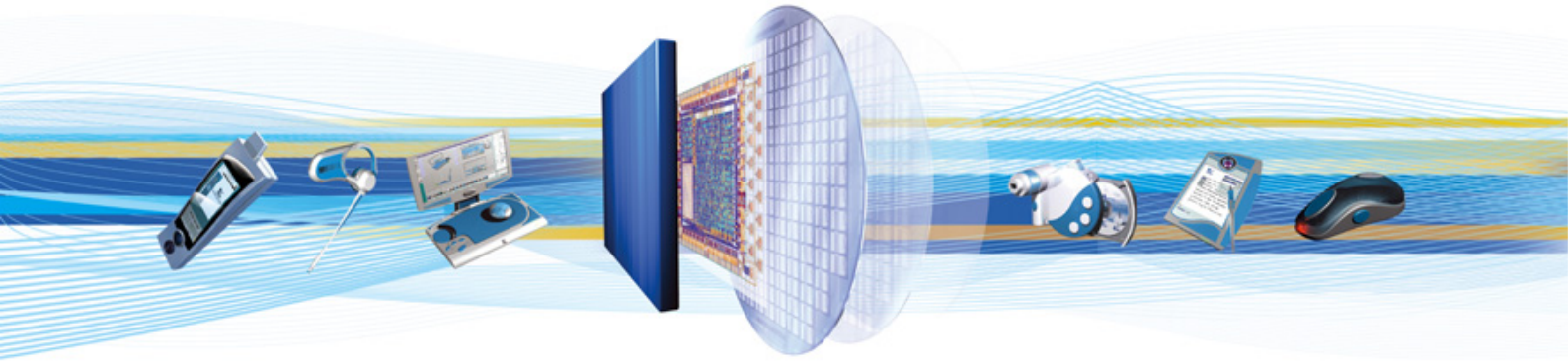
CSR Synergy Bluetooth 18.2.0

HDP – Health Device Profile

API Description

November 2011



Cambridge Silicon Radio Limited

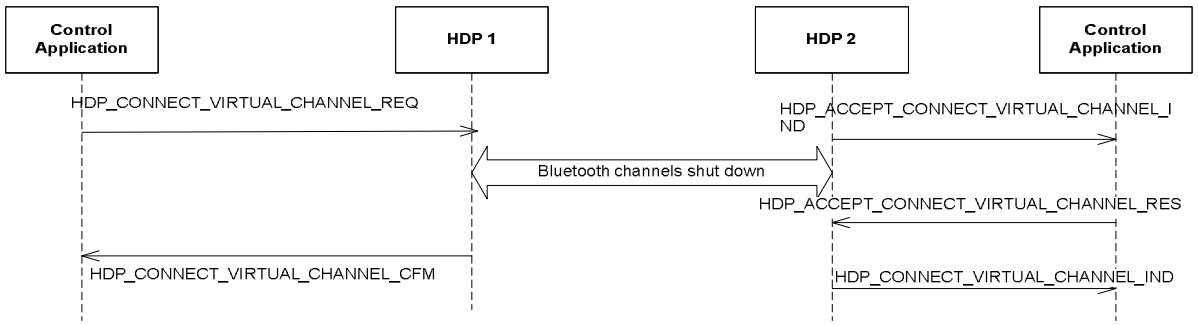
Churchill House
Cambridge Business Park
Cowley Road
Cambridge CB4 0WZ
United Kingdom

Registered in England and Wales 3665875

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

Contents

List of Figures



List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the functionality and message interface provided by CSR Synergy Bluetooth for using the Health Device Profile (HDP) specified by the Bluetooth Special Interest Group (SIG) and the IEEE 20601 Data Exchange Protocol IEEE.

2 Description

2.1 Introduction

The Health Device Profile (HDP) is an application profile that defines the requirements for qualified Bluetooth Healthcare and Fitness (referred to as 'health') device implementations. This profile is used for connecting application data *Source* devices such as blood pressure monitors, weight scales, glucose meters, thermometers, and pulse oximeters to application data *Sink* devices such as mobile phones, laptops, desktop computers, and health appliances without the need for cables. This profile makes use of the Multi-Channel Adaptation Protocol (MCAP) [2] and new L2CAP features such as Enhanced Retransmission Mode, Streaming Mode and optional FCS [1] to define the interoperability requirements.

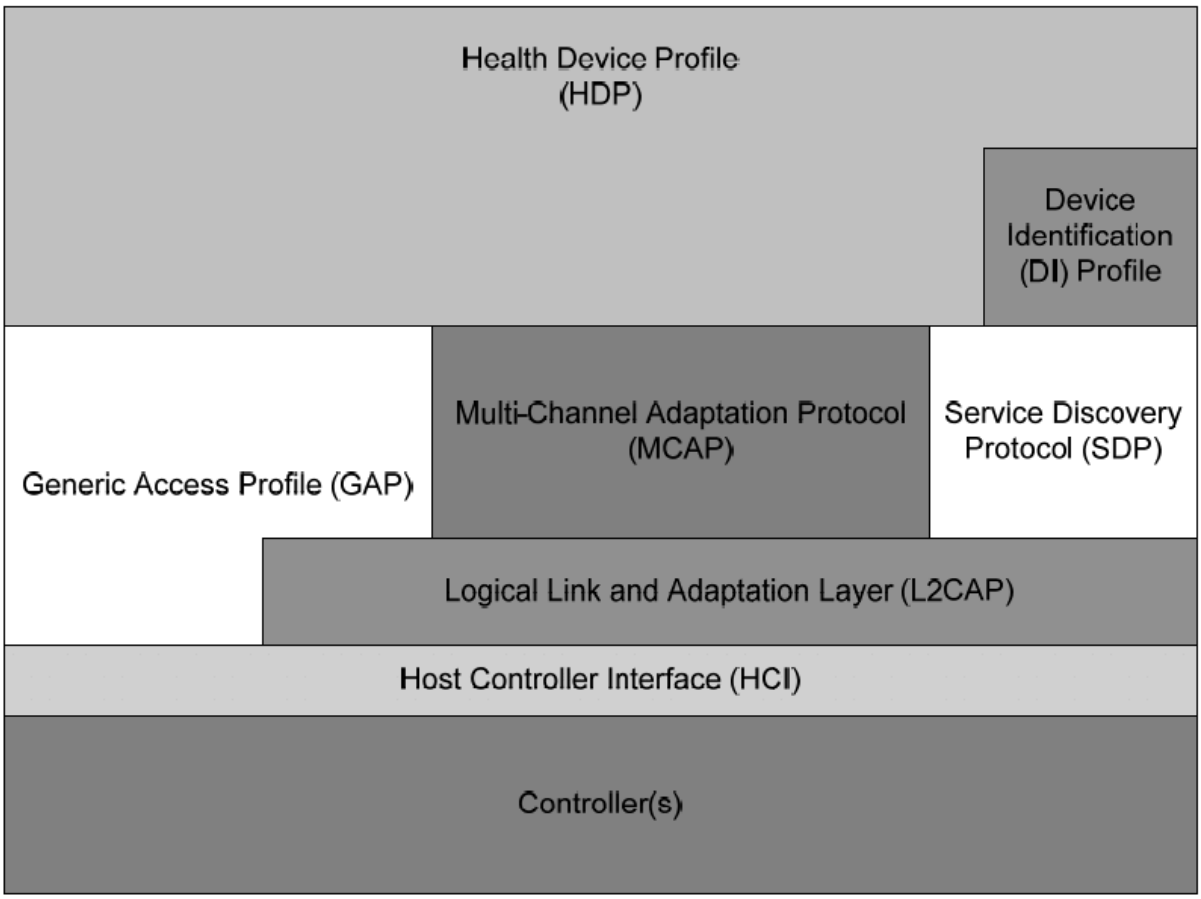


Figure 1: Protocol Stack

2.2 Sequence Overview

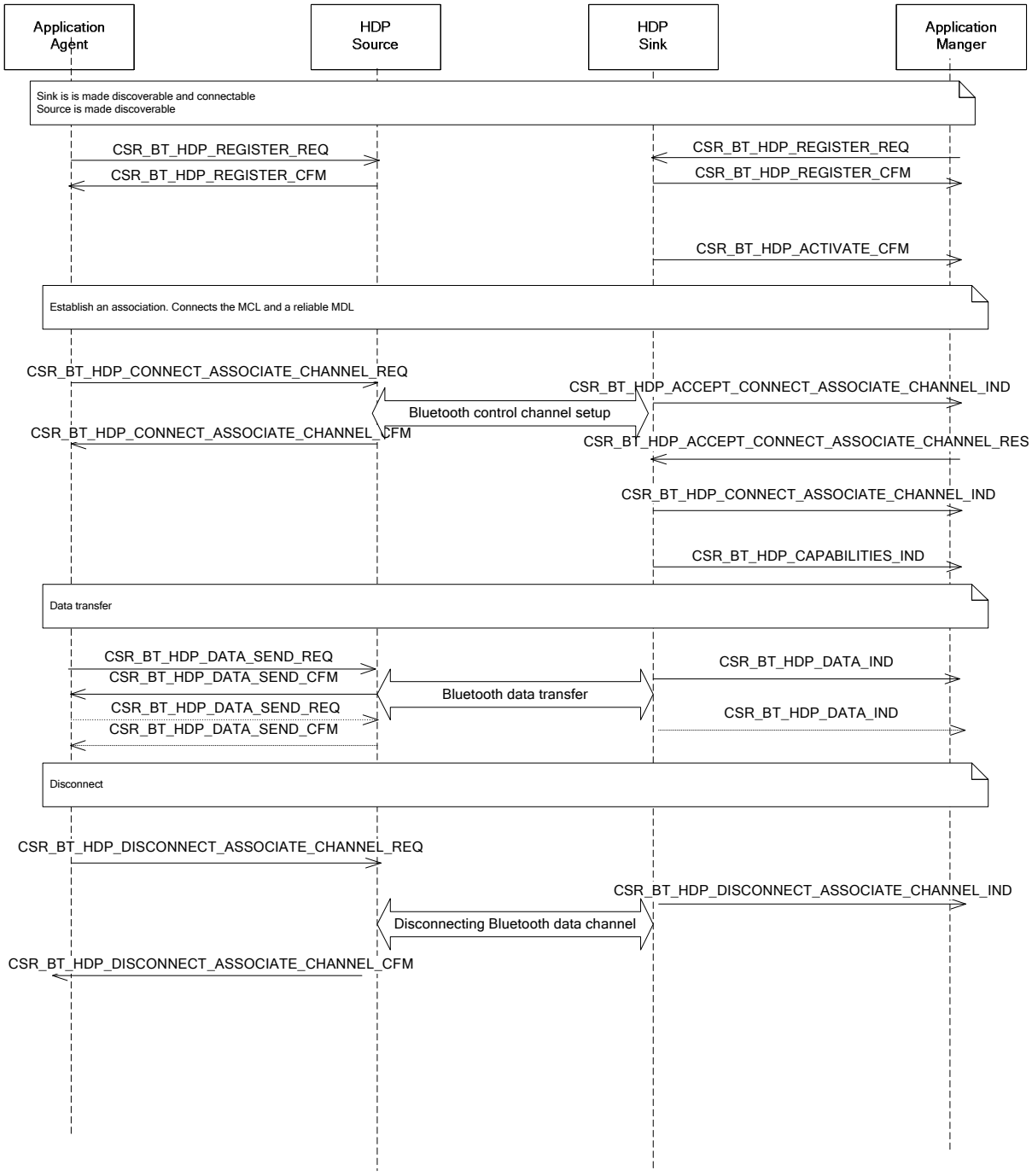


Figure 2 : HDP Agent Manager Scenario

3 Interface Description

In this section, a series of MSCs will be shown to explain the usage of the HDP profile. The primitives and the functions available to the application are also described in the subsections of this chapter.

3.1 Register

Sending a CSR_BT_HDP_REGISTER_REQ register's a service record in the Service Discovery Server.

The Supported Features attribute for a HDP service record requires extra definition depending on how the application wants to advertise health device modalities. These device modalities can be provided on or more end-points or multiplexed on a single endpoint. These end points are identified by a MDEPID. Hence a phase of IND/RSP transactions is initiated on receiving the REQ, to allow the profile to enumerate this information in the record.

On a successful registration with the Service discovery server, a CSR_BT_HDP_REGISTER_CFM is sent to the application.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Queue handle of the application
2	dm_security_level_t	secLevel	Security level of the device being activated
3	CsrCharString	serviceName	The ServiceName attribute is a string containing the name of the service represented by a service record.
4	CsrCharString	serviceDescription	String containing a brief description of the service. Less than 200 characters.
5	CsrCharString	providerName	String containing the name of the person or organization providing the service
6	CsrUint8	dataExchangeSpecification	Byte mask - Data Exchange Protocol
7	CsrTime	sniffTimeOut	Time (in ms) the MCL must be idle before requesting sniff mode for the connection. If set to 0x0000, MCAP will never request sniff mode for the connection.
8	CsrUint8	numOfMdep	Number of end points to advertise in the service record
9	CsrUint8	supportedProcedures	This is a one byte bit-mask that indicates the MCAP procedures that are to be supported by HDP service CSR_BT_HDP_SUPPORT_RECONNECT_INITIATION – Supports Reconnect Initiation 3 CSR_BT_HDP_SUPPORT_RECONNECT_ACCEPTANCE - Supports Reconnect Acceptance 4 CSR_BT_HDP_SUPPORT_CSP – Supports Clock Synchronization Protocol CSR_BT_HDP_SUPPORT_SYNC_MASTER_ROLE – Supports Sync-Master Role

Table 1 : Arguments for CsrBtHdpRegisterReqSend

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_REGISTER_IND
2	CsrSchedQid	appHandle	Queue handle of the application
3	CsrBtMdepId	mdepId	An identifier that identifies one or more logical functions advertised in the service record. It allows HDP to multiplex communication to various data layer end points. This is generated by HDP service layer. The response will indicate the functions the application wishes to advertise on this endpoint.

Table 2: Parameters in a CSR_BT_HDP_REGISTER_IND primitive

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Queue handle of the application
2	CsrBtMdepDataType	datatype	Identifies the Device Data Specialization advertised on this end point
3	CsrBtMdepRole	role	Iindicates whether this MDEP is a Source of the identified data type, or a Sink
4	CsrUtf8String	description	Description of the end-point
5	CsrBool	reuseMdepId	Indicates if the device data specialisation should be multiplexed on the MDEP-ID.

Table 3: Arguments for CsrBtHdpRegisterRspSend

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_REGISTER_CFM
2	CsrBtResultCode	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.
3	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 4: Parameters in a CSR_BT_HDP_REGISTER_CFM primitive

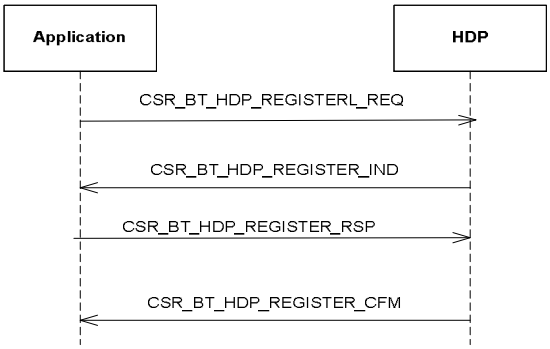


Figure 3 Registration Sequence

3.2 Unregister

Sending a CSR_BT_HDP_UNREGISTER_REQ will unregister the service record from the Service Discovery Server. Any ongoing connection will be disconnected.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Queue handle of the application

Table 5 : Arguments for CsrBtHdpUnRegisterReqSend

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_REGISTER_CFM

#	Type	Argument	Description
2	CsrBtResultCode	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.
3	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 6: Parameters in a CSR_BT_HDP_UNREGISTER_CFM primitive

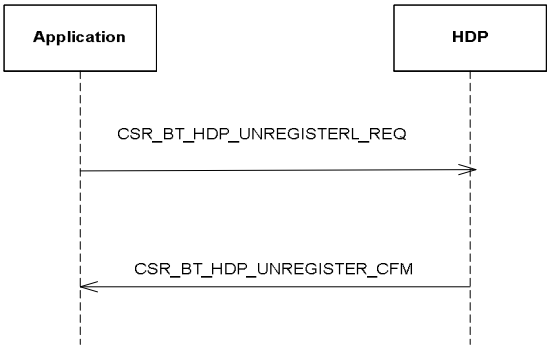


Figure 4 Unregister Sequence

3.3 Activate

For an application to be ready to accept an incoming HDP connection request, the HDP profile must be made connectable. Activation is done by sending the CSR_BT_HDP_ACTIVATE_REQ by calling CsrBtHdpActivateRqSend function.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Queue handle of the application
2	CsrUint8	noOfConnections	Number of simultaneous connections, which can be handled on the local PSM – from different Bluetooth devices.

Table 7 Arguments for CsrBtHdpActivateReqSend

On a successful activation, a CSR_BT_HDP_ACTIVATE_CFM is sent back to the application.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_ACTIVATE_CFM
2	CsrBtResultCode	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.
3	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 8: Parameters in a CSR_BT_HDP_ACTIVATE_CFM primitive

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.

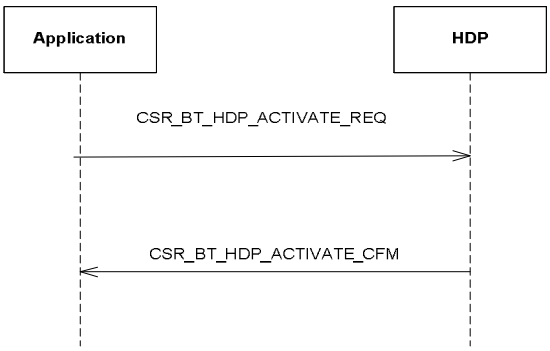


Figure 5 Activation Sequence

3.4 Deactivate

Sending the CSR_BT_HDP_DEACTIVATE_REQ by calling CsrBtHdpDeactivarteReqSend would deactivate HDP. Any transaction in progress/ongoing will be terminated immediately.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Queue handle of the application

Table 9 Arguments for CsrBtHdpActivateReqSend

On a successful activation, a CSR_BT_HDP_ACTIVATE_CFM is sent back to the application.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_DEACTIVATE_CFM
2	CsrBtResultCode	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.
3	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 10: Parameters in a CSR_BT_HDP_DEACTIVATE_CFM primitive

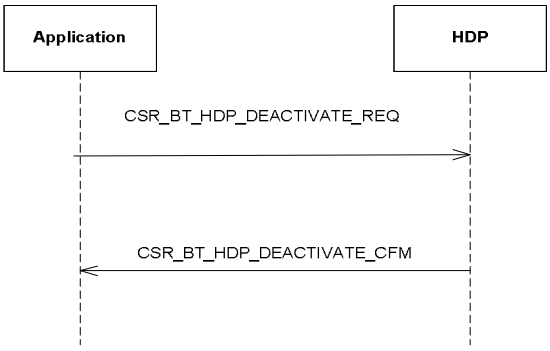


Figure 6 Deactivate Sequence

3.5 Get Capabilities

Before requesting a connection a HDP enabled device has to perform a search for the features supported by the peer. The search is based on the device specialization, device role. This is done by calling the CsrBtHdpControlGetCapablitiesReqSend function, which will send a CSR_BT_HDP_CTRL_GET_CAPABLITIES_REQ to the HDP profile. The arguments for the CsrBtHdpControlGetCapablitiesReqSend function are found in Table 14.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_HDP_CTRL_CONNECT_CFM is sent by the HDP profile
2	CsrBtDeviceAddr	deviceAddr	Bluetooth address of the peer device.
3	CsrUint8	mdepDataTypeMask	Bit Mask of the Device Specializations.
4	CsrUint32	mdepDataTypeConditionMask	A bit mask that specifies the Device specializations that have to be present.
5	CsrUint8	mdepRoleMask	Value that identifies the role (SOURCE or SINK or ANY ROLE) to be searched for.

Table 11: Arguments for CsrBtHdpCtrlGetCapabReqSend function

For each service record that matches the search criteria a CSR_BT_HDP_CTRL_GET_CAPABLITIES_IND is sent to the application. The parameters for the CSR_BT_HDP_CTRL_GET_CAPABLITIES_IND primitive are found in Table 17 : Parameters in a CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_CFM primitive

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_CTRL_GET_CAPABLITIES_CFM
2	CsrBtDeviceAddr	deviceAddr	Bluetooth address of the peer device.
3	CsrUint32	hdpInstanceId	The Instance Id is a unique Id generated for each connection. The higher 16 bits contain the Control PSM of the peer device and the lower 16 bits the Data PSM.
4	CsrCharString*	providerName	The ProviderName attribute is a string containing the name of the person or organization providing the service.
5	CsrCharString*	serviceName	The ServiceName is a string containing the name of the service represented by a service record.
6	CsrUint8	dataexchangeSpecification	This attribute is a one-byte reference, with the value taken from the Bluetooth Assigned Numbers [3] to identify the Data Exchange Protocol used.
7	CsrCharString*	serviceDescrip	The Service Description is a string containing a brief description of the service.
8	CsrUint8	supportedProcedures	This attribute is a one byte bit-mask that indicates the MCAP procedures that are supported by this HDP service.
9	CsrBtHdpMdep	supportedFeatureList	A list of MDEP endpoint ID and its associated features like MDEP Datatype, Role and Description.

Table 12: Parameters in a CSR_BT_HDP_GET_CAPABLITIES_IND primitive

The completion of the operation is notified to the application by CSR_BT_HDP_CTRL_GET_CAPABLITIES_CFM. The parameters for the CSR_BT_HDP_CTRL_GET_CAPABLITIES_CFM primitive are found in Table 13.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_CTRL_GET_CAPABLITIES_CFM

#	Type	Argument	Description
2	CsrBtResultCode	resultCode	The result code of the operation. Possible values depends on the value of resultSupplier. If eg. 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.
3	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
4	CsrBtDeviceAddr	deviceAddr	Bluetooth address of the peer device.

Table 13 : Parameters in a CSR_BT_HDP_GET_CAPABLITIES_CFM primitive

The get capabilities procedure is depicted in Figure 7.

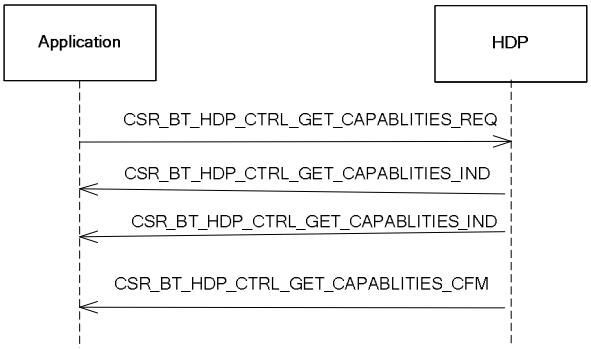


Figure 7: Get Capabilities Procedure

3.6 Establishing a Associate Channel Connection

When HDP enabled device wants to connect to another (registered and activated) HDP enabled device, it must first establish a Bluetooth connection. This is done by calling the CsrBtHdpConnectAssociateChannelReq`Send` function, which will send a CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_REQ to the HDP profile. This connect primitive would establish an MCL connection between the two devices. After a successful MCL creation the first reliable MDL connection is also established. The device must have registered and performed a Get Capabilities before sending this request. The arguments for the CsrBtHdpConnectAssociateChannel`ReqSend function` are found in Table 14 : Arguments for CrsBtHdpConnectAssociateChannelReq function.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_REQ is sent by the HDP profile
2	CsrUint32	hdpInstanceId	This ID is received after a successful Get Capabilities Operation. The higher 16 bits contain the Control PSM of the peer device and the lower 16 bits the Data PSM of the peer device.
3	CsrBtDeviceAddr	deviceAddr	Bluetooth address of the peer device.
4	CsrBtMdepId	mdepId	The endpoint to identify an MDEP (data-layer end point) of the remote HDP implementation.
5	CsrUint16	mpedDatatype	This attribute is a 16-bit value, with the value taken from the Bluetooth Assigned Numbers [3] to identify the Device Data Specialization code.
6	CsrUint8	mdepRole	MDEP Role shall indicate whether this MDEP is a *Source* of the identified data type, or a *Sink*.
	CsrUint16	maxPacketLength	Maximum transmission unit supported by the device for the connection.

Table 14 : Arguments for CrsBtHdpConnectAssociateChannelReq function

The creation of an MDL after successful MCL creation is notified to the application by CSR_BT_HDP_ACCEPT_CONNECT_ASSOCIATE_CHANNEL_IND. The application has the choice to accept or reject this connection. This is done by calling the CsrBtHdpAcceptConnectAssociateChannelResSend function, which will send a CSR_BT_HDP_ACCEPT_CONNECT_ASSOCIATE_CHANNEL_RES to the HDP profile. The parameters for the CSR_BT_HDP_ACCEPT_CONNECT_ASSOCIATE_CHANNEL_IND primitive are found in Table 18 : Parameters in the CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_IND primitive. The arguments for the CsrBtHdpAcceptConnectAssociateChannelResSend function are found in Table 16 : Arguments for CsrBtHdpAcceptConnectAssociateChannelResSend function.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_CTRL_GET_CAPABLITIES_CFM
2	CsrBtDeviceAddr	deviceAddr	Bluetooth address of the connecting device.
3	CsrBtMdepId	mdepId	The endpoint to identify an MDEP (data-layer end point) of the remote HDP implementation.
4	CsrUint32	assocChannelId	A 32 bit integer ID generated for each association whose lower 16 bits from the reliable MDL ID and the higher 16 bits the MCL ID of the corresponding connection.
5	CsrBtConnId	btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend-` function.

Table 15 : Parameters in a CSR_BT_HDP_ACCEPT_CONNECT_ASSOCIATE_CHANNEL_IND primitive

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Handle to the application.
2	CsrUint8	responseCode	Response from the application indicating acceptance or rejection of the connection.
3	l2ca_mtu_t	mtu	Maximum transmission unit supported by the device for the connection.
4	CsrUint32	assocChannelId	The association channel id that is received in the previous message.

Table 16 : Arguments for CsrBtHdpAcceptConnectAssociateChannelResSend function

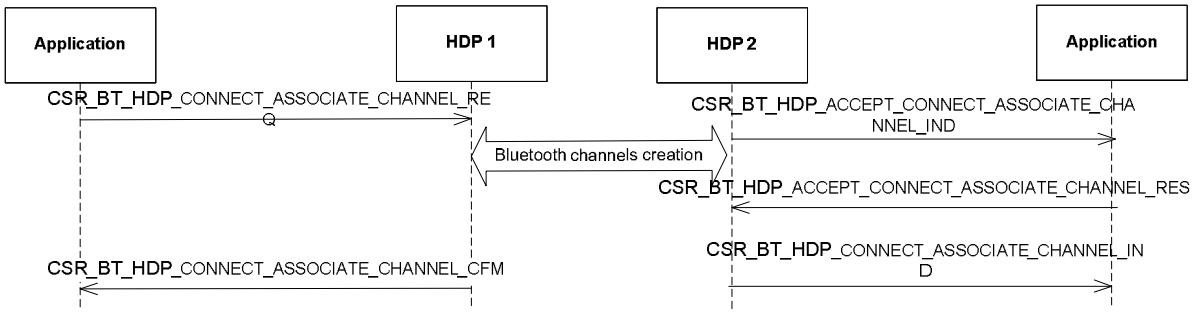


Figure 8 Association Connection Establishment

The completion of the operation at the initiating entity is indicated by CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_CFM signal as shown in the figure above. The parameters for the CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_CFM primitive are found in Table 17 : Parameters in a CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_CFM primitive. Similarly, at the peer end, the completion is indicated by a CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_IND signal. The parameters for the CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_IND primitive are found in Table 18 : Parameters in the CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_IND primitive

#	Type	Argument	Description

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_CONNECT_IND
2	CsrUint32	hdpInstanceId	A 32 bit Id identifying the PSM of the peer device. The higher 16 bits contain the Control PSM of the peer device and the lower 16 bits the Data PSM of the peer device.
3	CsrBtDeviceAddr	deviceAddr	Bluetooth address of the peer device.
4	CsrBtMdepId	mdepId	The endpoint to identify an MDEP (data-layer end point) of the remote HDP implementation.
5	CsrBtMdepDataType	dataType	Identifies the Device Data Specialization advertised on this end point
6	CsrUint32	assoChannelId	A 32 bit integer ID generated for each association whose lower 16 bits from the reliable MDL ID and the higher 16 bits the MCL ID of the corresponding connection.
7	CsrUint16	macPacketLength	Maximum transmission unit supported by the device for the connection.
8	CsrBtResultCode	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.
9	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
10	CsrBtConnId	btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend-` function.

Table 17 : Parameters in a CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_CFM primitive

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_CONNECT_IND
2	CsrUint32	hdpInstanceId	A 32 bit Id identifying the PSM of the peer device. The higher 16 bits contain the Control PSM of the peer device and the lower 16 bits the Data PSM of the peer device.
3	CsrBtDeviceAddr	deviceAddr	Bluetooth address of the peer device.
4	CsrBtmdepId	mdepId	The endpoint to identify an MDEP (data-layer end point) of the remote HDP implementation.
5	CsrBtMdepDataType	dataType	Identifies the Device Data Specialization advertised on this end point
6	CsrUint32	assocChannelId	A 32 bit integer ID generated for each association whose lower 16 bits from the reliable MDL ID and the higher 16 bits the MCL ID of the corresponding connection.
7	CsrUint16	maxPacketLenth	Maximum transmission unit supported by the device for the connection.
8	CsrBtResultCode	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.
9	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 18 : Parameters in the CSR_BT_HDP_CONNECT_ASSOCIATE_CHANNEL_IND primitive

3.7 Associate Channel Disconnect

The application can disconnect the Association channel using the CsrBtHdpDisconnectAssociateChannelReqSend function, which will send a CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_REQ to the HDP profile. This disconnect procedure will disconnect all the Virtual and Association Channels established between the two devices. The arguments for the CsrBtHdpDisconnectAssociateChannelReqSend `function` are found in Table 19.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_CFM is sent by the HDP profile
2	CsrUint32	assocChId	A 32 bit integer ID generated for each association whose lower 16 bits from the reliable MDL ID and the higher 16 bits the MCL ID of the corresponding connection.
3	CsrUint32	hdpInstanceId	A 32 bit Id identifying the PSM of the peer device. The higher 16 bits contain the Control PSM of the peer device and the lower 16 bits the Data PSM of the peer device.

Table 19 : Argument to CsrBtHdpDisconnectAssociateChannelReqSend function

The disconnect procedure is depicted in Figure 9.

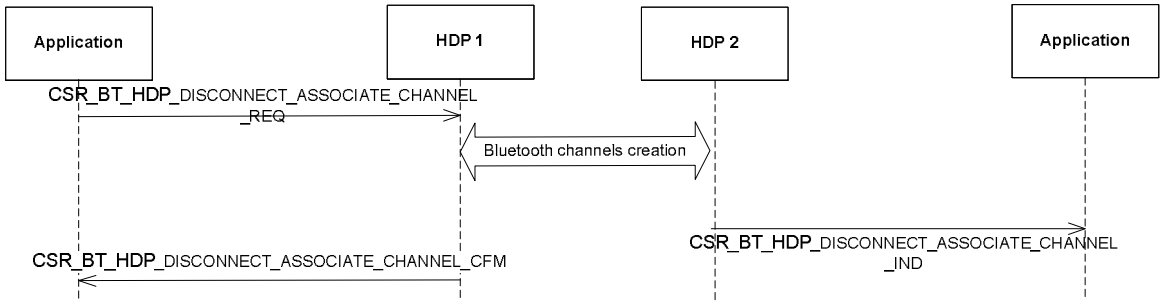


Figure 9 : Associate Channel Disconnect Procedure

After the disconnecting has finished, the HDP profile will send a CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_CFM to the application as depicted in Figure 9.

The disconnection of the channels is indicated to the application of the remote device by a CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_IND. The parameters for the CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_IND primitive is found in Table 20, while the parameters for the CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_CFM primitive is found in Table 21.

NOTE HDP will shut down the Bluetooth connections only if there is no IEEE activity between the two devices on the specified csr_bt_hdp_cid. If there is any activity a failure result code is sent back to the control application.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_IND
2	CsrUint32	assocId	A 32 bit Id identifying the association. It contains the MCL Id and the Id of the reliable MDL created for the association.
3	CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
4	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 20 : Parameters in a CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_IND primitive

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_CFM
2	CsrUint32	assocChId	A 32 bit Id identifying the association. It contains the MCL Id and the Id of the reliable MDL created for the association.
3	CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
4	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 21 : Parameters in a CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_CFM primitive

3.8 Virtual Channel Connect

In addition to the association channel, the application can open multiple virtual channels depending on the requirement of the device specialisation and the number of such logical functions enumerated on a particular endpoint.

The application establishes a virtual channel by calling CsrBtHdpConnectVirtualChannelReqSend function.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_CFM is sent by the HDP profile
2	CsrUint32	assocChannelId	Identifier of the association channel
3	CsrUint8	virtualChannelConfig	Configuration parameter for the channel
4	CsrUint16	maxPacketLength	Maximum transmission unit supported by the device for the connection.

Table 22 : Argument to CsrBtHdpConnectVirtualChannelReqSend function

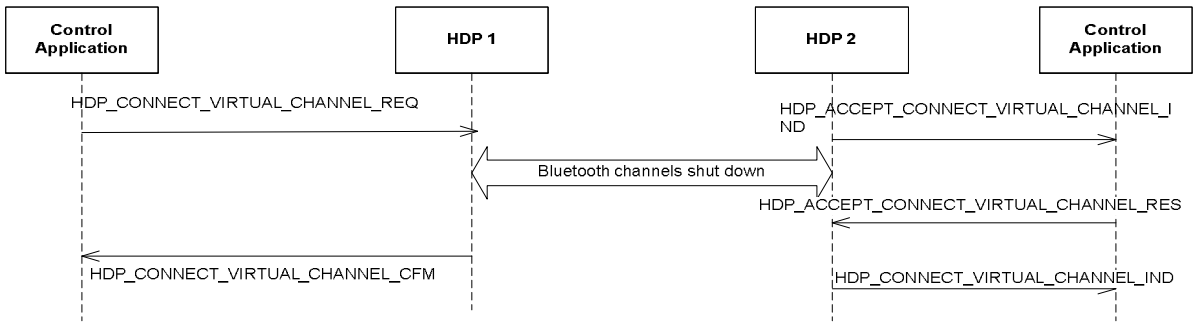


Figure 10 : Virtual Channel Connect Sequence

The peer entity shall receive an indication in the form of CSR_BT_HDP_ACCEPT_CONNECT_VIRTUAL_CHANNEL_IND. The entity can accept the incoming connection by CsrBtHdpAcceptConnectVirtualChResSend.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_ACCEPT_CONNECT_VIRTUAL_CHANNEL_IND

#	Type	Argument	Description
2	CsrUint32	assocChannelId	Identifier of the association channel
3	CsrUint32	virtualChannelId	Identifier of the virtual channel
4	CsrUint8	virtualChannelConfig	Configuration parameter for the channel
5	CsrBtConnId	btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend-` function.

Table 23 : Parameters in CSR_BT_HDP_ACCEPT_CONNECT_VIRTUAL_CHANNEL_IND primitive

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_HDP_DISCONNECT_ASSOCIATE_CHANNEL_CFM is sent by the HDP profile
2	CsrUint32	virtualChannelId	Identifier of the virtual channel
3	CsrUint8	virtualChannelConfig	Configuration parameter for the channel
4	CsrUint16	maxPacketLength	Maximum transmission unit supported by the device for the connection.
5	CsrUint8	responseCode	Result of the creation process. On success this parameter will equal CSR_BT_HDP_SUCCESS

Table 24 : Argument to CsrBtHdpAcceptConnectVirtualChannelResSend function

The initiating entity will receive a CSR_BT_HDP_CONNECT_VIRTUAL_CHANNEL_CFM and the peer will receive a CSR_BT_HDP_CONNECT_VIRTUAL_CHANNEL_IND.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_ACCEPT_CONNECT_VIRTUAL_CHANNEL_IND
2	CsrUint32	assocChannelId	Identifier of the association channel
3	CsrUint32	virtualChannelId	Identifier of the virtual channel
4	CsrUint16	maxPacketLen	Maximum transmission unit supported by the device for the connection.
5	CsrBtResultCode	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.
6	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
7	CsrBtConnId	btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend-` function.

Table 25 : Parameters in CSR_BT_HDP_ CONNECT_VIRTUAL_CHANNEL_CFM primitive

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_ACCEPT_CONNECT_VIRTUAL_CHANNEL_IND
2	CsrUint32	assocChannelId	Identifier of the association channel
3	CsrUint32	virtualChannelId	Identifier of the virtual channel
4	CsrUint16	maxPacketLen	Maximum transmission unit supported by the device for the connection.

#	Type	Argument	Description
5	CsrBtResultCode	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.
6	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
7	CsrBtConnId	btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend-` function.

Table 26 : Parameters in CSR_BT_HDP_ CONNECT_VIRTUAL_CHANNEL_IND primitive

3.9 Virtual Channel Disconnect

The application can disconnect a virtual data channel using the CsrBtHdpDisconnectVirtualChannelReqSend function, which will send a CSR_BT_HDP_DISCONNECT_VIRTUAL_CHANNEL_REQ to the HDP profile. This disconnect procedure will disconnect the corresponding data channel between the two devices. The arguments for the CsrBtHdpDisconnectVirtualChannelReqSend `function` are found in Table 25.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_HDP_VIRTUAL_CHANNEL_DISCONNECT_CFM is sent by the HDP profile
2	CsrUint32	virtualChId	A 32 bit Id identifying the secondary channel. It contains the MCL Id and the Id of the secondary MDL created for the association.

Table 27 : Argument to CsrBtHdpDisconnectVirtualChannelReqSend function

The disconnect procedure is depicted in Figure 10.

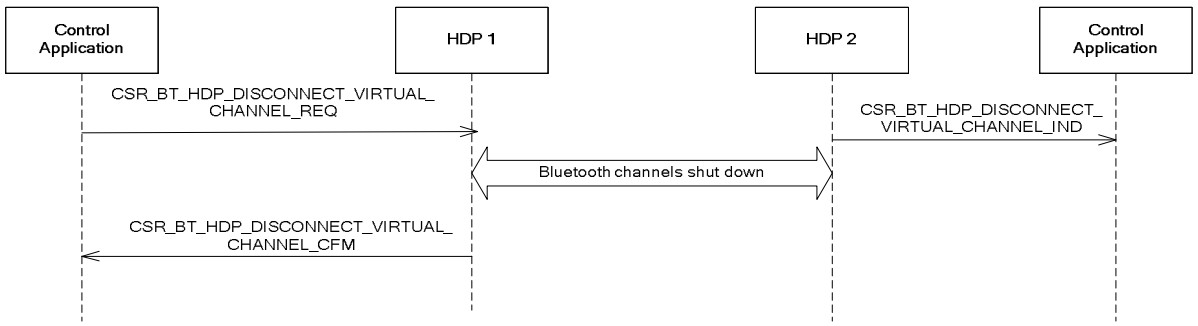


Figure 11 : Virtual Channel Disconnect Procedure

After the data channel has been disconnected, the HDP profile will send a CSR_BT_HDP_DISCONNECT_VIRTUAL_CHANNEL_CFM to the application as depicted in .

The disconnection of the channel is indicated to the application of the remote device by a CSR_BT_HDP_DISCONNECT_ VIRTUAL_CHANNEL_IND. The parameters for the CSR_BT_HDP_DISCONNECT_ VIRTUAL_CHANNEL_IND primitive is found in Table 28, while the parameters for the CSR_BT_HDP_DISCONNECT_ VIRTUAL_CHANNEL_CFM primitive is found in Table 29.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_DISCONNECT_VIRTUAL_CHANNEL_IND
2	CsrUint32	virtualChId	A 32 bit Id identifying the secondary channel. It contains the MCL Id and the Id of the secondary MDL created for the association.
3	CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
4	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 28 : Parameters in a CSR_BT_HDP_DISCONNECT_VIRTUAL_CHANNEL_IND primitive

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_DISCONNECT_VIRTUAL_CHANNEL_CFM
2	CsrUint32	virtualChId	A 32 bit Id identifying the secondary channel. It contains the MCL Id and the Id of the secondary MDL created for the association.
3	CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
4	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 29 : Parameters in a CSR_BT_HDP_DISCONNECT_ VIRTUAL _CHANNEL_CFM primitive

3.10 Sending data

When an MDL is connected, it is possible to send data to the remote device. To send data a CSR_BT_HDP_DATA_REQ must be sent to the MCAP profile by calling the `CsrBtHdpDataReqSend` function, which takes the arguments stated in Table 30 : Arguments for CsrBtHdpDataReqSend function

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Handle to application, i.e. the queue on which the CSR_BT_HDP_VIRTUAL_CHANNEL_DISCONNECT_CFM is sent by the HDP profile
2	CsrUint32	chId	A 32 bit Id that identifies the channel on which the data should be sent.
3	CsrUint16	dataLen	Length of the data in bytes
4	CsrUint8	*data	Pointer to payload data

Table 30 : Arguments for CsrBtHdpDataReqSend function

After receiving the CSR_BT_HDP_DATA_REQ, the HDP profile will locally generate a CSR_BT_HDP_DATA_CFM primitive and send it to the application. The parameters for this primitive are found in Table 32 : Parameters in a CSR_BT_HDP_DATA_IND primitive

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_DATA_CFM

#	Type	Argument	Description
2	CsrBtResultCode	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.
3	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 31 : Parameters in a CSR_BT_HDP_DATA_CFM primitive

The CSR_BT_HDP_DATA_REQ will appear in the remote device application as a CSR_BT_HDP_DATA_IND as depicted in Figure 12. The parameters of the CSR_BT_HDP_DATA_IND primitive are described in Table 32 : Parameters in a CSR_BT_HDP_DATA_IND primitive

NOTE that the CSR_BT_HDP_DATA_IND should not be answered.

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set to CSR_BT_HDP_DATA_SEND_IND
2	CsrUint16	length	Length of the data in bytes
3	CsrUint8	*data	Pointer to payload data

Table 32 : Parameters in a CSR_BT_HDP_DATA_IND primitive

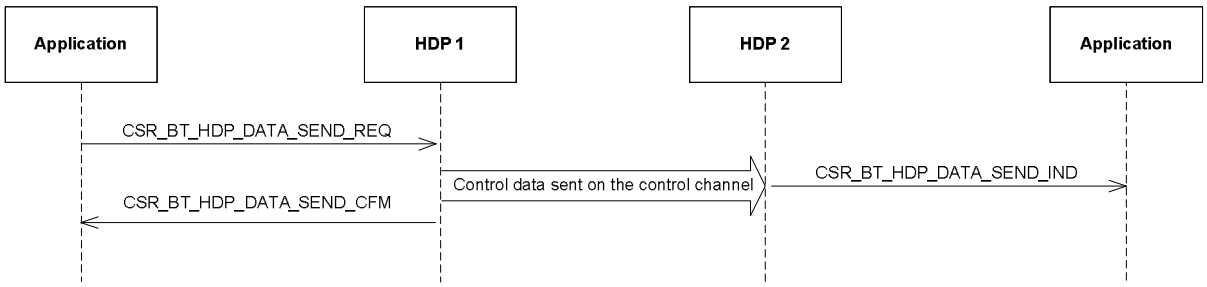


Figure 12 : Data send process

3.11 Suspend

Sending the CSR_BT_HDP_SUSPEND_REQ will suspend the Bluetooth connection. All MDLs pertaining to assocCHId will be disconnected. The MCL will be disconnected only if no other assocCh is active between the source and sink health device. The information on assocChId and virtualChId will however be saved to allow a reconnection later.

The success of the operation will be indicated to the application in the CSR_BT_HDP_SUSPEND_CFM.

The application can suspend the connection only if it has indicated that it supports reconnect. Refer CSR_BT_HDP_REGISTER_REQ.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Queue handle of the application
	CsrUint32	assocChId	Identifies the association channel

Table 33 : Arguments of CsrBtHdpSuspendReqSend

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set CSR_BT_HDP_SUSPEND_CFM
2	CsrUint32	assocChId	Identifies the association channel
3	CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
4	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 34 : Parameters in a CSR_BT_HDP_SUSPEND_CFM primitive

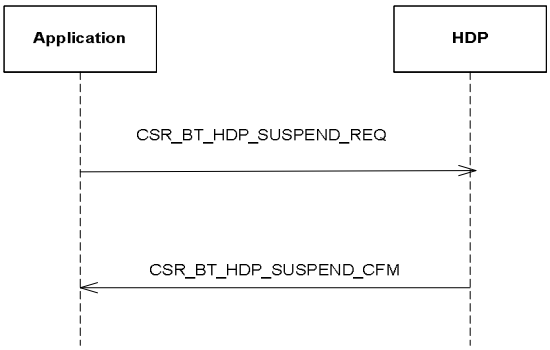


Figure 13 : Suspend Sequence

3.12 Resume

Sending the CSR_BT_HDP_RESUME_REQ will resume the Bluetooth connection. All MDLs pertaining to assocCHId will be reconnected. The MCL will be connected if no other assocChId is active between the source and the sink.

A CSR_BT_HDP_RESUME_IND would be sent to the peer application on the other end. The application will indicate if HDP can proceed with the resume in the CSR_BT_HDP_RESUME_RSP.

The success of the operation will be indicated to the application in the CSR_BT_HDP_RESUME_CFM.

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Queue handle of the application
2	CsrUint32	assocChId	Identifies the association channel

Table 35 : Arguments for CsrBtHdpResumeReqSend

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set CSR_BT_HDP_RESUME_IND
2	CsrUint32	chId	Identifies the association or virtual channel
3	CsrBtResultCode	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.

#	Type	Argument	Description
4	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 36 : Arguments for CSR_BT_HDP_RESUME_IND

#	Type	Argument	Description
1	CsrSchedQid	appHandle	Queue handle of the application
2	CsrUint32	chId	Identifies the association or virtual channel
3	l2ca_mtu_t	mtu	Maximum transmission unit supported by the device for the connection.
4	CsrBool	resume	Indicate TRUE if resume can proceed

Table 37 : Arguments for CsrBtHdpResumeResSend

#	Type	Argument	Description
1	CsrBtHdpPrim	type	Signal identity – always set CSR_BT_HDP_RESUME_IND
2	CsrUint32	assocChId	Identifies the association channel
3	CsrBtResultCode	resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
4	CsrBtResultSupplier	resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

Table 38 : Arguments for CSR_BT_HDP_RESUME_CFM

Note: After a resume is initiated, the identifier of the established association may change.

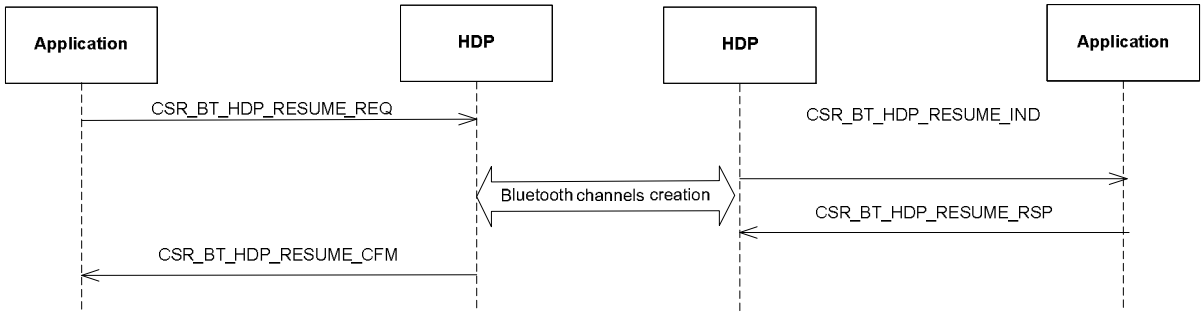


Figure 14 : Resume Sequence

4 Document References

Document	Reference
Bluetooth Core Specification Addendum 1, dated 26 June 2008. Available for download at www.bluetooth.org	[1]
ULTI-CHANNEL ADAPTATION PROTOCOL specification v. 10r00, dated 26 June. 2008. Available for download at www.bluetooth.org	[2]

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
HCI	Host Controller Interface
MCAP	Multi-Channel Adaptation Protocol
MCL	MCAP Communication Link
MDL	MCAP Data Link
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards

Document History

Revision	Date	History
1	26 SEP 11	Ready for release 18.2.0

TradeMarks, Patents and Licences

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

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

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

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

Life Support Policy and Use in Safety-critical Compliance

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

Performance and Conformance

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