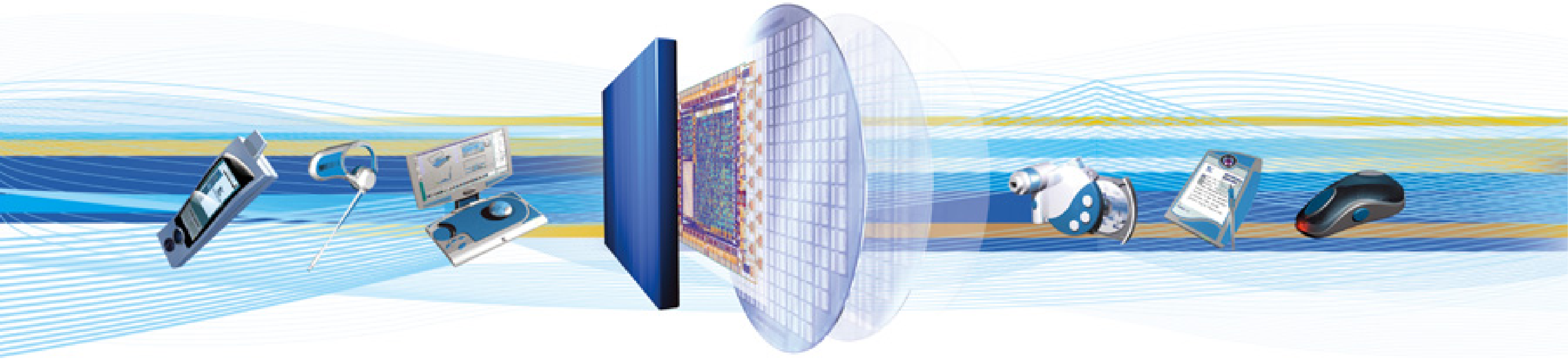
CSR Synergy®

CSR Synergy Bluetooth 18.2.0

DUN Gateway

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

Bluetooth®

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the gateway part of the Dial-Up Network profile as specified in [DUN]. The dialing and control functionality in DUN is **not** included in this component.

The dialing and control functionality in DUN is implemented in a separate component called the AT-Interpreter. The AT-Interpreter can be put on top of the DUN GW and all communication with the application is then going through the AT-Interpreter. The application will therefore see the AT-Interpreter as a Dial-Up Network gateway with AT-Interpreter functionality. If the AT-Interpreter functionality is needed, please refer to [AT DUN GW API].

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. DUN GW and the application
- Only one instance of the DUN GW is active at any time

2 Description

2.1 Introduction

The DUN profile manager supplies the interface for applications that should provide Dial-Up Networking functionality and conforms to the gateway part of the Dial-Up Network profile. The DUN profile is implemented as specified in the Dial-Up Network profile (K-7) [DUN]. The interface is in this document defined to be event based both in the upwards and downwards direction, but can be mapped to a function based interface.

The DUN profile layer provides functionality for:

- Establishing and maintaining a channel between the DUN profile manager and a terminal which conforms to the terminal part of the Dial-Up Network profile in part K-7 of [DUN]
- Sending and receiving data between terminal and gateway

2.2 Reference Model

The basic Dial-Up gateway functionality is based on the model outlined below in Figure 1. Figure 1 is the basic model applied where no AT-Command interpretation is included in the DUN profile manager part.

The application must interface to the DG, the Security Controller (SC) and the Connection Manager. The Connection Manager has an interface e.g. service activation, and the Security Controller has an interface e.g. for bonding functionality.

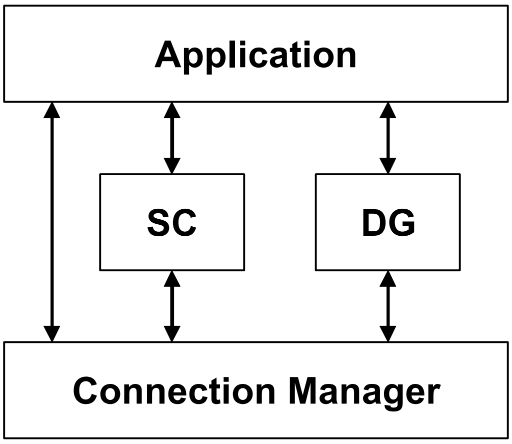


Figure 1: Basic DUN GW reference model

The AT-Command handling is left to the layer above the DUN profile manager as this may be included here already from legacy applications.

The SC is optional and only necessary if Security is enabled. There is no profile requirement to use Security in the DG. Security can be defined in the csr_bt_profiles.h file.

2.3 Communication Flow Architecture

The DG profile implementation optionally supports divided communication channels (or paths): A Bluetooth® management signal flow and a data management flow. The communication flow architecture is depicted in Figure 2.

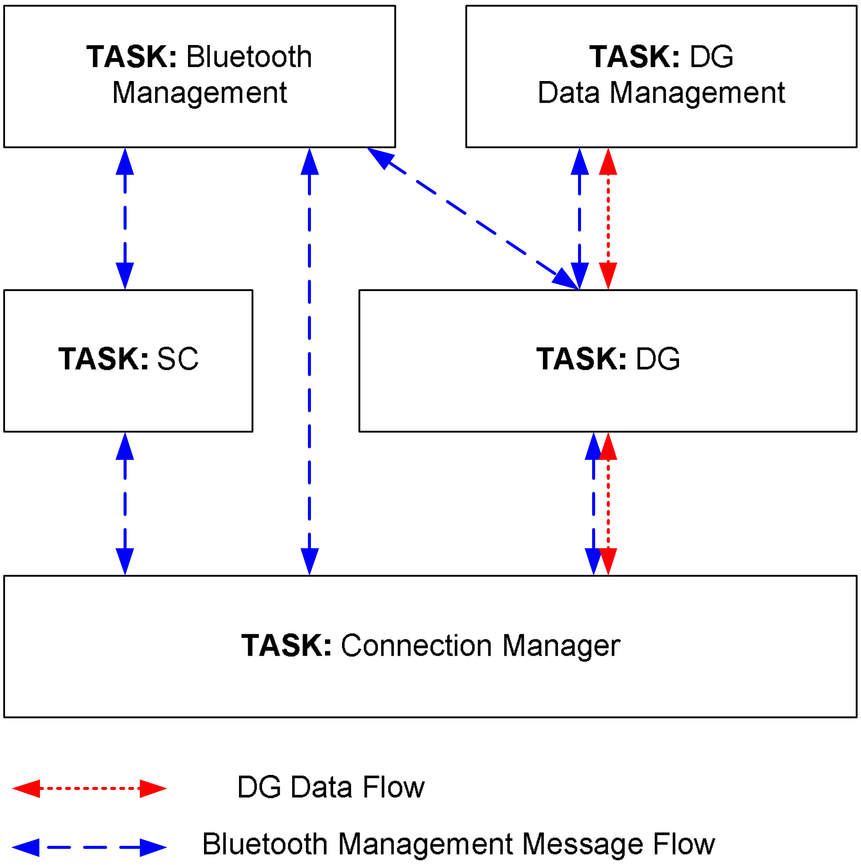


Figure 2: Communication Flow Architecture

The blue arrows depict the communication flow necessary for managing the Bluetooth® connection, whereas the red arrows depict the communication flow to be considered as DG data.

This architecture implies that two application tasks must be registered in DG when using its functionality[1]. The idea of supporting a Bluetooth® management flow and a DG data flow enables more flexibility in the implementation of the application layer above the SPP profile.

Registration of the *Bluetooth Management* task is done when requesting a connection establishment. The *Bluetooth Management* task will receive all messages (both management and data path messages) until a *SPP Data Management* task is registered. When the *SPP Data Management* task is registered, all messages related to the SPP data will be forwarded to the SPP *Data Management* task instead of the *Bluetooth Management* task.

An example of an application layer utilising the more flexible structure could be a separate application for controlling establishment of the Bluetooth® connection. Another separate application could be a device driver appearing as a serial port device in the Operating System handling the internet connection establishment and forwarding of the received data to the IP stack of the OS.

In the interface and primitive descriptions it will be described explicitly whether the primitives are used for the Bluetooth® Management message flow or the *SPP Data Management* message flow.

[1] It can be the same task that is registered for the Bluetooth connection management task and the DUN-DT data task.

2.4 Sequence Overview

A normal scenario is that the DUN gateway, usually a phone, will pair with a terminal, and hence create a bond between the pair of devices, in advance using the DUN functionality.

However, if this is not the case, the security manager will, depending on the security level, request the passkey from the application before accepting the connection. The bonding procedure is described in [SC].

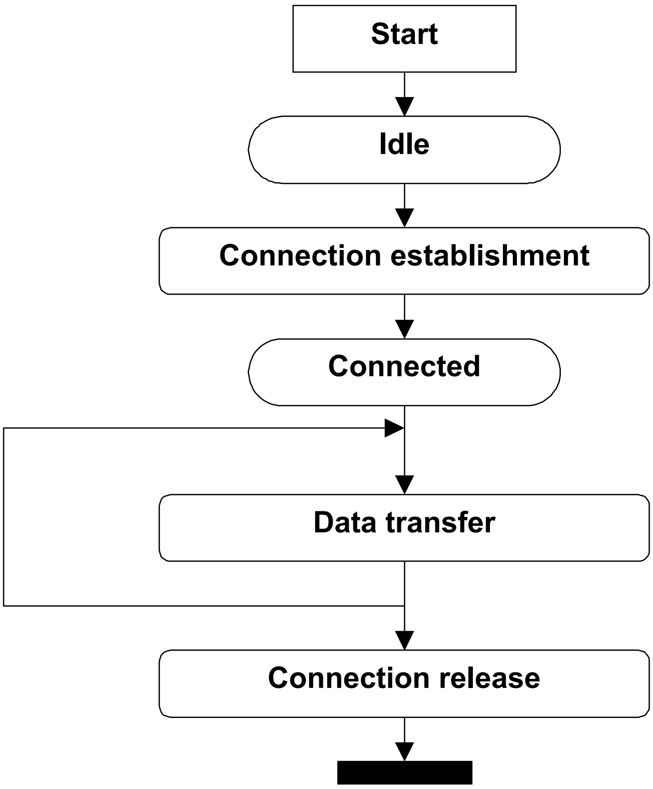


Figure 3: Sequence overview

According to [DUN], the Bluetooth® connection is initiated from the terminal side; this is described in the connection establishment (section 3.1.1). Upon connection completion, data may be exchanged between the two parties.

The data may include AT-Commands with control information. The interpretation of the AT-Command set may be done in either the existing application or in the AT-Handler, which is an optional component that can be added.

3 Interface Description

3.1 Connection Management

3.1.1 Connection Establishment

Connections are established from the terminal side, typically a laptop or PDA. The DUN profile manager and the local device must be set in a mode where incoming connections are accepted. This is accomplished by issuing the CSR_BT_DG_ACTIVATE_REQ from the application towards the DUN profile manager. After sending the CSR_BT_DG_ACTIVATE_REQ the local device is connectable.

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 DUN profile manager remains in the active state until the service is Deactivate from the application or the time expires.

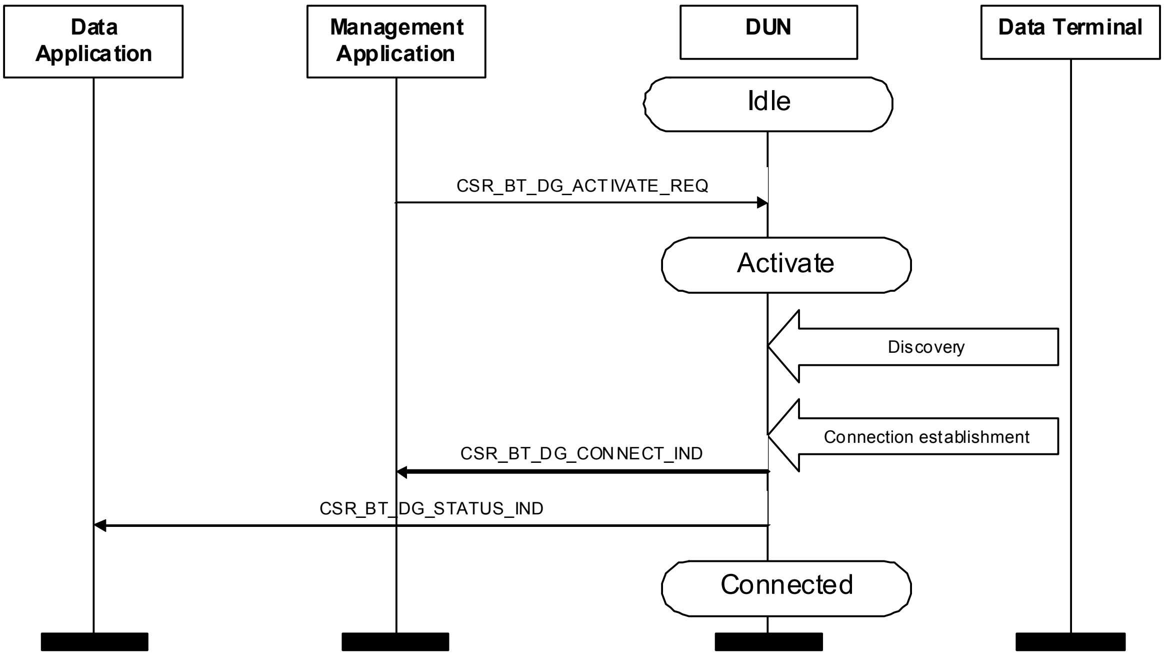


Figure 4: Connection establishment sequence

Before the DUN GW will accept incoming connections a bond between the two devices must exist. Is this not the case the SC will ensure that the passkey is requested from the application, see [SC]. The CSR_BT_DG_CONNECT_IND is an indication that the connection is established, authenticated and encryption is enabled for any subsequent data transfer. Furthermore, a CSR_BT_DG_STATUS_IND is sent to the data flow with the *connect* parameter set to TRUE to indicate that the connection has been successfully established.

3.1.2 Service Deactivation

If the application decides that the DUN service should no longer be connectable, the application may send a CSR_BT_DG_DEACTIVATE_REQ and get a CSR_BT_DG_DEACTIVATE_CFM return when deactivated. This is possible both in activate and connected state.

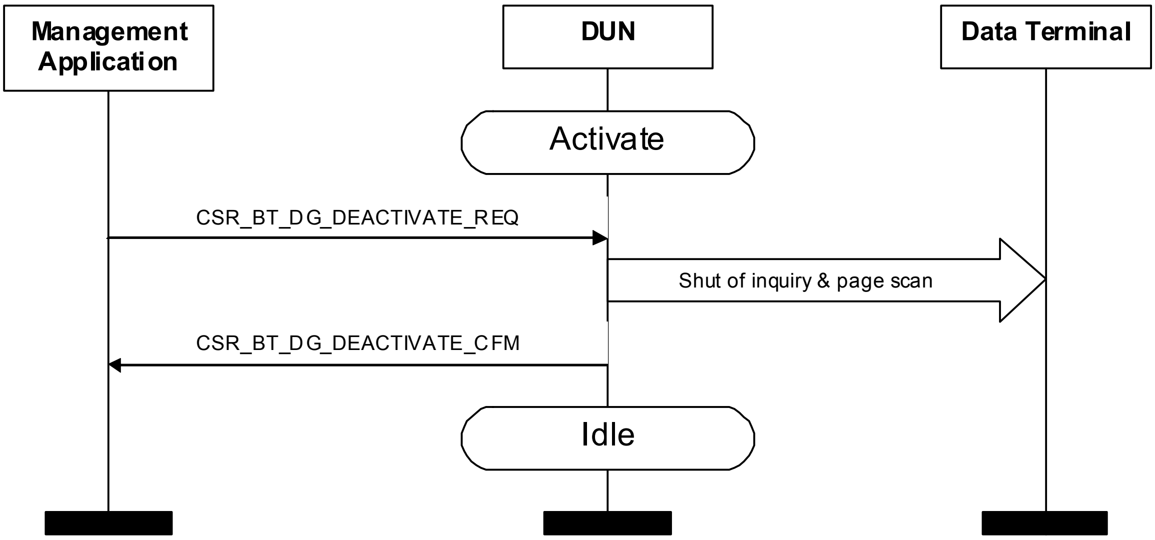


Figure 5: Connection establishment sequence

3.2 Connection Release

3.2.1 Normal Connection Release

The application layer may decide to release the established connection at any time. Release is done by sending a CSR_BT_DG_DISCONNECT_REQ to the DUN profile manager.

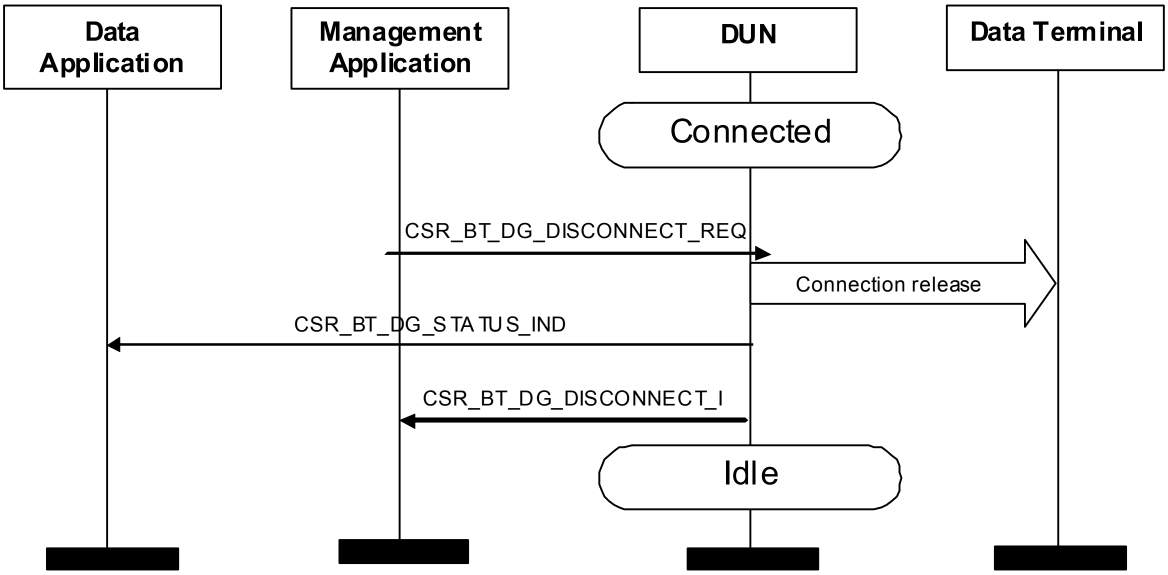


Figure 6: Normal connection release sequence

Depending on the actual state of the connections between the local and remote device, the physical link may be released as a consequence of the disconnection.

3.2.2 Abnormal Connection Release

The connected terminal at the remote end may at any time release the connection. Further, 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.

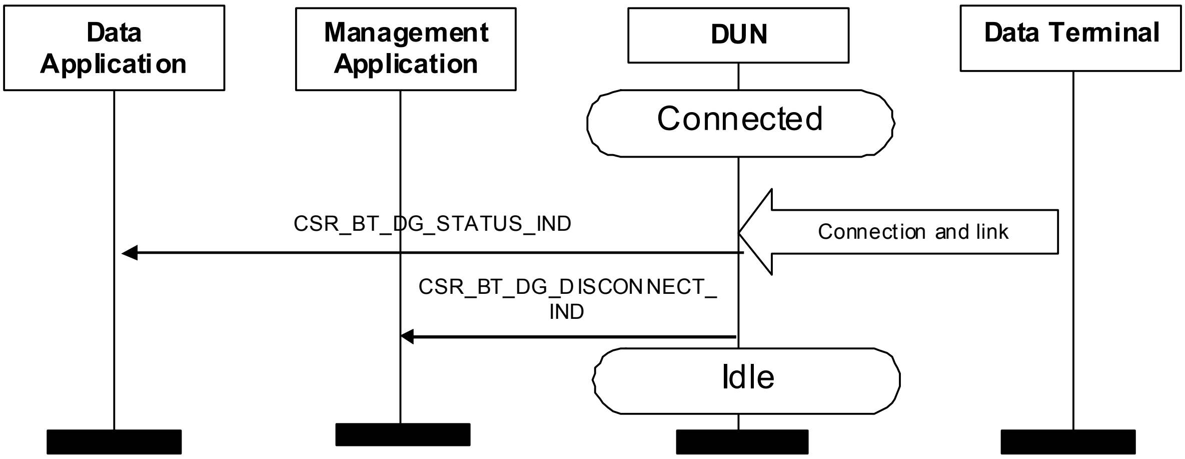


Figure 7: Abnormal connection release sequence

The DUN profile manager will handle the disconnect signal in a similar manner no matter the reason for the release. A new CSR_BT_DG_ACTIVATE_REQ may be issued to make the service available again. When the connection has been released, a CSR_BT_DG_STATUS_IND with the *connect* parameter set to false will be sent to the data flow.

3.3 Data Transfer

3.3.1 Upstream Data

Data can be received from the peer Bluetooth® device and is forwarded to the application. Received data is forwarded using the CSR_BT_DG_DATA_IND signal with a reference to the data payload.

Due to efficiency reasons it is preferred to apply credit based flow control between the local and remote device when data is transmitted.

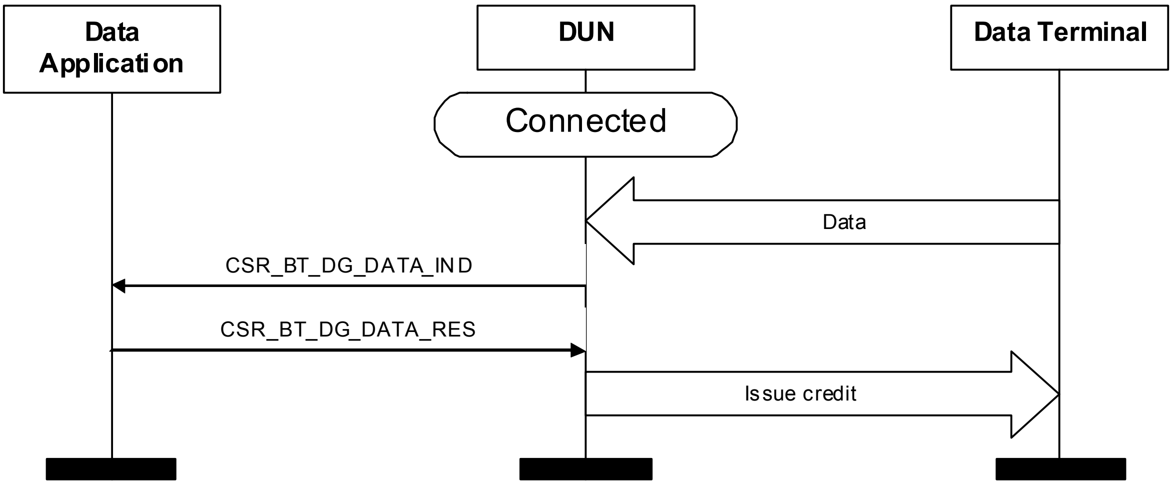


Figure 8: Upstream data transfer

Using credit based flow control implies that the Bluetooth® stack needs to issue credits back to the data terminal side when the received data is consumed and more data can be received. The CSR_BT_DG_DATA_RES must be used by the application layer to inform the DUN profile manager that data is handled and that the application layer is able to receive more data payload. Thus, as long as the CSR_BT_DG_DATA_RES has not been sent to the DUN profile manager no more data packets will be sent towards the application layer. It is the responsibility of the application layer to release the data payload when found appropriate by the application layer.

Once the CSR_BT_DG_DATA_RES is sent the DUN profile manager will ensure that a credit information package is issued to the data terminal side.

3.3.2 Downstream Data

Using the CSR_BT_DG_DATA_REQ signal the application layer can send data payload to the remote device. The CSR_BT_DG_DATA_REQ must include a reference to the data payload to be issued.

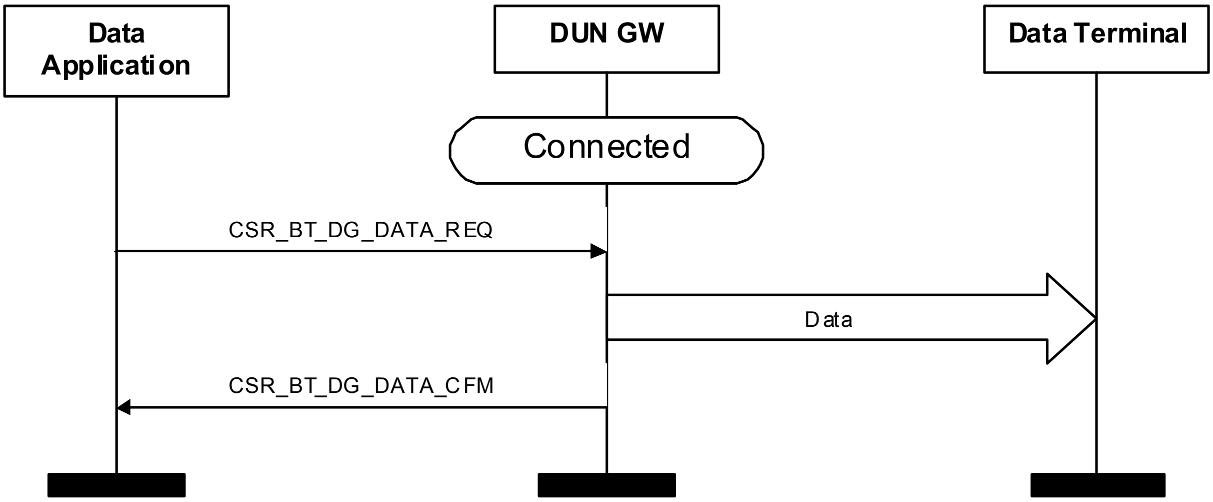


Figure 9: Downstream data

The DUN profile manager will forward the received data to the lower layers of the Bluetooth® stack for transmission to the terminal side. Once the data is sent to the lower layers a CSR_BT_DG_DATA_CFM is issued to the application layer. Receiving the CSR_BT_DG_DATA_CFM does not necessarily imply that the data has been received on the terminal side but only that the profile manager is capable of receiving more data.

Please note that the application layer is only allowed to send one data signal, i.e. one CSR_BT_DG_DATA_REQ without receiving a CSR_BT_DG_DATA_CFM and that the profile manager is responsible for the data payload reference; i.e. the application layer must not reuse or release the data payload memory.

3.4 Modem Control Signal

The modem line status can be controlled and read from the application.

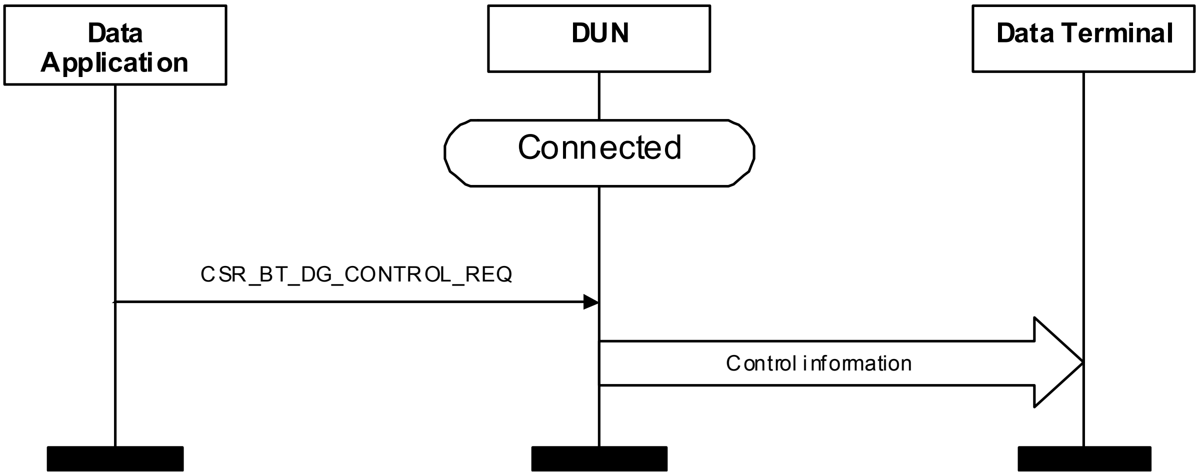


Figure 10: Set the modem status

Furthermore, it is possible to read state of the remote device.

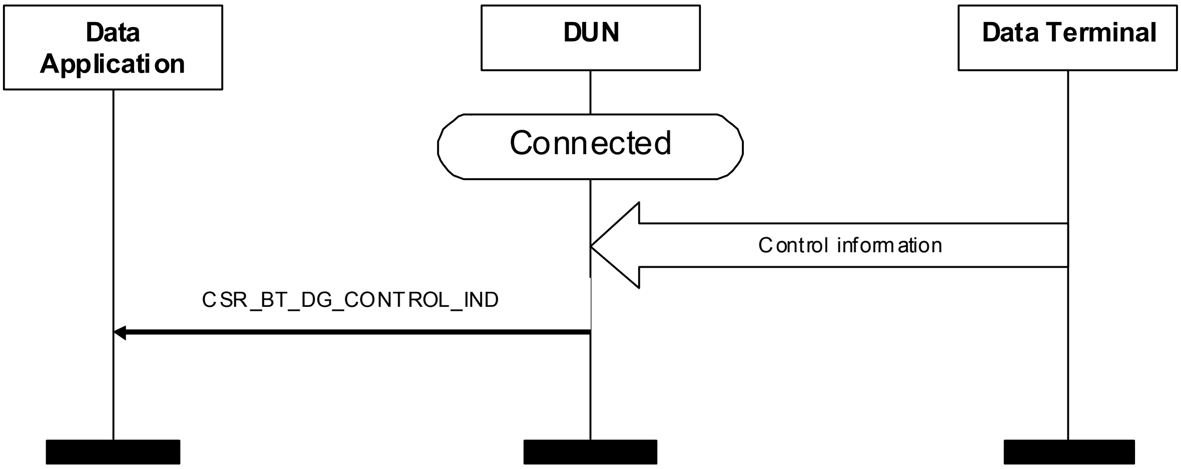


Figure 11: Read the modem status

3.5 Port Negotiation Signal

The port negotiate signal from the remote device has requested it to set port parameters for the server channel as given in the CSR_BT_DG_PORTNEG_IND signal. The port parameter consists of the port data rate, number of data bits, number of stop bits, parity and parity type.

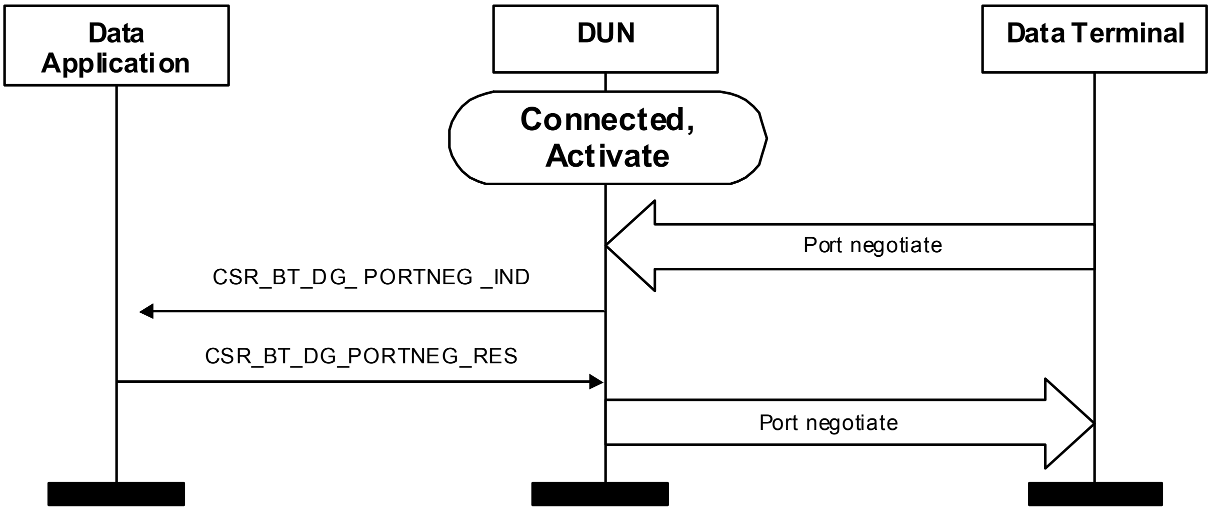


Figure 12: Response to a port negotiate indication

4 DUN Primitives

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

4.1 List of All Primitives

Primitives	Reference
CSR_BT_DG_ACTIVATE_REQ	See section 4.2
CSR_BT_DG_DEACTIVATE_REQ	See section 4.3
CSR_BT_DG_DEACTIVATE_CFM	See section 4.3
CSR_BT_DG_CONNECT_IND	See section 4.4
CSR_BT_DG_DATA_REQ	See section 4.5
CSR_BT_DG_DATA_CFM	See section 4.5
CSR_BT_DG_DATA_IND	See section 4.5
CSR_BT_DG_DATA_RES	See section 4.5
CSR_BT_DG_CONTROL_REQ	See section 4.6
CSR_BT_DG_CONTROL_IND	See section 4.6
CSR_BT_DG_PORTNEG_IND	See section 4.7
CSR_BT_DG_PORTNEG_RES	See section 4.7
CSR_BT_DG_DISCONNECT_REQ	See section 4.8
CSR_BT_DG_DISCONNECT_IND	See section 4.8
CSR_BT_DG_REGISTER_DATA_PATH_HANDLE_REQ	See section 4.9
CSR_BT_DG_REGISTER_DATA_PATH_HANDLE_CFM	See section 4.9
CSR_BT_DG_DATA_PATH_STATUS_REQ	See section 4.10
CSR_BT_DG_DATA_PATH_STATUS_IND	See section 4.10
CSR_BT_DG_STATUS _IND	See section 4.11
CSR_BT_DG_SECURITY_IN_REQ	See section 4.12
CSR_BT_DG_SECURITY_IN_CFM	See section 4.12

Table 1: List of all primitives

4.2 CSR_BT_DG_ACTIVATE

Parameters Primitives	type	phandle	timeout	role
CSR_BT_DG_ACTIVATE_REQ	✓	✓	✓	✓

Table 2: CSR_BT_DG_ACTIVATE Primitive

Description

This signal is used for activating a service and make it connectable. The process includes:

- Registration of the DUN gateway service in the service discovery database
- Enabling page scan

The service availability can be time limited or may run forever. See also section 4.9.

Parameters

type — Signal identity, CSR_BT_DG_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.

timeout — The time, in number of seconds, where the service should be available for remote connections. The maximum Timeout time allowed is defined by the CSR_BT_MAX_TIME in the file “csr_bt_profiles.h”. A timeout value of 0 (zero) or the define CSR_BT_INFINITE_TIME in the file “csr_bt_profiles.h” indicates an infinity timeout time. When a connection is made, the service available stops.

role — Decides if DG role is DTC or DCE.

4.3 CSR_BT_DG_DEACTIVATE

Parameters / Primitives	type	resultCode	resultSupplier
CSR_BT_DG_DEACTIVATE_REQ	✓		
CSR_BT_DG_DEACTIVATE_CFM	✓	✓	✓

Table 3: CSR_BT_DG_DEACTIVATE Primitives

Description

Deactivate a service that has been activated previously. The service will no longer be connectable.

Parameters

type Signal identity, CSR_BT_DG_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

4.4 CSR_BT_DG_CONNECT

Parameters / Primitives	type	deviceAddr	btConnId	serverChannel	profileMaxFrameSize	resultCode	resultSupplier
CSR_BT_DG_CONNECT_IND	✓	✓	✓	✓	✓	✓	✓

Table 4: CSR_BT_DG_CONNECT Primitives

Description

A connection has been established.

Parameters

type	Signal identity, CSR_BT_DG_CONNECT_IND.
deviceAddr	The Bluetooth® device address to which a connection must be/or is established.
serverChannel	The local server number, which is a reference ID used by the Connection Manager.
profileMaxFrameSize	The maximum payload length allowed in downstream data requests. Please note that it is possible to receive one octet (byte) more than indicated in the profileMaxFrameSize parameter.
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	Connection Identifier

4.5 CSR_BT_DG_DATA

Parameters / Primitives	type	btConnId	payloadLength	*payload
CSR_BT_DG_DATA_REQ	✓	✓	✓	✓
CSR_BT_DG_DATA_CFM	✓	✓		
CSR_BT_DG_DATA_IND	✓	✓	✓	✓
CSR_BT_DG_DATA_RES	✓	✓		

Table 5: CSR_BT_DG_DATA Primitives

Description

Send and receive data.

Parameters

type	Signal identity, CSR_BT_DG_DATA_REQ/CFM/IND/RES.
btConnId	Connection Identifier
payloadLength	Length of data in number of octets.
*payload	Pointer reference to the actual payload.

4.6 CSR_BT_DG_CONTROL

Primitives \ Parameters	type	btConnId	modemStatus	break_signal
CSR_BT_DG_CONTROL_REQ	✓	✓	✓	✓
CSR_BT_DG_CONTROL_IND	✓	✓	✓	✓

Table 6: CSR_BT_DG_CONTROL Primitives

Description

Send and receive modem status information.

Parameters

type Signal identity, CSR_BT_DG_CONTROL_REQ/IND.

btConnId Connection Id

modemStatus The modem status contains the following bit from the RS-232 interface:

Bit 0	CTS (Clear to Send)
Bit 1	RTS (Request to Send)
Bit 2	DSR (Data Set Ready)
Bit 3	DTR (Data terminal Ready)
Bit 4	RI (Ring Indicator)
Bit 5	DCD (Data carrier Detect)

There is mask code for this bit in the csr_bt_profiles.h.

break_signal The `break_signal` is encoded as follows:

Bit 0	Not used.
Bit 1	0: No break signal encoded. 1: Break signal encoded.
Bit 2	Not used.
Bit 3	Not used.
Bit 4-7	Duration of break signal in 200mS increments.

4.7 CSR_BT_DG_PORTNEG

Parameters Primitives	type	btConnId	portPar	request
CSR_BT_DG_PORTNEG_REQ	✓	✓	✓	
CSR_BT_DG_PORTNEG_CFM	✓	✓	✓	
CSR_BT_DG_PORTNEG_IND	✓	✓	✓	✓
CSR_BT_DG_PORTNEG_RES	✓	✓	✓	

Table 7: CSR_BT_DG_PORTNEG Primitives

Description

Send and receive port set-up information.

Parameters

type	Signal identity, CSR_BT_DG_PORTNEG_ IND/RES.
btConnId	Connection Identifier
portPar	The portPar is a structure defined as RFC_PORTNEG_VALUES_T. The RFC_PORTNEG_VALUES_T structure, shown below, is included into the PORTNEG primitive. In the library function call the RFC_PORTNEG_VALUES_T structure should be called as a pointer and the library function will copy the data into the PORTNEG primitive.
Request	TRUE: If the request is TRUE, the remote device requests the local RFC_PORTNEG_VALUES_T settings. The receiver must respond with current RFC_PORTNEG_VALUES_T settings. FALSE: The remote device must confirm the settings or propose new ones. Setting new suggested values includes setting also the proper parameter mask.

```
typedef struct
{
   CsrUint8         baud_rate;   /* port speed indicator - see #defines above */
   CsrUint8         data_bits;   /* DATA_BITS_5, _6, _7 or _8 - see above */
   CsrUint8         stop_bits;   /* STOP_BITS_ONE or _ONE_AND_A_HALF - see above */
   CsrUint8         parity;      /* PARITY_OFF or PARITY_ON */
   CsrUint8         parity_type; /* PARITY_TYPE_ODD, _EVEN, _MARK or _SPACE */
   CsrUint8         flow_ctrl_mask;/* 6 bits - use FLC_ #defines above (see 07.10)*/
   CsrUint8         xon;         /* xon character  (default DC1 0x11) */
   CsrUint8         xoff;        /* xoff character (default DC3 0x13) */
   CsrUint16        parameter_mask;  /* 16 bits (top two reserved) see PM_
                   #defines */
} RFC_PORTNEG_VALUES_T;
```

baud_rate	Takes the form RFC_xxxx_BAUD, where `xx` is the port speed in bits per second.

	Encoded as: 0x00 = 2400 0x01 = 4800 0x02 = 7200 0x03 = 9600 0x04 = 19200 0x05 = 38400 0x06 = 57600 0x07 = 115200 0x08 = 230400 0xFF = RFC_UNKNOWN_BAUD
data_bits	Number of data bits encoded as an unsigned integer. Valid values are 5, 6, 7 and 8. Encoded as: 0x00 = 5 bit 0x02 = 6 bit 0x01 = 7 bit 0x03 = 8 bit
stop_bits	Encoded as: 0x00 = 1 stop bit 0x01 = 1.5 stop bits
Parity	Encoded as: 0x00 = PARITY_OFF 0x01 = PARITY_ON
parity_type	Encoded as: 0x00 odd parity 0x02 even parity 0x01 mark parity 0x03 space parity
flow_ctrl_mask	Encoded as: Bit 0 XON / XOFF, input Bit 1 XON / XOFF, output Bit 2 RTR input Bit 3 RTR output Bit 4 RTC input Bit 5 RTC output
xon	Xon character (default DC1, 0x11)
xoff	Xoff character (default DC3, 0x13)
parameter_mask	`parameter_mask` is used for indicating which parameters in the Remote Port Negotiation command is negotiate able. For a command, `parameter_mask` is interpreted as follows: 0 no change 1 change For a response, `parameter_mask` is interpreted as follows: 0 not accepted / not supported 1 accepted proposal - the new values are used

	The bit mask is shown as: Bit0 bit rate Bit1 data bits Bit2 stop bits Bit3 Parity Bit4 parity type Bit5 XON character Bit6 XOFF character Bit7 Reserved Bit8 XON / XOFF, input Bit9 XON / XOFF, output Bit10 RTR input Bit11 RTR output Bit12 RTC input Bit13 RTC output Bit14 Reserved, set 0 by sender Bit15 Reserved, set 0 by sender

4.8 CSR_BT_DG_DISCONNECT

Primitives \ Parameters	type	btConnId	deviceAddr	localTerminated	reasonCode	reasonSupplier
CSR_BT_DG_DISCONNECT_REQ	✓	✓				
CSR_BT_DG_DISCONNECT_IND	✓	✓	✓	✓	✓	✓

Table 8: CSR_BT_DG_DISCONNECT Primitives

Description

Release a connection between the local and remote device. Depending on the number of connections and other active services, the link may or may not be released.

Parameters

type	Signal identity, CSR_BT_DG_DISCONNECT_REQ/IND.
btConnId	Connection Identifier
deviceAddr	The Bluetooth® device address to which a connection must be/or is established.
localTerminated	TRUE if termination of connection happened on request from the local host; FALSE otherwise.
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

4.9 CSR_BT_DG_REGISTER_DATA_PATH_HANDLE

Parameters Primitives	type	dataAppHandle	resultCode	resultSupplier
CSR_BT_DG_REGISTER_DATA_PATH_HANDLE_REQ	✓	✓		
CSR_BT_DG_REGISTER_DATA_PATH_HANDLE_CFM	✓		✓	✓

Table 9: CSR_BT_DG_REGISTER_DATA_PATH_HANDLE Primitives

Description

Register special application handle to which all future data-related messages will be sent to instead of the standard application handle. Data related messages are defined as: CSR_BT_DG_DATA, CSR_BT_DG_CONTROL, CSR_BT_DG_PORTNEG.

Parameters

type Signal identity, CSR_BT_DG_REGISTER_DATA_PATH_HANDLE_REQ/CFM.

dataAppHandle Application handle to receive future data messages.

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

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

4.10 CSR_BT_DG_DATA_PATH_STATUS

Parameters Primitives	type	dgInstanceQueue	status	resultCode	resultSupplier
CSR_BT_DG_DATA_PATH_STATUS_REQ	✓	✓	✓		
CSR_BT_DG_DATA_PATH_STATUS_IND	✓	✓	✓	✓	✓

Table 10: CSR_BT_DG_DATA_PATH_STATUS Primitives

Description

When a data path has been registered using the CSR_BT_DG_REGISTER_DATA_PATH_HANDLE signals (see section 4.9), the data application may need to notify the regular application of changes with regard to data availability. These availability states are “open”, “closed” and “lost”.

The data application request the change to be sent (the REQ), and the DG then forwards the signal to the regular application (the IND).

Parameters

type	Signal identity, CSR_BT_DG_DATA_PATH_STATUS_REQ/IND.
dgInstanceQueue	Handle of DG for which the data path is changing status.
status	Data path status code. See csr_bt_profiles.h.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.11 CSR_BT_DG_STATUS

Parameters / Primitives	type	btConnId	deviceAddr	connect	maxMsgSize
CSR_BT_DG _STATUS_IND	✓	✓	✓	✓	✓

Table 11: CSR_BT_DG_STATUS Primitive

Description

When connections are established or released, a status indication is sent to the data path, such that a data application may prepare to receive data.

Note that the CSR_BT_DG_STATUS_IND signal is always sent *after* a CSR_BT_DG_CONNECT signal and always *before* a CSR_BT_DG_DISCONNECT.

Also note that the CSR_BT_DG_STATUS_IND signal will be sent only to the data path flow. If a data path handle is not registered, or the data and control handle are equal, the signal is not sent.

Parameters

type	Signal identity, CSR_BT_DG_STATUS_IND.
btConnId	Connection Identifier
deviceAddr	Bluetooth address of remote device which has been connected (zero otherwise).
connect	True if connection established, false if released.
maxMsgSize	Maximum frame size in a CSR_BT_DG_DATA_REQ/IND message.

4.12 CSR_BT_DG_SECURITY_IN

Parameters Primitives	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_DG_SECURITY_IN_REQ	✓	✓	✓		
CSR_BT_DG_SECURITY_IN_CFM	✓			✓	✓

Table 12: CSR_BT_DG_SECURITY_IN_REQ 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_DG_SECURITY_IN_REQ* signal sets up the security level for new incoming connections. Already established or pending connections are not altered.

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 API] for further details.

Parameters

type — Signal identity CSR_BT_DG_SECURITY_IN_REQ/CFM.

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

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

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

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

- CSR_BT_SEC_AUTHORISATION: Require authorisation
- CSR_BT_SEC_AUTHENTICATION: Require authentication
- CSR_BT_SEC_ SEC_ENCRYPTION: Require encryption (implies
- authentication)
- CSR_BT_SEC_MITM: Require MITM protection (implies encryption)

resultCode — The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are

	currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

5 Document References

Document	Reference
Bluetooth® Core Specification v.1.1, v.1.2 and v.2.0, section N/A	[BT]
The Bluetooth® Specification, the Dial-Up Network Profile. Section K:7	[DUN]
CSR Profile Layer, Security Controller Interface Description. Section N/A	[SC]
api-0105-dg_at.pdf (part of CSR Synergy Bluetooth user documentation)	[AT DUN GW API]
api-0102-sc.pdf (part of CSR Synergy Bluetooth user documentation)	[SC API]
CSR Synergy Bluetooth. CM – Connection Manager API Description, doc. no. api-0101-cm.pdf	[CM]

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
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards
DG	Dial-up networking Gateway. The gateway part of the dial-up networking profile as specified in [DUN].
SIG	Special Interest Group.
Data Terminal	This is the device that uses the dial-up service of the gateway. Typically this is a PC, laptop or PDA.

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.