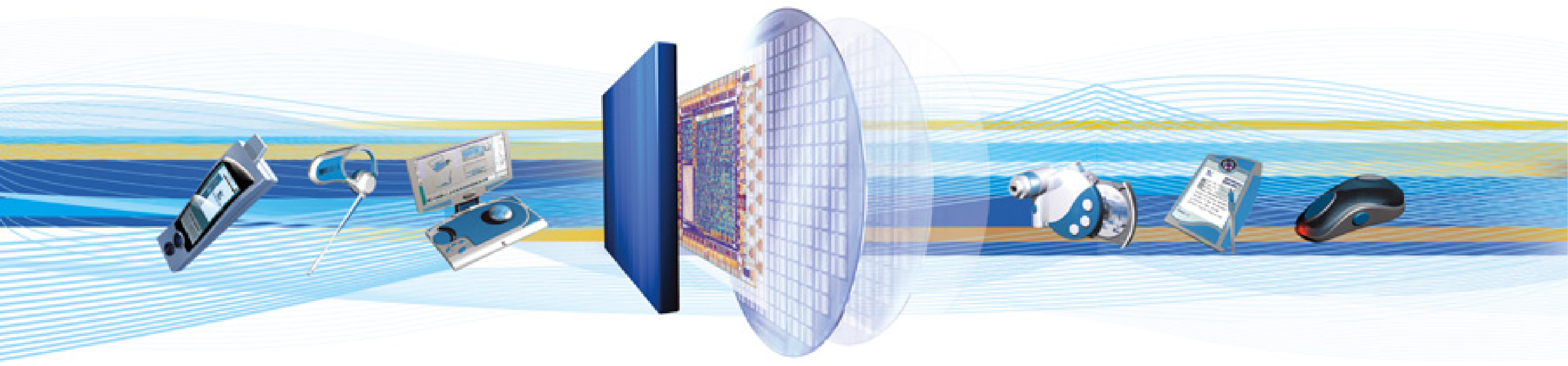
CSR Synergy Bluetooth 18.2.0

SAPC – SIM Access Client Profile

API Description

November 2011



Cambridge Silicon Radio Limited

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

Registered in England and Wales 3665875

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

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the SIM Access Profile Client (SAPC). The SAPC conforms to the client side of the SIM Access Profile, ref. [SAPSPEC].

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. SAPC and the application

It is assumed that the user has knowledge of the SIM Access Profile specification.

2 Description

2.1 Introduction

The SAPC profile covers the following scenario:

- Access to and control of a remote SIM card using Bluetooth®, i.e. possibility for e.g. a car phone to use the SIM card of the driver's private mobile phone

SAPC accesses the files and services of the SIM card transparent for the user, i.e. as the SIM card is directly connected to the client via a cable. The responsibility of SAPC is to supply the application with an interface offering the necessary functionality to provide this transparent SIM card access. More specifically, SAPC supplies the following services to the application:

- Connection handling (establishment and termination of a SIM Access connection)
- Transfer of command APDUs and reception of response APDUs
- Accessing the remote SIM card, i.e.:
    - Power SIM card off
    - Power SIM card on
    - Reset SIM card
    - Transfer ATRs
    - Transfer card reader status
    - Set Transfer Protocol

The application is responsible for sending requests to SAPC and handles the corresponding upcoming responses.

NOTE: According to [SAPSPEC] this profile must use an encryption key length of minimum 64 bits. Hence the Bluetooth® chip must support at least 64 bits encryption.

2.2 Reference Model

The SAPC interfaces with the Connection Manager (CM), which handles the connection established to the SIM Access Profile Server, as depicted in Figure 1.

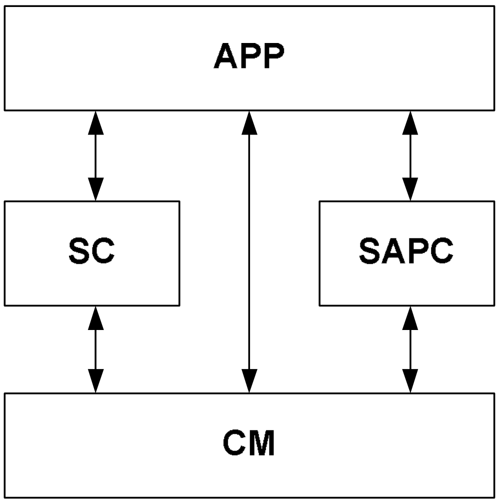


Figure 1: Reference model

The application interfaces SAPC in order to establish a connection to the remote SIM card and to control it. Furthermore, the application interfaces the Security Controller (SC) and the CM, because no functionalities for pairing and discovery are implemented in the SAPC. The SC is utilized during the pairing with the SIM Access Profile Server (SAPS) Bluetooth® device, whereas the CM is interfaced when making a discovery for the SAPS Bluetooth® device.

2.3 Sequence Overview

Figure 2 outlines the basic functionality of the SAPC by the means of a High-level Message Sequence Chart (HMSC). The related message sequence charts can be seen in section Interface Description.

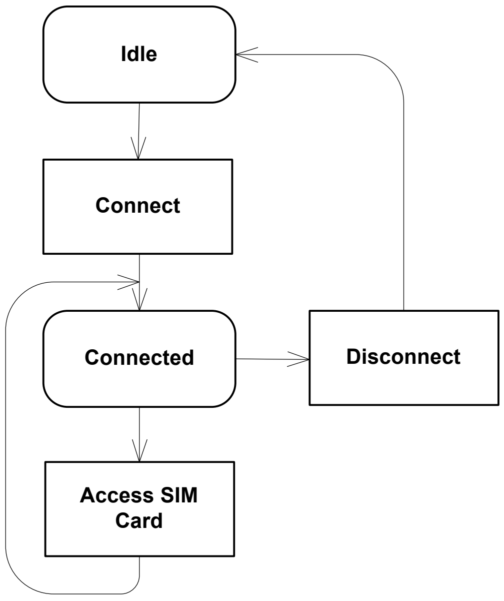


Figure 2: Sequence overview

The diagram outlines a normal scenario where a SIM Access connection is established to the SAPS.

NOTE: The shown sequence overview requires that the Bluetooth® device address is known, which can be found by performing a discovery. Discovery is described in [CM]. Furthermore, pairing is necessary for the server and client to connect, which is described in [SC].

Initially, SAPC is in the idle state waiting for the application to request a connection. Once being connected, the SIM Card can be accessed through various primitives, which are described in section SIM Access Profile Client Primitives. The application or server can initiate a disconnection, where after SAPC will return to the Idle state.

3 Interface Description

This section outlines the MSCs from Figure 2 (the rectangular boxes), and thereby describing how the application interfaces and utilizes the SAPC. The primitives used in this section are more thoroughly described in section SIM Access Profile Client Primitives.

3.1 Connect

Before accessing the remote SIM card a connection to the SAPS must be established. Figure 3 shows the connection establishment phase.

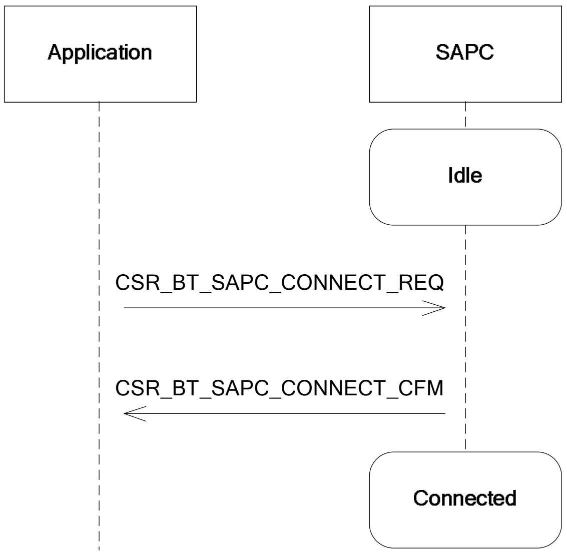


Figure 3: SIM Access connection establishment

A connection establishment is always initiated by the application at the client side and the CSR_BT_SAPC_CONNECT_REQ and CSR_BT_SAPC_CONNECT_CFM primitives are used during this phase, which are described more thoroughly in section 4.2. Initially, the application sends a CSR_BT_SAPC_CONNECT_REQ containing the server's Bluetooth® device address and the maximum message size supported by the application.

The maximum message size to be used between SAPC and SAPS are negotiated between the SAPS and SAPC. If they agree upon a message size, it is included in the CSR_BT_SAPC_CONNECT_CFM and the *result* field is set to CSR_BT_OK_CONNECT. If the message size proposed by the application is too small, the application receives a CSR_BT_SAPC_CONNECT_CFM primitive with CSR_BT_MSG_SIZE_TO_SMALL in the *result* field, hence no connection is established. The possible values for the *result* field can be seen in csr_bt_sap_common.h and csr_bt_profiles.h.

3.2 Access SIM Card

Accessing the SIM card can done in various sequences according to the purpose with the access. This section describes the different ways to access the SIM card.

3.2.1 Transfer APDU

Transferral of APDUs implies two primitives, CSR_BT_SAPC_TRANSFER_APDU_REQ and CSR_BT_SAPC_TRANSFER_APDU_CFM. Figure 4 shows the usage of the two parameters.

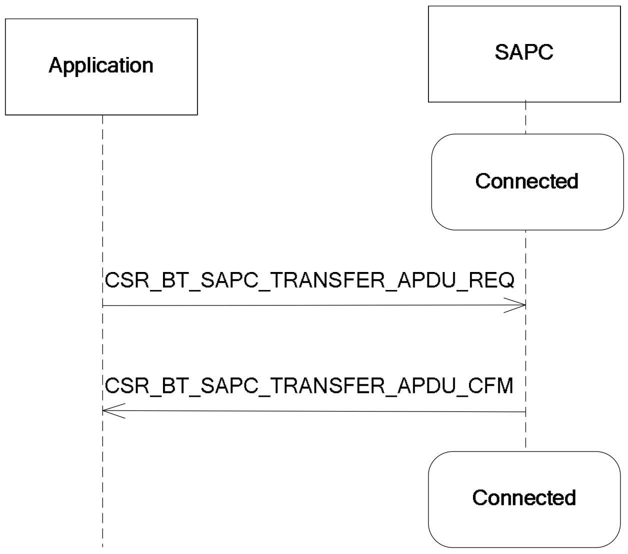


Figure 4: Transferring an APDU

APDU transfers are always initiated by the application by sending a CSR_BT_SAPC_TRANSFER_APDU_REQ to the SAPC. The CSR_BT_SAPC_TRANSFER_APDU_REQ contains a pointer to a command APDU (as defined in GSM 11.11 or GSM 11.14). The memory allocated for this command APDU is released by the underlying layers when sent, hence this is not within the responsibility of the application. When the command APDU is executed at the server, a CSR_BT_SAPC_TRANSFER_APDU_CFM is sent to the application. If a successful execution was performed, the CSR_BT_SAPC_TRANSFER_APDU_CFM will contain a CSR_BT_RSLT_OK_REQUEST in the *resultCode*. Furthermore, a pointer to the response APDU is contained in the CSR_BT_SAPC_TRANSFER_APDU_CFM. It is the responsibility of the application to free the memory allocated for the pointer, when no longer being used. If an error occurred when executing the CSR_BT_SAPC_TRANSFER_APDU_REQ, the *resultCode* in the received CSR_BT_SAPC_TRANSFER_APDU_CFM will contain an appropriate result code, and no response APDU will be contained. A list of the possible values for *resultCode* can be seen in csr_bt_sap_common.h.

3.2.2 Transfer ATR

For the application at the client side to request the ATR from the SIM card, two primitives are used, namely CSR_BT_SAPC_TRANSFER_ATR_REQ and CSR_BT_SAPS_TRANSFER_ATR_CFM. The use of the two primitives is shown in Figure 5.

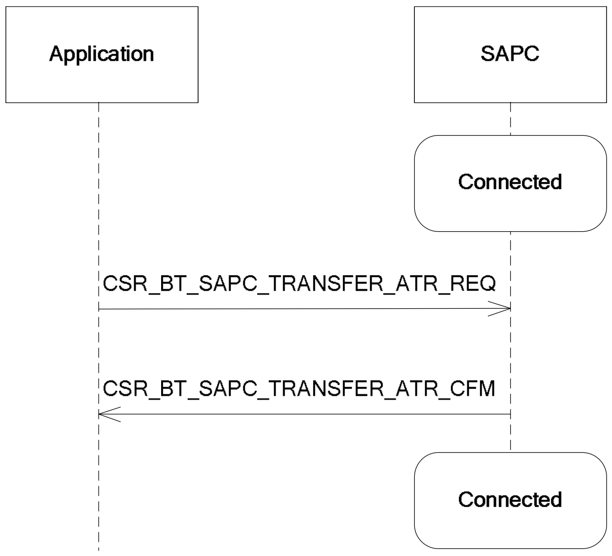


Figure 5: Transferring an ATR

The application sends a CSR_BT_SAPC_TRANSFER_ATR_REQ to the SAPC. When the ATR request is processed at the server, the SAPC replies the application by sending a CSR_BT_SAPC_TRANSFER_ATR_CFM primitive. This primitive contains a *resultCode*, which indicates the result of the request. If CSR_BT_RSLT_OK_REQUEST is received, the ATR request was processed successfully and a pointer to the actual ATR is also contained in the CSR_BT_SAPC_TRANSFER_ATR_CFM. It is the responsibility of the application to release the memory allocated for the ATR. If the ATR request was not processed successfully an error code is included in the *resultCode* of the CSR_BT_SAPC_TRANSFER_ATR_CFM primitive and no ATR data is included.

3.2.3 Reset SIM Card

If the client application wants the server to reset the SIM card, the CSR_BT_SAPC_RESET_SIM_REQ and CSR_BT_SAPC_RESET_SIM_CFM primitives must be used. The use of the primitives is shown in Figure 6.

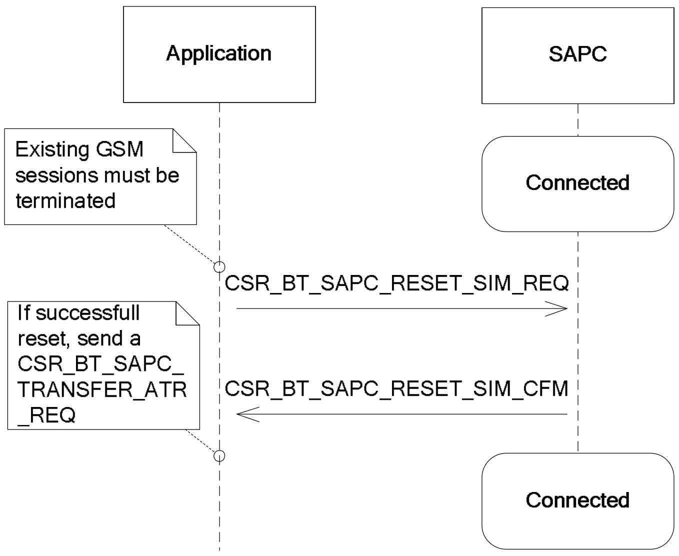

Application
SAPC
Existing GSM sessions must be terminated
Connected
CSR_BT_SAPC_RESET_SIM_REQ
If successfull reset, send a CSR_BT_SAPC_ TRANSFER_ATR _REQ
CSR_BT_SAPC_RESET_SIM_CFM
Connected

Figure 6: Reset the SIM card

Before sending the reset request, the application shall first terminate any existing GSM session involving the SIM card in the server. The application requests a SIM card reset by sending the CSR_BT_SAPC_RESET_SIM_REQ to the SAPC. This is sent to the server, which performs the reset of the SIM card and informs the SAPC about the result. After this, SAPC replies to the application using the CSR_BT_SAPC_RESET_SIM_CFM primitive. This primitive includes a *resultCode* indicating the result of the reset request. If CSR_BT_RSLT_OK_REQUEST is received, the reset was performed successfully, and the application must request the ATR of the SIM card. The ATR transfer is described in section Transfer ATR. If the reset was not successful the error is indicated in the *resultCode* field.

3.2.4 Power SIM Card Off

If the application on the client side wants the server to power off the SIM card the CSR_BT_SAPC_POWER_SIM_OFF_REQ and CSR_BT_SAPC_POWER_SIM_OFF_CFM primitives shall be used. The usage of the two primitives is depicted in Figure 7.

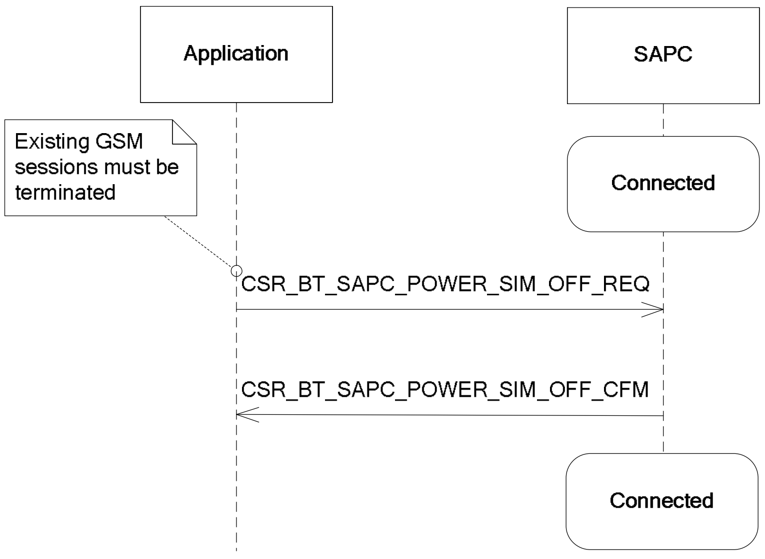

Application
SAPC
Existing GSM sessions must be terminated
Connected
CSR_BT_SAPC_POWER_SIM_OFF_REQ
CSR_BT_SAPC_POWER_SIM_OFF_CFM
Connected

Figure 7: Power the SIM card off

Before sending the power off request existing GSM session involving the SIM card must be terminated. The application requests the SIM card power off by sending the CSR_BT_SAPC_POWER_SIM_OFF_REQ to the SAPC. This request is sent to the server side, which upon reception powers off the SIM card i.e. removes the voltage from the card. The SAPC then replies to the application by sending the CSR_BT_SAPC_POWER_SIM_OFF_CFM primitive. The CSR_BT_SAPC_POWER_SIM_OFF_CFM includes a *resultCode* field indicating the result of the power down. If CSR_BT_RSLT_OK_REQUEST is included in *resultCode*, the power off was processed successfully. If an error has occurred during powering off the SIM card, the error is indicated in the *resultCode* field as well.

3.2.5 Power SIM Card On

If the application on the client side wants the server to power on the SIM card (i.e. to apply the supply voltage and clock signal to the SIM card) the CSR_BT_SAPC_POWER_SIM_ON_REQ and CSR_BT_SAPC_POWER_SIM_ON_CFM primitives are used. The usage of the two primitives is depicted in Figure 8.

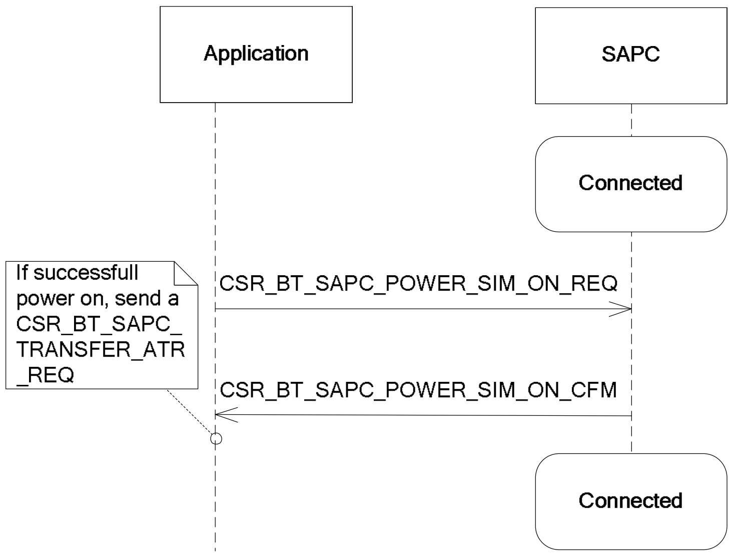

Application
SAPC
Connected
If successfull power on, send a CSR_BT_SAPC_TRANSFER_ATR_REQ
CSR_BT_SAPC_POWER_SIM_ON_REQ
CSR_BT_SAPC_POWER_SIM_ON_CFM
Connected

Figure 8: Power the SIM card on

The application sends the CSR_BT_SAPC_POWER_SIM_ON_REQ to the SAPC if powering on the SIM card is wanted. The server now powers on the SIM card and SAPC replies the application by sending the

CSR_BT_SAPC_POWER_SIM_ON_CFM primitive. The primitive includes a *resultCode* field indicating the result of the SIM card power on. If a CSR_BT_RSLT_OK_REQUEST is included the power on was processed successfully, and the application must request the ATR of the SIM card. The ATR transfer is described in section 3.2.2. If an error has occurred during powering on the SIM card, the error is indicated in the *resultCode* field as well. A complete list of possible values of the *resultCode* field can be seen in csr_bt_sap_common.h.

3.2.6 Transfer Card Reader Status

If the application on the client side wants the server to return the status of the card reader the CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_REQ and CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_CFM primitives are used. The usage of the two primitives is shown in Figure 9.

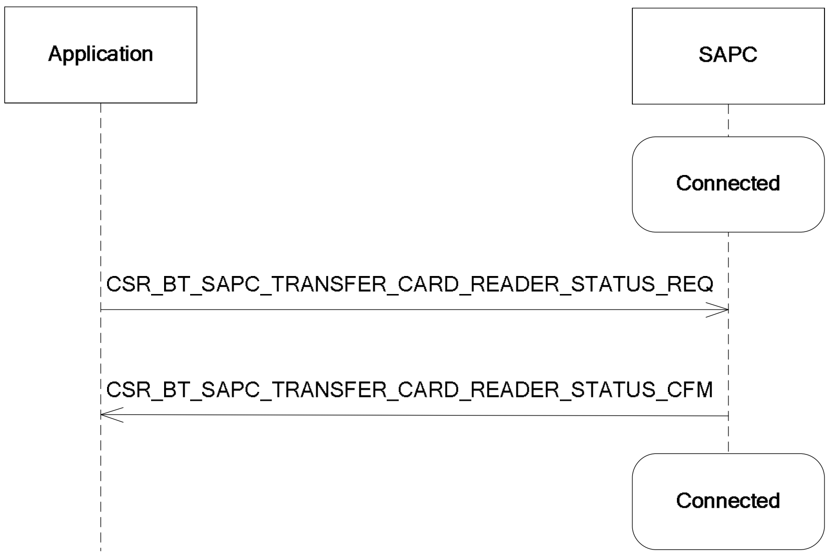

Application
SAPC
Connected
CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_REQ
CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_CFM
Connected

Figure 9: Transfer the card reader status

The application requests the card reader status by sending a CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_REQ to the SAPC. SAPC replies by sending a CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_CFM primitive to the application. This primitive includes both a *resultCode* and *cardReaderStatus* field. The *resultCode* indicates the result of the transfer card reader status request. If the value of the *resultCode* field is CSR_BT_RSLT_OK_REQUEST, the status was read and returned successfully and the *cardReaderStatus* then indicates the status of the card reader. If an error has occurred while reading the status of the card reader it is indicated in the *resultCode* and no *cardReaderStatus* will then be included. The possible values of the *resultCode* are defined in csr_bt_sap_common.h, whereas the values of the *cardReaderStatus* are defined in GSM 11.14.

3.2.7 Set Transfer Protocol

If the client wants to change the transfer protocol used for communicating to the SIM card the CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_REQ/CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_CFM primitives are used. The usage of the primitives is shown in Figure 10. In the CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_REQ the desired transfer protocol shall be specified. The possible values are the protocols T=0 or T=1. In csr_bt_sap_common.h there are two defines corresponding to the two protocols that shall be used when changing the protocol. The transfer protocols are defined in the ISO 7816-4 specification.

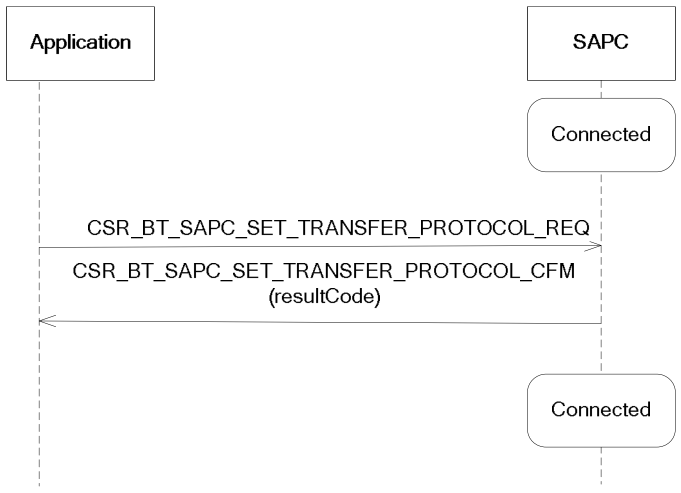

Application
SAPC
Connected
CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_REQ
CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_CFM (resultCode)
Connected

Figure 10: Set Transfer Protocol

The values for the result code parameter are defined in the csr_bt_sap_common.h file. The possible values in the result code are: CSR_BT_RSLT_OK_REQUEST or CSR_BT_RSLT_ERR_NOT_SUPPORTED.

3.2.8 Report Status

The report status scenario contains only one primitive, the CSR_BT_SAPC_STATUS_IND primitive. The usage of this primitive is shown in Figure 11.

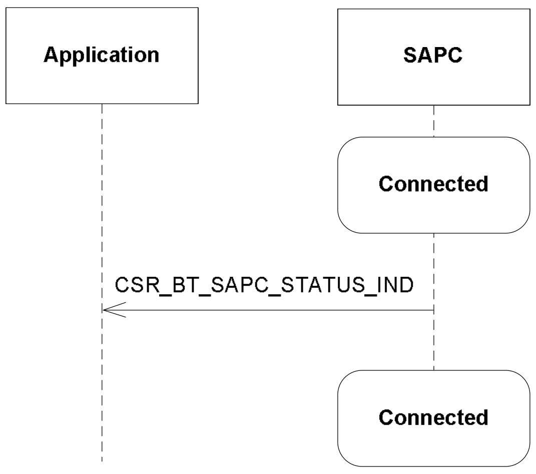


Figure 11: Report status

The above procedure is deployed during connection setup and whenever a change in the physical connection between the server and SIM card occurs during the lifetime of a SAP connection. A change in the physical connection between the SIM card and the server results in a CSR_BT_SAPC_STATUS_IND to the application informing about the change. The CSR_BT_SAPC_STATUS_IND primitive includes a *statusChange* field indicating the reason for the change in the status of the SIM card. The possible values for the *statusChange* field can be seen in csr_bt_sap_common.h.

3.3 Disconnect

Disconnection of the SIM Access connection can be done using two different disconnection schemes; a server or client initiated disconnect. These will be described in the following sections.

3.3.1 Disconnect Initiated by the Server Side

Two different disconnection schemes are applied to the server; immediate or graceful disconnection.

Graceful Disconnection:

The graceful disconnection scenario includes two primitives, namely CSR_BT_SAPC_DISCONNECT_IND and CSR_BT_SAPC_DISCONNECT_REQ. The usage of the two primitives is shown in Figure 12.

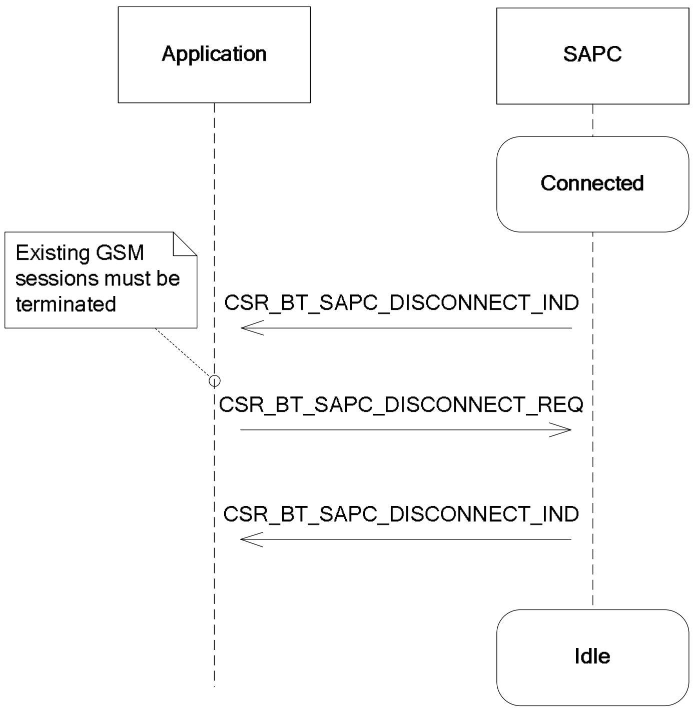

Application
SAPC
Connected
Existing GSM sessions must be terminated
CSR_BT_SAPC_DISCONNECT_IND
CSR_BT_SAPC_DISCONNECT_REQ
CSR_BT_SAPC_DISCONNECT_IND
Idle

Figure 12: Graceful server initiated disconnect

If the server wants to terminate the SIM Access Profile connection gracefully, the application receives a CSR_BT_SAPC_DISCONNECT_IND from the SAPC where the *disconnectType* field is set to CSR_BT_GRACEFUL_DISCONNECT. The graceful disconnection type allows the client to finish the transfer of APDUs and terminate any existing GSM session before disconnecting. When the existing GSM session is terminated the application sends the CSR_BT_SAPC_DISCONNECT_REQ to the SAPC informing the server that it is now ready for disconnection. SAPC replies with a CSR_BT_SAPC_DISCONNECT_IND including CSR_BT_IMMEDIATE_DISCONNECT in the *disconnectType* field meaning that the SIM Access connection is released.

Immediate Disconnection:

The immediate disconnection scenario only contains one primitive, namely CSR_BT_SAPC_DISCONNECT_IND. The usage of the primitive is depicted in Figure 13.

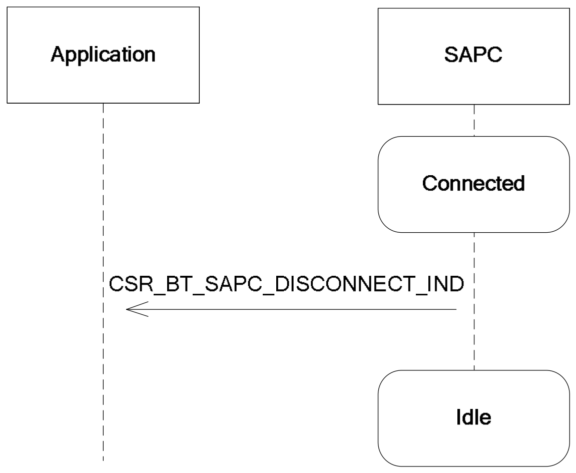


Figure 13: Immediate server initiated disconnect

From the application's point of view the immediate server initiated disconnect is indicated by reception of a CSR_BT_SAPC_DICONNECT_IND primitive from the SAPC with the *disconnectType* field set to CSR_BT_IMMEDIATE_DISCONNECT. Reception of this primitive means, that the SIM Access connection is released.

3.3.2 Disconnect Initiated by the Client Side

If the application of the client side wants to disconnect from SAPS the CSR_BT_SAPC_DISCONNECT_REQ and CSR_BT_SAPC_DISCONNECT_IND primitives must be used. The usage of these two primitives in relation to a client side initiated disconnection can be seen in Figure 14.

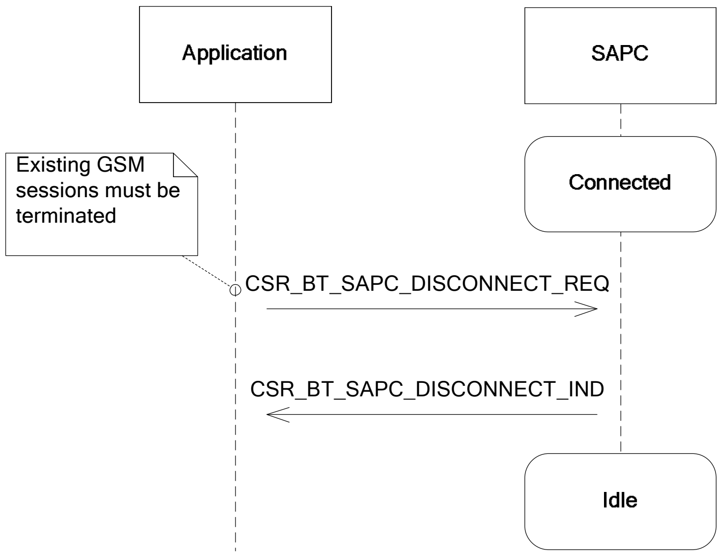

Application
SAPC
Existing GSM sessions must be terminated
Connected
CSR_BT_SAPC_DISCONNECT_REQ
CSR_BT_SAPC_DISCONNECT_IND
Idle

Figure 14: Client initiated disconnect

Before requesting the disconnection, the application must terminate any existing GSM sessions. The disconnection request is initiated from the application by sending a CSR_BT_SAPC_DISCONNECT_REQ to the SAPC. SAPC replies with a CSR_BT_SAPC_DISCONNECT_IND with CSR_BT_IMMEDIATE_DISCONNECT in the *disconnectField* and the SIM Access connection is successfully released.

NOTE: In the disconnection scenarios a CSR_BT_SAPC_DISCONNECT_IND with the *disconnectField* equals CSR_BT_IMMEDIATE_DISCONNECT always means that the connection is released.

4 SIM Access Profile Client Primitives

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

4.1 List of All Primitives

Primitives:	Reference:
CSR_BT_SAPC_CONNECT_REQ	See section 4.2
CSR_BT_SAPC_CONNECT_CFM	See section 4.2
CSR_BT_SAPC_DISCONNECT_REQ	See section 4.3
CSR_BT_SAPC_DISCONNECT_IND	See section 4.3
CSR_BT_SAPC_TRANSFER_APDU_REQ	See section 4.4
CSR_BT_SAPC_TRANSFER_APDU_CFM	See section 4.4
CSR_BT_SAPC_TRANSFER_ATR_REQ	See section 4.5
CSR_BT_SAPC_TRANSFER_ATR_CFM	See section 4.5
CSR_BT_SAPC_POWER_SIM_OFF_REQ	See section 4.6
CSR_BT_SAPC_POWER_SIM_OFF_CFM	See section 4.6
CSR_BT_SAPC_POWER_SIM_ON_REQ	See section 4.7
CSR_BT_SAPC_POWER_SIM_ON_CFM	See section 4.7
CSR_BT_SAPC_RESET_SIM_REQ	See section 4.8
CSR_BT_SAPC_RESET_SIM_CFM	See section 4.8
CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_REQ	See section 4.9
CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_CFM	See section 4.9
CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_REQ	See section 4.10
CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_CFM	See section 4.10
CSR_BT_SAPC_STATUS_IND	See section 4.11
CSR_BT_SAPC_SECURITY_OUT_REQ	See section 4.12
CSR_BT_SAPC_SECURITY_OUT_CFM	See section 4.12

Table 1: List of all primitives

4.2 CSR_BT_SAPC_CONNECT

Primitives \ Parameters	type	appHandle	bdAddr	maxMsgSize	resultCode	resultSupplier	btConnId
CSR_BT_SAPC_CONNECT_REQ	✓	✓	✓	✓			
CSR_BT_SAPC_CONNECT_CFM	✓			✓	✓	✓	✓

Table 2: CSR_BT_SAPC_CONNECT Primitives

Description

To start SIM Access session against a SIM Access server, the application must send a CSR_BT_SAPC_CONNECT_REQ primitive to SAPC. SAPC will then respond with a CSR_BT_SAPC_CONNECT_CFM primitive, which indicates whether a connection is established or not. A CSR_BT_OK_CONNECT indicates a successful connection, whereas all other values indicate a failure during the connection setup.

Parameters

type	Signal identity, CSR_BT_SAPC_CONNECT_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is always returned to appHandle .
bdAddr	The Bluetooth® address of the device to connect to.
maxMsgSize	In a connection request, the parameter is used for informing the server about the maximum message size supported by the application. This value is used during connect negotiation, and the result of the negotiation is presented to the application in the CSR_BT_SAPC_CONNECT_CFM, and will be the maximum allowed message size to use on the SIM Access connection.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. All values which are currently not specified in the respective prim.h file are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend`-function.

4.3 CSR_BT_SAPC_DISCONNECT

Primitives \ Parameters	type	resasonCode	reasonSupplier
CSR_BT_SAPC_DISCONNECT_REQ	✓		
CSR_BT_SAPC_DISCONNECT_IND	✓	✓	✓

Table 3: CSR_BT_SAPC_DISCONNECT Primitives

Description

The CSR_BT_SAPC_DISCONNECT primitives are used during the disconnection phase. The CSR_BT_SAPC_DISCONNECT_REQ must be sent by the application if it wants to disconnect from SAPS. The SAPC replies with CSR_BT_SAPC_DISCONNECT_IND indicating that the connection is terminated. Furthermore, the server can initiate a disconnection. In this case the application will receive a CSR_BT_SAPC_DISCONNECT_IND, indicating if the server wants the client to disconnect gracefully or immediately. Hence, the application must send a CSR_BT_SAPC_DISCONNECT_REQ. A CSR_BT_IMMEDIATE_DISCONNECT in the disconnectType parameter always means that the connection is released.

Parameters

type	Signal identity, CSR_BT_SAPC_DISCONNECT_REQ/IND
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

4.4 CSR_BT_SAPC_TRANSFER_APDU

Primitives \ Parameters	type	*commandApdu	commandApduLength	resultCode	resultSupplier	*responseApdu	apdu7816Type
CSR_BT_SAPC_TRANSFER_APDU_REQ	✓	✓	✓				✓
CSR_BT_SAPC_TRANSFER_APDU_CFM	✓		✓	✓	✓	✓	

Table 4: CSR_BT_SAPC_TRANSFER_APDU Primitives

Description

For transferring an APDU between the SIM Access client and server these two primitives are used. The application must send a CSR_BT_SAPC_TRANSFER_APDU_REQ if it wants to send a command APDU to the SIM card at the server. SAPC replies with a CSR_BT_SAPC_TRANSFER_APDU_CFM containing the response APDU from the SIM card. Furthermore, a result code is contained indicating the result of the execution of the command sent to the SIM card.

Parameters

type	Signal identity, CSR_BT_SAPC_TRANSFER_APDU_REQ/CFM.
*commandApdu	This is the Command APDU to be processed by the SIM card. The format of this APDU is defined in GSM 11.11 or GSM 11.14. Releasing the memory allocated for this pointer is not within the responsibility of the application.
commandApduLength	The length of the APDU.
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
*responseApdu	This is only valid if the Command APDU was processed successfully, and contains the SIM's response to the Command APDU. The memory allocated for this pointer must be released by the application.
apdu7816Type	This boolean parameter is used for indicating that the content of the APDU is coded in accordance with ISO 7816-4. If this parameter is set to false to content shall be coded according to the GSM 11.11 specification.

4.5 CSR_BT_SAPC_TRANSFER_ATR

Primitives \ Parameters	type	resultCode	resultSupplier	*atr	atrLength
CSR_BT_SAPC_TRANSFER_ATR_REQ	✓				
CSR_BT_SAPC_TRANSFER_ATR_CFM	✓	✓	✓	✓	✓

Table 5: CSR_BT_SAPC_TRANSFER_ATR Primitives

Description

These primitives are used when the application wants the server to send the ATR (Answer To Reset) from the SIM card. The CSR_BT_SAPC_TRANSFER_ATR_REQ can be used by the application to perform this task. SAPC responds with the CSR_BT_SAPC_TRANSFER_ATR_CFM containing the result of the ATR request and the ATR itself.

Parameters

type	Signal identity, CSR_BT_SAPC_TRANSFER_ATR_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 areregarded 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
*atr	This is only included if the resultCode indicated no error, and is ATR from the SIM card. The memory allocated for this pointer must be released by the application.
atrLength	The length of the ATR.

4.6 CSR_BT_SAPC_POWER_SIM_OFF

Primitives \ Parameters	type	resultCode	resultSupplier
CSR_BT_SAPC_POWER_SIM_OFF_REQ	✓		
CSR_BT_SAPC_POWER_SIM_OFF_CFM	✓	✓	✓

Table 6: CSR_BT_SAPC_POWER_SIM_OFF Primitives

Description

These two primitives are relevant when the application wants to power off the SIM card. The application requests the SIM card power off by sending a CSR_BT_SAPC_POWER_SIM_OFF_REQ to the SAPC. The CSR_BT_SAPC_POWER_SIM_OFF_REQ powers off the SIM card, i.e. it removes the voltage from the card. SAPC replies with a CSR_BT_SAPC_POWER_SIM_OFF_CFM containing the result of the power SIM card off request.

Parameters

type	Signal identity, CSR_BT_SAPC_POWER_SIM_OFF_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

4.7 CSR_BT_SAPC_POWER_SIM_ON

Primitives \ Parameters	type	resultCode	resultSupplier
CSR_BT_SAPC_POWER_SIM_ON_REQ	✓		
CSR_BT_SAPC_POWER_SIM_ON_CFM	✓	✓	✓

Table 7: CSR_BT_SAPC_POWER_SIM_ON Primitives

Description

If the SIM card is powered off the application can request the server to power on the SIM card again. The application requests the SIM card power on by sending a CSR_BT_SAPC_POWER_SIM_ON_REQ to SAPC. SAPC will then respond with a CSR_BT_SAPC_POWER_SIM_ON_CFM informing about the result of the request.

Parameters

type	Signal identity, CSR_BT_SAPC_POWER_SIM_ON_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

4.8 CSR_BT_SAPC_RESET_SIM

Primitives \ Parameters	type	resultCode	resultSupplier
CSR_BT_SAPC_RESET_SIM_REQ	✓		
CSR_BT_SAPC_RESET_SIM_CFM	✓	✓	✓

Table 8: CSR_BT_SAPC_RESET_SIM Primitives

Description

These two primitives are used when the application wants the server to reset the SIM card. The application requests the reset by sending a CSR_BT_SAPC_RESET_SIM_REQ to SAPC. When the reset is performed, SAPC replies with a CSR_BT_SAPC_RESET_SIM_CFM indicating the result of the reset request.

Parameters

type	Signal identity, CSR_BT_SAPC_RESET_SIM_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

4.9 CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS

Primitives \ Parameters	type	resultCode	resultSupplier	cardReaderStatus
CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_REQ	✓			
CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_CFM	✓	✓	✓	✓

Table 9: CSR_BT_SAPC_ TRANSFER_CARD_READER_STATUS Primitives

Description

If the application wants to know the status of the card reader these two primitives are used. The application requests the status of the card reader by sending a CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_REQ to SAPC. When the request is performed, SAPC replies with a CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_CFM containing both a result of the request and the status of the card reader.

Parameters

type	Signal identity, CSR_BT_SAPC_TRANSFER_CARD_READER_STATUS_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
cardReaderStatus	Is the card reader status as described in GSM 11.14.

4.10 CSR_BT_SAPC_SET_TRANSFER_PROTOCOL

Parameters / Primitives	type	resultCode	resultSupplier	transportProtocol
CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_REQ	✓			✓
CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_CFM	✓	✓	✓	

Table 10: CSR_BT_SAPC_SET_TRANSFER_PROTOCOL Primitives

Description

If the SAP Client wants to change the transfer protocol used for transmitting APDUs it should send the CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_REQ signal with the transportProtocol parameter indicating which transport protocol to use.

In the CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_CFM signal it will be indicated if the host supports this protocol and has changed to this protocol.

Parameters

type	Signal identity, CSR_BT_SAPC_SET_TRANSFER_PROTOCOL_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 areregarded 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
transportProtocol	Can be either T=0 or T=1 and is defined in csr_bt_sap_common.h.

4.11 CSR_BT_SAPC_STATUS

Parameters / Primitives	type	statusChange
CSR_BT_SAPC_STATUS_IND	✓	✓

Table 11: CSR_BT_SAPC_STATUS Primitives

Description

The CSR_BT_SAPC_STATUS_IND is used for informing the application about a change in the availability of the SIM card. The CSR_BT_SAPC_STATUS_IND can arrive at any time from SAPC while being connected.

Parameters

type	Signal identity, CSR_BT_SAPC_STATUS_IND
statusChange	This is the reason for the change in the status of the SIM card. The possible values for this parameter are defined in csr_bt_sap_common.h.

4.12 CSR_BT_SAPC_SECURITY_OUT

Primitives \ Parameters	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_SAPC_SECURITY_OUT_REQ	✓	✓	✓		
CSR_BT_SAPC_SECURITY_OUT_CFM	✓			✓	✓

Table 12: CSR_BT_SAPC_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_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_SAPC_SECURITY_OUT_REQ/CFM.

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

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

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

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

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

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

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

5 Document References

Document	Reference
CSR Synergy Bluetooth. CM – Connection Manager API Description, doc. no. api-0101-cm	[CM]
CSR Synergy Bluetooth, SC – Security Controller API Description.	[SC]
SIG SIM Access Profile, Interoperability Specification v. 1.1	[SAPSPEC]

Terms and Definitions

APDU	Application Protocol Data Unit
ATR	Answer To Reset
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
CM	Connection Manager
CSR	Cambridge Silicon Radio
HMSC	High-level Message Sequence Chart
L2CAP	Logical Link Control and Adaptation Protocol
SAPC	SIM Access Profile Client
SAPS	SIM Access Profile Server
SC	Security Controller
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

Life Support Policy and Use in Safety-critical Compliance

Performance and Conformance

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