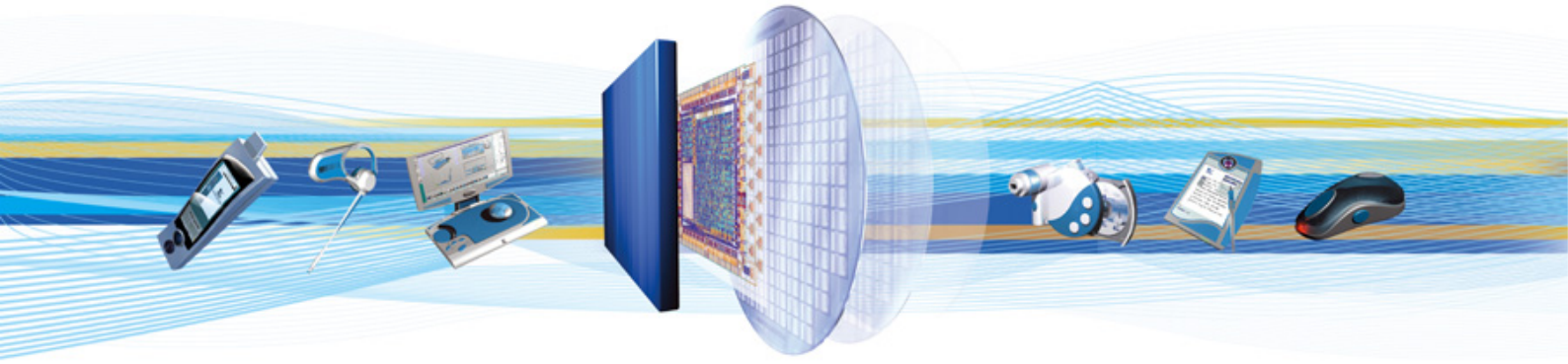
CSR Synergy Bluetooth 18.2.0

SyncML Client

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 OBEX SyncML Client implementation (SMLC). The SMLC conforms to the client side of the OBEX-SyncML binding description concerning the mandatory issues, ref. [SMLOBEXBINDING].

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. SMLC and the application
- The SMLC shall only handle one request at the time
- Bonding (pairing) is NOT handled by the SMLC

2 Description

2.1 Introduction

The scenarios covered by this profile are the following:

- Usage of a Bluetooth® device e.g. a notebook PC to be able to synchronize PIM stores of it self and another Bluetooth device e.g. a mobile phone. Synchronisation involves exchanging SyncML-messages (objects holding the needed sync-information to be exchanged between the involved units as described in the [SMLREPPROT] and the [SMLOBEXBINDING].

The SMLC provides the following services to the application:

- Bluetooth connection handling
- OBEX protocol handling

The application is responsible for handling the requests and confirms from the SMLC with correct data (objects) as described in the IrOBEX specification [OBEX] and [SMLOBEXBINDING]. The SMLC is not checking if the data (SyncML-messages) are packed correctly with certain data placed in certain places, for details see ref. [SMLREPPROT].

2.2 Reference Model

The SMLC interfaces to the Connection Manager (CM) and the SMLC is a service layer to the APP-layer (APP-layer holding SyncML-client application code).

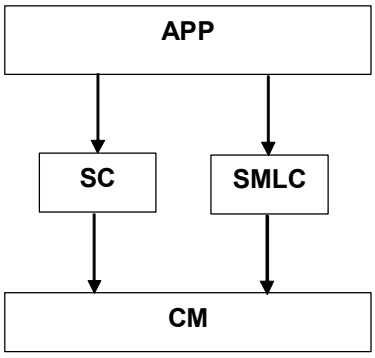


Figure 1: Reference model

2.3 Sequence Overview

When the SMLC starts up it enters IDLE state. To activate the SMLC making it ready for use as an OBEX-client a (CSR_BT_SMLC_ACTIVATE_REQ) has to be imposed to it making it to enter the ACTIVE state. After the SMLC has activated it sends back the signal CSR_BT_SMLC_ACTIVATE_CFM indicating to the application that it has finished the activation procedure. The application has to wait for the CSR_BT_SMLC_ACTIVATE_CFM before any other signals are sent to the SMLC.

If, in ACTIVE state and a (CSR_BT_SMLC_CONNECT_REQ) is received from the application, the SMLC starts to connect to the specified device and the CONNECTED state is entered, then the application receives a confirmation on the connect. The application can then issue a request to (put- or get) to start the transfer of SyncML data objects to/from the connected remote SyncML-server. The application can perform multiple gets/puts (one at the time to fulfill all SyncML data-transfer) before disconnecting the connection. When the application disconnects, the SMLC re-enters ACTIVE state.

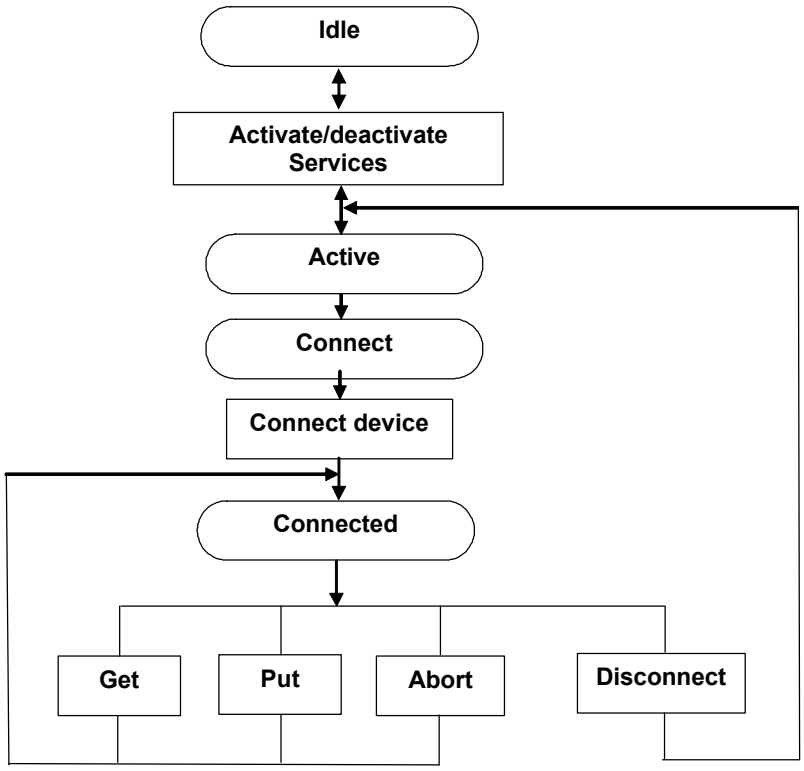


Figure 2: SMLC state diagram

3 Interface Description

3.1 Activation

To activate the SMLC to get it ready for OBEX-SyncML-Client service, the application has to send a CSR_BT_SMLC_ACTIVATE_REQ to the SMLC. After SMLC activation the SMLC sends back a CSR_BT_SMLC_ACTIVATE_CFM indicating to the application that the SMLC is active. Possible request parameters are (advertise_enable) and accept_incoming_connect. Both these parameters always have to be set to FALSE when sending the request. When the SMLC has performed activation it sends CSR_BT_SMLS_ACTIVATE_CFM back to the application and the SMLC is now ready to handle any incoming requests from the application.

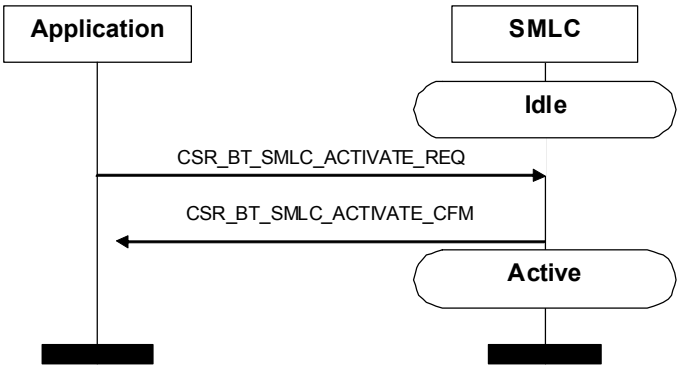


Figure 3: SMLC activation

3.2 Deactivation

To deactivate the SMLC, the application has to send a CSR_BT_SMLC_DEACTIVATE_REQ. When the SMLC receives the CSR_BT_SMLC_DEACTIVATE_REQ it closes down the current ongoing activities and then sends back a CSR_BT_SMLC_DEACTIVATE_CFM indicating to the application that the SMLC is deactivated. Deactivation may take some time depending on the state in which the SMLC receives the CSR_BT_SMLC_DEACTIVATE_REQ.The application has to wait for the CSR_BT_SMLC_DEACTIVATE_CFM before e.g. re-activating the SMLC again.

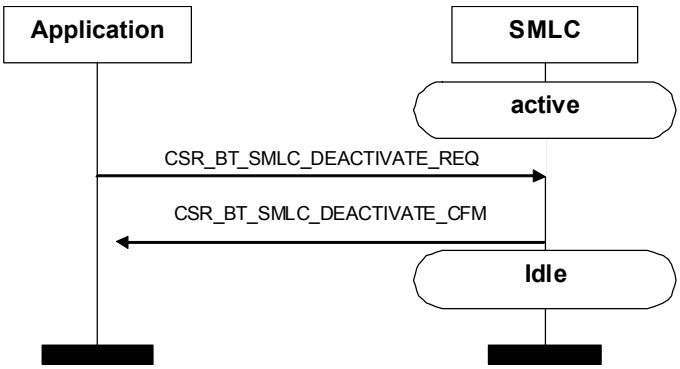


Figure 4: SMLC deactivation

3.3 Connect

When the application wants to connect to a SyncML Server it has to send a CSR_BT_SMLC_CONNECT_REQ to the SMLC. In this message the application has to specify which device to connect to. The parameter 'authorize' is controlling if the SyncML client wants to OBEX-authenticate the SyncML server. If the parameter is TRUE the application has to specify the password parameter. This message has also a parameter called maxPacketSize, which indicates the maximum Obex packet size, which the application wants to receive from the SyncML server side. The value can be between 255 bytes to 64Kbytes – 1, see definition in ref. [OBEX]. If the packet size is large it is optimizing for faster data-transfer, but the disadvantage will be use of big memory blocks.

The SMLC sends a CSR_BT_SMLC_CONNECT_CFM message to the application, which has the status of the connection establishment - this is the parameter result code. For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure in the connection.

Once the Obex connection is established, the SMLC will, transparently for the application layer, make use of low power modes. This implies use of sniff if supported by the SyncML-Server side. Low power modes are enabled using a supervision timer. If no data is received within the specified time interval, the SMLC manager will attempt a change to low power mode if possible. The value of the timer is determined by the CSR_BT_SMLC_LP_SUPERVISION_TIMEOUT and is defined in the csr_bt_smlc_handler.h file.

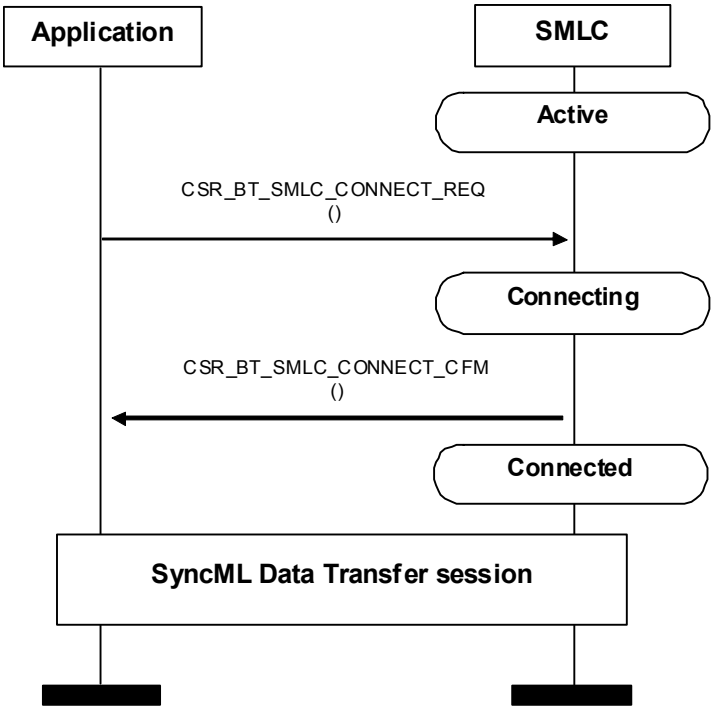


Figure 5: Connection handling

3.4 Cancel Connect

The application can cancel an outgoing connection request by sending a CSR_BT_SMLC_CANCEL_CONNECT_REQ. If the outgoing connection can be cancelled the responses will be a CSR_BT_SMLC_CONNECT_CFM with a response code different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE.

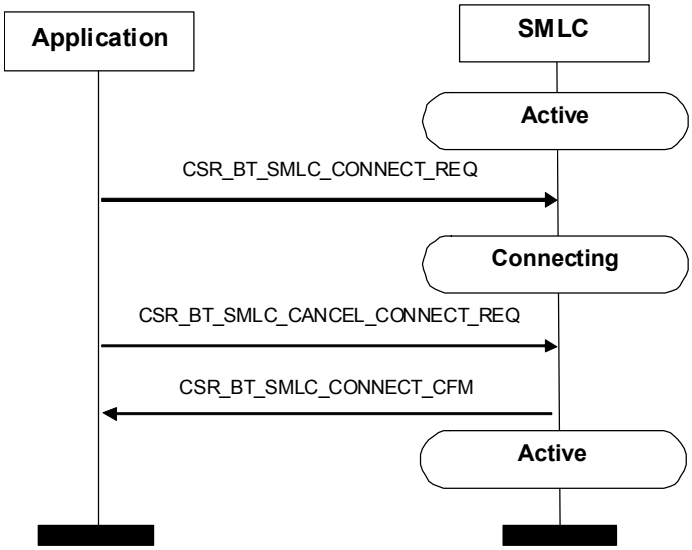


Figure 6: Cancel Connect I

Please note that if the application requests a CSR_BT_SMLC_CANCEL_CONNECT_REQ while the SMLC is sending a CSR_BT_SMLC_CONNECT_CFM with the response code CSR_BT_OBEX_SUCCESS_RESPONSE_CODE to the application, then the SMLC will consider the CSR_BT_SMLC_CANCEL_CONNECT_REQ as a CSR_BT_SMLC_DISCONNECT_REQ and the application will receive a CSR_BT_SMLC_DISCONNECT_IND.

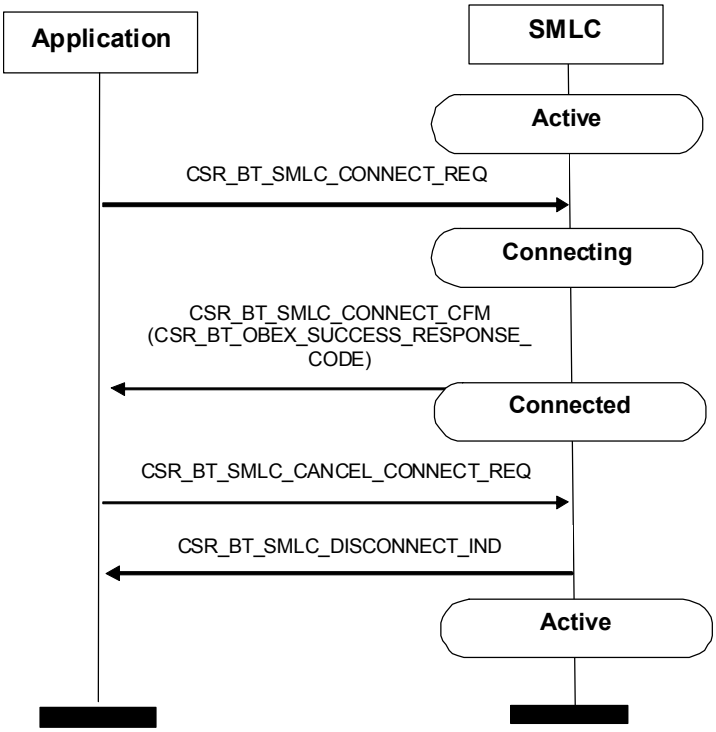


Figure 7: Cancel Connect II

3.5 SyncML Message Transfer

SyncML packets holding one or more SyncML messages are transmitted to the SyncML server by issuing a CSR_BT_SMLC_PUT_SML_MSG_OBJ_REQ followed by one or more CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ's. The server side responds with the result of the operation in a CSR_BT_SMLC_PUT_SML_MSG_OBJ_CFM signal. In case the application wants to fragment the body due to memory considerations it can do so by sending a CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ with finalFlag set to FALSE. On the confirm it can continue to send the next fragment. It can continue until it sets the finalFlag to TRUE.

Attention a SyncML-packet containing more SyncML messages has to be split when transferring it. The split has to be done so that each individual SyncML message is sent in a separate set of PUT/PUT_NEXTCHUNK sessions. So a SyncML-packet containing e.g. four SyncML-messages has to be transferred in four successive but individual PUT/PUT_NEXTCHUNK sessions.

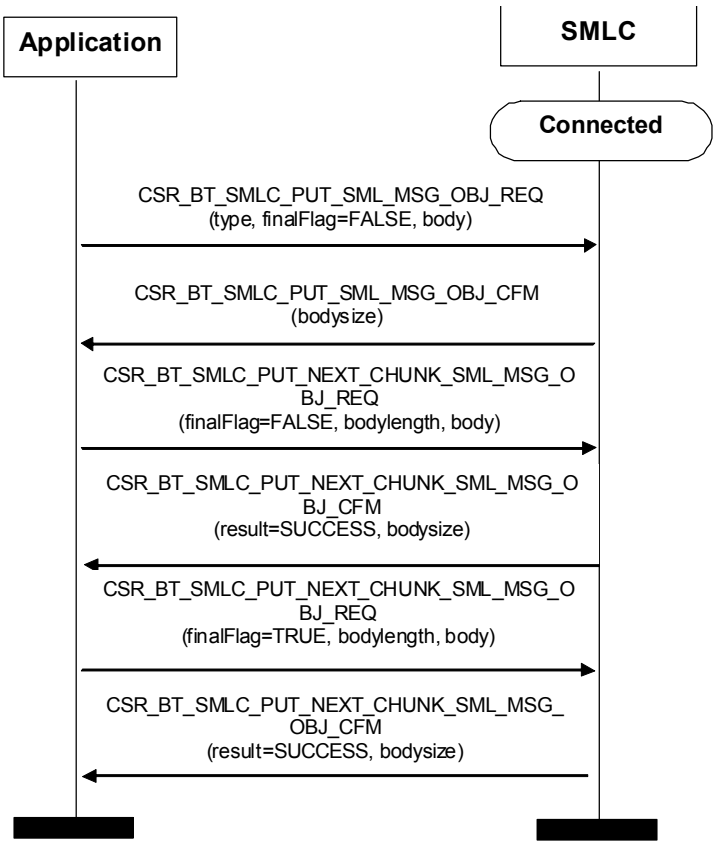


Figure 8: Put smlmsg object

SyncML packets holding one or more SyncML messages can be pulled from the server by issuing a CSR_BT_SMLC_GET_SML_MSG_OBJ_REQ. The server responds with the result of the operation in a CSR_BT_SMLC_GET_SML_MSG_OBJ_CFM signal. If the server responds with multiple fragments, the first will be received by the application as a CSR_BT_SMLC_GET_SML_MSG_OBJ_CFM and the application has to send a CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_REQ to get the next fragment. This is the pattern until the finalFlag parameter in the confirm is set indicating that it is the last SyncML message fragment. If there are multiple SyncML messages in the SyncML packet that are pulled, the application has to initiate yet another GET/GET_NEXT_CHUNK session until the “SyncML-packet-end”-code is received in a (the last) SyncML message received. This indicates to the application that the complete SyncML packet has been received.

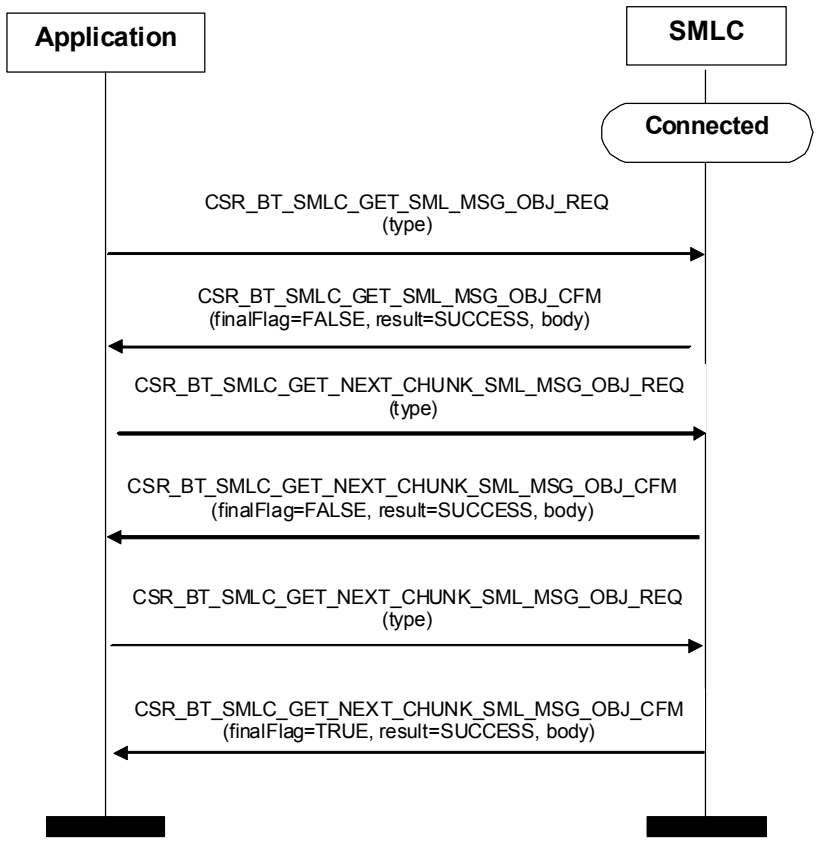


Figure 9: Get smlmsg object

3.6 Abort Operation

The abort request (CSR_BT_SMLC_ABORT_REQ) is used when the application decides to terminate a multi-packet operation (such as PUT or GET) before it ends. The response (CSR_BT_SMLC_ABORT_CFM) is received indicating that the abort request is a success. It is also indicating that the abort request is received by the server-side and the SyncML server is now resynchronized with the client. If anything else is returned the SMLC will disconnect the link and send a CSR_BT_SMLC_DISCONNECT_IND to the application.

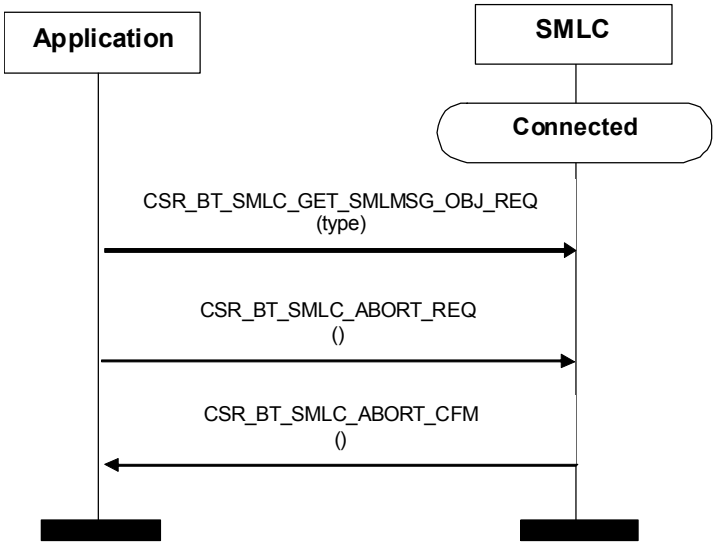


Figure 10: Abort operation

3.7 Obex Authentication

The application can OBEX authenticate the SyncML server by setting the authorize and password parameters in the CSR_BT_SMLC_CONNECT_REQ. The SMLC will then authenticate the SyncML server under the Obex connection establishment. A SyncML server can authenticate the SMLC on every operation individually. For each application request it sends it can receive a CSR_BT_SMLC_AUTHENTICATE_IND instead of the corresponding confirm message. If the application receives a CSR_BT_SMLC_AUTHENTICATE_IND it must response with a CSR_BT_SMLC_AUTHENTICATE_RES signal using the password or pin number that the SyncML server requires. An example of the authentication sequence is illustrated below.

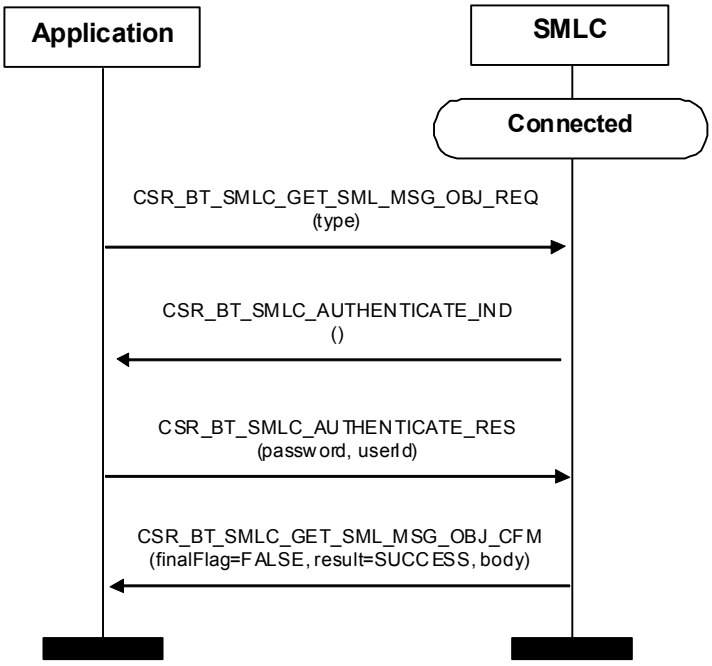


Figure 11: Authenticate get smlmsg object

3.8 Disconnect

Sending a CSR_BT_SMLC_DISCONNECT_REQ to the SMLC disconnects the current connection (if any). To disconnect may take some time and is confirmed with a CSR_BT_SMLC_DISCONNECT_CFM signal.

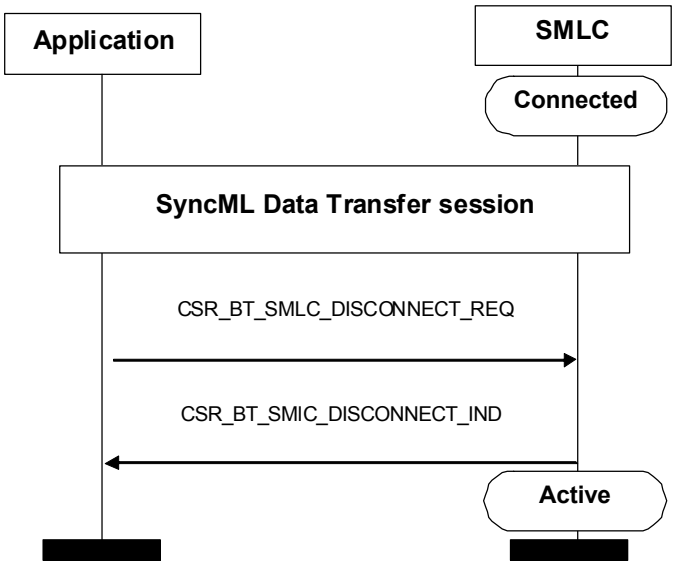


Figure 12: Normal disconnect

In case the peer side prematurely disconnects, the SMLC sends a CSR_BT_SMLC_DISCONNECT_IND to the application and enters ACTIVE state.

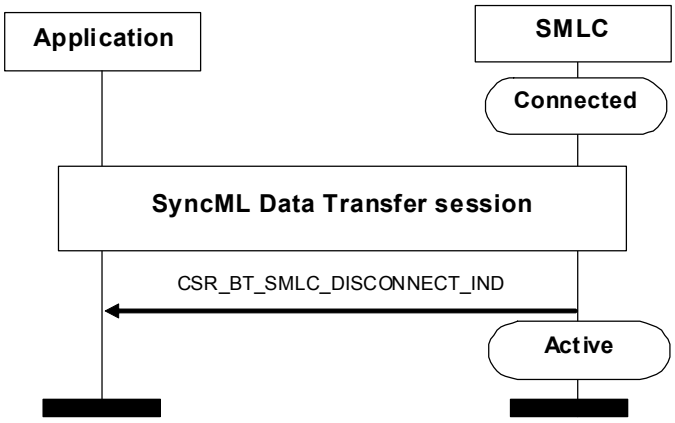


Figure 13: Abnormal disconnect

3.9 Payload Encapsulated Data

3.9.1 Using Offsets

As many OBEX messages contain multiple parameters with variable length, some of the parameters are based on *offsets* instead of standard pointers to the data. Signals with offset-based data can easily be recognized as they have both a *payload* and a *payloadLength* parameter. The *payload* contains the actual data, on which the offset is based. For example, a typical signal may contain the following:

```
CsrBtCommonPrim    type;
CsrUint8               result;
CsrUint16          bodyOffset;
CsrUint16          bodyLength;
CsrUint16          payloadLength;
CsrUint8               *payload;
```

In this example, one offset parameter can be found, namely *bodyOffset*. To obtain the actual data, the offset value is added to the *payload* pointer, which yields a pointer to the data, i.e.:

```
CsrUint8 *body;
body = (CsrUint8*)(primitive->payload + primitive->bodyOffset);
```

As can be seen, the offset contains the number of bytes within the *payload* where the information begins.

And to illustrate the usage of the *length* parameter, which is also a common parameter, to copy the body one would typically use:

```
CsrMemCpy( copyOfBody, body, primitive->bodyLength );
```

Offset parameters will always have an “Offset” suffix on the name, and offsets are *always* relative to the “payload” parameter.

3.9.2 Payload Memory

When the application receives a signal which has a *payload* parameter, the application must always free the payload pointer to avoid memory leaks, for example

```
CsrPfree(primitive->payload);
CsrPfree(primitive);
```

will free both the payload data and the message itself. Note that when the payload has been freed, offsets can not be used anymore, as the actual data is contained within the payload.

Signals that do not use the *payload* parameter must still have each of their pointer-based parameters freed.

Likewise, the profile will free any pointers received as parameters in API signals or functions

4 OBEX SyncML 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_smlc_prim.h file.

4.1 List of All Primitives

Primitives:	Reference:
CSR_BT_SMLC_ACTIVATE_REQ	See section 4.2
CSR_BT_SMLC_ACTIVATE_CFM	See section 4.2
CSR_BT_SMLC_DEACTIVATE_REQ	See section 4.3
CSR_BT_SMLC_DEACTIVATE_CFM	See section 4.3
CSR_BT_SMLC_CONNECT_REQ	See section 4.4
CSR_BT_SMLC_CONNECT_CFM	See section 4.4
CSR_BT_SMLC_AUTHENTICATE_IND	See section 4.5
CSR_BT_SMLC_AUTENTICATE_RES	See section 4.5
CSR_BT_SMLC_GET_SML_MSG_OBJ_REQ	See section 4.6
CSR_BT_SMLC_GET_SML_MSG_OBJ_CFM	See section 4.6
CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_REQ	See section 4.7
CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_CFM	See section 4.7
CSR_BT_SMLC_PUT_SML_MSG_OBJ_REQ	See section 4.8
CSR_BT_SMLC_PUT_SML_MSG_OBJ_CFM	See section 4.8
CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ	See section 4.8
CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_CFM	See section 4.8
CSR_BT_SMLC_ABORT_REQ	See section 4.9
CSR_BT_SMLC_ABORT_CFM	See section 4.9
CSR_BT_SMLC_DISCONNECT_REQ	See section 4.10
CSR_BT_SMLC_DISCONNECT_IND	See section 4.10
CSR_BT_SMLC_CANCEL_CONNECT_REQ	See section 4.11
CSR_BT_SMLC_SECURITY_OUT_REQ	See section 4.12
CSR_BT_SMLC_SECURITY_OUT_CFM	See section 4.12

Table 1: List of all primitives

4.2 CSR_BT_SMLC_ACTIVATE

Primitives \ Parameters	type	appHandle	advEnable	acceptServConnect	advEnabled	acceptEnabled	resultCode	resultSupplier	windowSize	smEnable
CSR_BT_SMLC_ACTIVATE_REQ	✓	✓	✓	✓					✓	✓
CSR_BT_SMLC_ACTVATE_CFM	✓				✓	✓	✓	✓		

Table 2: CSR_BT_SMLC_ACTIVATE Primitives

Description

To activate the OBEX SyncML client (SMLC), the application sends a CSR_BT_SMLC_ACTIVATE_REQ; the contained control booleans all have to be set to FALSE….. The SMLC responds with a CSR_BT_SMLC_ACTIVATE_CFM. In case the result in the confirmation is CSR_BT_SMLC_SUCCES the activation is established. Any other value indicates a failure in the activation-process.

Parameters

type	Signal identity, CSR_BT_SMLC_ACTIVATE_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 returned to appHandle.
advEnable	Setup parameter setting the way the SMLC is to work. ALWAYS has to be set to FALSE see csr_bt_smlc_prim.h.
acceptServConnect	Setup parameter setting the way the SMLC is to work. ALWAYS has to be set to FALSE see csr_bt_smlc_prim.h.
advEnabled	Acknowledge parameter, telling the application how the SMLC is working.
acceptEnabled	Acknowledge parameter, telling the application how the SMLC is working.
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. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h 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
windowSize	Controls how many packets the OBEX profile (and lower protocol layers) are allowed to cache on the data receive side. A value of zero (0) will cause the system to auto-detect this value.
srmEnable	Enable local support for Single Response Mode.

4.3 CSR_BT_SMLC_DEACTIVATE

Primitives \ Parameters	type
CSR_BT_SMLC_DEACTIVATE_REQ	✓
CSR_BT_SMLC_DEACTVATE_CFM	✓

Table 3: CSR_BT_SMLC_DEACTIVATE Primitives

Description

To deactivate the OBEX SyncML client side (SMLC), the application sends a CSR_BT_SMLC_DEACTIVATE_REQ. SMLC responds with a CSR_BT_SMLC_DEACTVATE_CFM.

Parameters

type Signal identity, CSR_BT_SMLC_DEACTIVATE_REQ/CFM.

4.4 CSR_BT_SMLC_CONNECT

Primitives \ Parameters	type	appHandle	maxPacketSize	destination	targetService	authorize	resultCode	resultSupplier	obexPeerMaxPacketSize	availablePutBodySize	realmLength	*realm	passwordLength	*password	*userId	length	count	btConnId	windowSize	srmEnable
CSR_BT_SMLC_CONNECT_REQ	✓	✓	✓	✓	✓	✓					✓	✓	✓	✓	✓	✓	✓		✓	✓
CSR_BT_SMLC_CONNECT_CFM	✓						✓	✓	✓	✓								✓		

Table 4: CSR_BT_SMLC_CONNECT Primitives

Description

To start an OBEX SyncML transfer session against the SyncML server side, the application sends a CSR_BT_SMLC_CONNECT_REQ with the wanted max packet size allowed to send from the server. The server-side responds with a CSR_BT_SMLC_CONNECT_CFM. In case the result in the confirmation is CSR_BT_OBEX_SUCCESS_RESULT_CODE the connection is established. Any other value indicates a failure in the connection.

The connect messages between the OBEX SyncML transfer client and Server is guarded by a timer, thus if for some reason the server do not reply to the OBEX connect request within a fixed time interval the Bluetooth connection is disconnected direct. The timeout functionality is per default set to five seconds, or twenty seconds if the client request OBEX authentication. The timeout value can be disable, or change by changing CSR_BT_OBEX_CONNECT_TIMEOUT or CSR_BT_OBEX_CONNECT_WITH_AUTH_TIMEOUT, which is define in csr_bt_user_config.default.h. Note if the value of CSR_BT_OBEX_CONNECT_TIMEOUT or CSR_BT_OBEX_CONNECT_WITH_AUTH_TIMEOUT is change, it will influence all OBEX profiles.

The function:

```
CsrBtSmlcConnectReqSend(CsrSchedQid          appHandle,
                        CsrUint16           maxPacketSize,
                        CsrBtDeviceAddr     destination,
                        CsrBool             targetService,
                        CsrBool             authorize,
                        CsrUint16           realmLength,
                        data_ptr_t          *realm,
                        CsrUint16           passwordLength,
                        CsrUint8            *password,
                        CsrCharString       *userId,
                        CsrUint32           length,
                        CsrUint32           count,
                        CsrUint16           windowSize,
                        CsrBool             srmEnable );
```

defined in csr_bt_smlc_lib.h, builds and sends the CSR_BT_SMLC_CONNECT_REQ primitive to the SMLC profile.

Parameters

type	Signal identity, CSR_BT_SMLC_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 returned to appHandle.

maxPacketSize	The maximum obex packet size allowed sending to the client application.
destination	The Bluetooth® address of the device to connect to.
targetService	What type of SyncML-service is to be carried out on the OBEX-connection. see csr_bt_smlc_prim.h.
authorize	Is to control the OBEX authentication on connection to the SyncML server side. If TRUE the SMLC will initiate the authentication against the server.
password	Containing the challenge password of the OBEX authentication.
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. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h 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
obexPeerMaxPacketSize	Indicates the maximum size OBEX packet that is allowed to be sent to the server
availablePutBodySize	Indicates the present room for bodydata in e.g. when to start a put-session against the server-side.
realmLength	Number of bytes in realm of type CsrUint16 **Note** in this release version the 'realmLength' parameter is ignored right now.
*realm	A displayable string indicating for the user which userid and/or password to use. The first byte of the string is the character set of the string. The table below shows the different values for character set. Note that this pointer must be pfree by the application, and that this pointer can be NULL because the realm field is optional to set by the peer device. **Note** in this release version the 'realm' pointer is ignored.

Char set Code	Meaning
0	ASCII
1	ISO-8859-1
2	ISO-8859-2
3	ISO-8859-3
4	ISO-8859-4
5	ISO-8859-5
6	ISO-8859-6
7	ISO-8859-7
8	ISO-8859-8
9	ISO-8859-9

0xFF = 255	UNICODE

*userId	Zero terminated string (ASCII) containing the ‘userId’ for the authentication. This is a pointer which shall be allocated by the application. Note in this version the ‘userId’ is ignored.
length	Length is use to express the approximate total length of the bodies of all the objects in the transaction. If set to 0 this header will not be include.
count	Count is use to indicate the number of objects that will be sent during this connection. If set to 0 this header will not be include.
btConnId	Identifier used when moving the connection to another AMP controller, i.e. when calling the `CsrBtAmpmMoveReqSend`-function.
windowSize	Controls how many packets the OBEX profile (and lower protocol layers) are allowed to cache on the data receive side. A value of zero (0) will cause the system to auto-detect this value.
srmEnable	Enable local support for Single Response Mode.

4.5 CSR_BT_SMLC_AUTHENTICATE

Primitives \ Parameters	type	options	realmLength	* realm	deviceAddr	*password	passwordLength	*userId
CSR_BT_SMLC_AUTHENTICATE_IND	✓	✓	✓	✓	✓			
CSR_BT_SMLC_AUTHENTICATE_RES	✓					✓	✓	✓

Table 5: CSR_BT_SMLC_AUTHENTICATE Primitives

Description

The indication and response signal is used when the SyncML-server-side wants to OBEX authenticate the SyncML client-side. The application has to response with the password or pin number in the password and userId for the server-side to identify the proper password.

Parameters

type — Signal identity, CSR_BT_SMLC_AUTHENTICATE_IND/RES.

options — Challenge information of type CsrUint8.

Bit 0 controls the responding of a valid user Id.
If bit 0 is set it means that the application must response with a user Id in a CSR_BT_SMLC_AUTHENTICATE_RES message. If bit 0 is not set the application can just set the userId to NULL.

Bit 1 indicates the access mode being offered by the sender.
If bit 1 is set the access mode is read only. If bit 1 is not set the sender gives full access, e.g. both read and write.

Bit 2 - 7 is reserved.

realmLength — Number of bytes in realm of type CsrUint16

* realm — A displayable string indicating for the user which userid and/or password to use. The first byte of the string is the character set of the string. The table below shows the different values for character set.

Note that this pointer must be CsrPfree by the application, and that this pointer can be NULL because the realm field is optional to set by the peer device.

Char set Code	Meaning
0	ASCII
1	ISO-8859-1
2	ISO-8859-2
3	ISO-8859-3
4	ISO-8859-4

5	ISO-8859-5
6	ISO-8859-6
7	ISO-8859-7
8	ISO-8859-8
9	ISO-8859-9
0xFF = 255	UNICODE

deviceAddr	The Bluetooth address of the device that has initiated the OBEX authentication procedure
*password	Containing the response password of the OBEX authentication.
passwordLength	The length of the response password.
*userId	Pointer to a zero terminated string (ASCII) containing the userId for the authentication.

4.6 CSR_BT_SMLC_GET_SML_MSG_OBJ

Parameters / Primitives	type	mimeType	responseCode	lengthOfObject	finalFlag	bodyLength	bodyOffset	payloadLength	*payload	smpOn
CSR_BT_SMLC_GET_SML_MSG_OBJ_REQ	✓	✓								✓
CSR_BT_SMLC_GET_SML_MSG_OBJ_CFM	✓		✓	✓	✓	✓	✓	✓	✓	

Table 6: CSR_BT_SMLC_GET_SML_MSG_OBJ Primitives

Description

To retrieve a SyncML message object from the SyncML-server-side the application sends the CSR_BT_SMLC_GET_SML_MSG_OBJ_REQ to the SMLC. The SyncML server side responses with a CSR_BT_SMLC_GET_SML_MSG_OBJ_CFM. When a successful response for a SyncML message that fits entirely in one response packet is achieved the finalFlag is set, followed by the object body placed at “bodyOffset place”. If the response is too large to fit into one packet the client-side needs to request for the rest of the SyncML message object with the CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_REQ until the confirm signal (CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_CFM) has the finalFlag set. In case the result is different from success (when the result is different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE), the other parameters are invalid and not used.

Parameters

type	Signal identity, CSR_BT_SMLC_GET_SML_MSG_OBJ_REQ/CFM.
mimeType	Parameter setting what mimeType is to be used for the SyncML-data to be transferred. See csr_bt_smlc_prim.h. for appropriate values.
responseCode	The response 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. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h are regarded as reserved and the application should consider them as errors.
lengthOfObject	The total length of the SyncML message object to receive.
finalFlag	Indicates that the body (object) fits the whole SyncML message object or that it is the last part.
bodyLength	The length of the body (SyncML message object).
bodyOffset	Offset of SyncML message object relative to payload.
payloadLength	Number of bytes in payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.7 CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ

Parameters / Primitives	type	responseCode	finalFlag	bodyLength	bodyOffset	payloadLength	*payload	smpOn
CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_REQ	✓							✓
CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_CFM	✓	✓	✓	✓	✓	✓	✓	

Table 7: CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ Primitives

Description

This signal is used if the SyncML message object to pull is too large to fit into one Obex packet. The first Obex-packet is sent in CSR_BT_SMLC_GET_OBJ_CFM, the next part is sent in CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_CFM after sending the CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_REQ signal. The last response has to set the finalFlag parameter.

Parameters

type	Signal identity, CSR_BT_SMLC_GET_NEXT_CHUNK_SML_MSG_OBJ_REQ/CFM.
responseCode	The response 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. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h are regarded as reserved and the application should consider them as errors.
finalFlag	Indicates that the body (SyncML message object) is carrying the last part.
bodyLength	The length of the body (SyncML message object part).
bodyOffset	Offset relative to payload of the SyncML message object body.
payloadLength	Number of bytes in payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.8 CSR_BT_SMLC_PUT_SML_MSG_OBJ & CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ

Parameters / Primitives	type	mimeType	lengthOfObject	finalFlag	*body	bodyLength	maxBodySize	responseCode
CSR_BT_SMLC_PUT_SML_MSG_OBJ_REQ	✓	✓	✓	✓	✓	✓		
CSR_BT_SMLC_PUT_SML_MSG_OBJ_CFM	✓						✓	✓
CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ	✓			✓	✓	✓		
CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_CFM	✓						✓	✓

Table 8: CSR_BT_SMLC_PUT_SML_MSG_OBJ & CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ Primitives

Description

To push a SyncML message object to the SyncML server side, send the CSR_BT_SMLC_PUT_SML_MSG_OBJ_REQ to the SMLC using the given bodySize from “availablePutBodySize” in the CSR_BT_SMLC_CONNECT_CFM) to determine the available data-space for body to place the SyncML data in. The SMLC answers the application with a CSR_BT_SMLC_PUT_SML_MSG_OBJ_CFM with a resultcode “result” and a new “maxBodySize” to be used in an e.g. nextcoming CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ. If the SyncML message can not be hold in one body-signal-packet and therefore needs to be sent as multiple fragments, the application can continue sending CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ after receiving a CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_CFM message. The finalFlag indicates the last part of the body (SyncML message object). In case the result in the confirm is different from success, the other parameters are invalid and not used and the apps can stop sending more of this object (it indicates receiving-error on the receiving side of the link).

Attention if the SyncML packet to send holds more SyncML-messages, each separate message is sent in a separate set of the above described CSR_BT_SMLC_PUT_SML_MSG_OBJ_REQ/ CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ. The end of the SyncML-packet (holding more SyncML messages) transfer is determined by looking for the “SyncML-packet end code inside the “last” SyncML-message” each time a complete SyncML message has been gathered. When finding such a “SyncML-packet end code” the receiving application is able to put together the complete SyncML-packet by gathering the received SyncML-messages up until now.

Parameters

type — Signal identities, CSR_BT_SMLC_PUT_SML_MSG_OBJ_REQ/CFM and CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ/CFM.

mimeType — Indicates what kind of mimeType the body-data refers to, see csr_bt_smlc_prim.h for possible selections.

lengthOfObject — The total length of the object to send (separate SyncML-message size).

Note that if the total length of the object is known in advance, this parameter should be set, as it allows the receiver to quickly terminate transfers requiring too mush space, and also makes progress reporting easier. In the case that the total length of the object is unknown,

	this parameter can be set to 0.
maxBodySize	The maximum length of the available body to send in the next packet (CSR_BT_SMLC_PUT_NEXT_CHUNK_SML_MSG_OBJ_REQ).
finalFlag	Indicates that the body (SyncML message object) fits the whole object or that it is the last part.
bodyLength	The length of the body to be sent.
*body	Pointer to the data to be sent.
responseCode	The response 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. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h are regarded as reserved and the application should consider them as errors.

4.9 CSR_BT_SMLC_ABORT

Primitives \ Parameters	type
CSR_BT_SMLC_ABORT_REQ	✓
CSR_BT_SMLC_ABORT_CFM	✓

Table 9: CSR_BT_SMLC_ABORT Primitives

Description

The CSR_BT_SMLC_ABORT_REQ is used when the apps decides to terminate a multi-packet operation (such as GET/PUT) before it normally ends. The CSR_BT_SMLC_ABORT_CFM indicates that the server has received the abort response and the server is now resynchronized with the client. If the server does not respond the Abort Request or it response with a response code different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, the profile will disconnect the Bluetooth connection and send a CSR_BT_DISCONNECT_IND to the application.

Parameters

type Signal identity, CSR_BT_SMLC_ABORT_REQ/CFM.

4.10 CSR_BT_SMLC_DISCONNECT

Parameters / Primitives	type	normalDisconnect	reasonCode	reasonSupp.ier
CSR_BT_SMLC_DISCONNECT_REQ	✓	✓		
CSR_BT_SMLC_DISCONNECT_IND	✓		✓	✓

Table 10: CSR_BT_SMLC_DISCONNECT Primitives

Description

To disconnect a connection to a server (if any) send a CSR_BT_SMLC_DISCONNECT_REQ to the SMLC. When disconnected, the SMLC will respond with a CSR_BT_SMLC_DISCONNECT_IND. If the link is dropped in the middle of a session, the application will receive a CSR_BT_SMLC_DISCONNECT_IND indicating that the OBEX SyncML transfer session is finished, and is ready for a new one.

The disconnect messages between the OBEX SyncML transfer client and Server is guarded by a timer, thus if for some reason the server do not reply to the OBEX disconnect request within a fixed time interval the Bluetooth connection is disconnected direct. The timeout functionality is per default set to five seconds. The timeout value can be disabled, or change by changing CSR_BT_OBEX_DISCONNECT_TIMEOUT, which is define in csr_bt_user_config.default.h. Note if the value of CSR_BT_OBEX_DISCONNECT_TIMEOUT is change, it will influence all OBEX profiles.

Parameters

type	Signal identity, CSR_BT_SMLC_DISCONNECT_REQ/IND.
normalDisconnect	FALSE defines an Abnormal disconnect sequence where the Bluetooth connection is released directly. TRUE defines a normal disconnect sequence where the OBEX connection is released before the Bluetooth connection.
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. If the reasonSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h 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

4.11 CSR_BT_SMLC_CANCEL_CONNECT

Primitives \ Parameters	type
CSR_BT_SMLC_CANCEL_CONNECT_REQ	✓

Table 11: CSR_BT_SMLC_CANCEL_CONNECT Primitives

Description

The CSR_BT_SMLC_CANCEL_CONNECT_REQ can be used for cancelling an ongoing connect procedure. If the SMLC succeed to cancel the ongoing connection attempt the application will receive a CSR_BT_SMLC_CONNECT_CFM with a response code different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE.

Parameters

type Signal identity, CSR_BT_SMLC_CANCEL_CONNECT_REQ

4.12 CSR_BT_SMLC_SECURITY

Primitives \ Parameters	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_SMLC_SECURITY_OUT_REQ	✓	✓	✓		
CSR_BT_SMLC_SECURITY_OUT_CFM	✓			✓	✓
CSR_BT_SMLC_SECURITY_IN_REQ	✓	✓	✓		
CSR_BT_SMLC_SECURITY_IN_CFM	✓			✓	✓

Table 12: CSR_BT_SMLC_SECURITY 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 p 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_SMLC_SECURITY_OUT_REQ/CFM and CSR_BT_SMLC_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. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h 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
SyncML OBEX Binding Version 1.1 15 Feb 2002	[SMLOBEXBINDING]
SyncML Over Bluetooth Version 0.9 8 Aug 2001	[SMLOVERBLUETOOTH]
SyncML Representation Protocol, version 1.0.1	[SMLREPPROT]
IrDA Object Exchange Protocol - IrOBEX Version 1.2 18 March 1999	[OBEX]
Specifications for Ir Mobile Communications (IrMC) Version 1.1 01 March 1999	[IRMC]
CSR Synergy Bluetooth, SC – Security Controller API Description	[SC]

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
SDS	Service Discovery Server
SIG	Special Interest Group
SMLS	OBEX SyncML Server
SMLC	OBEX SyncML Client
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.