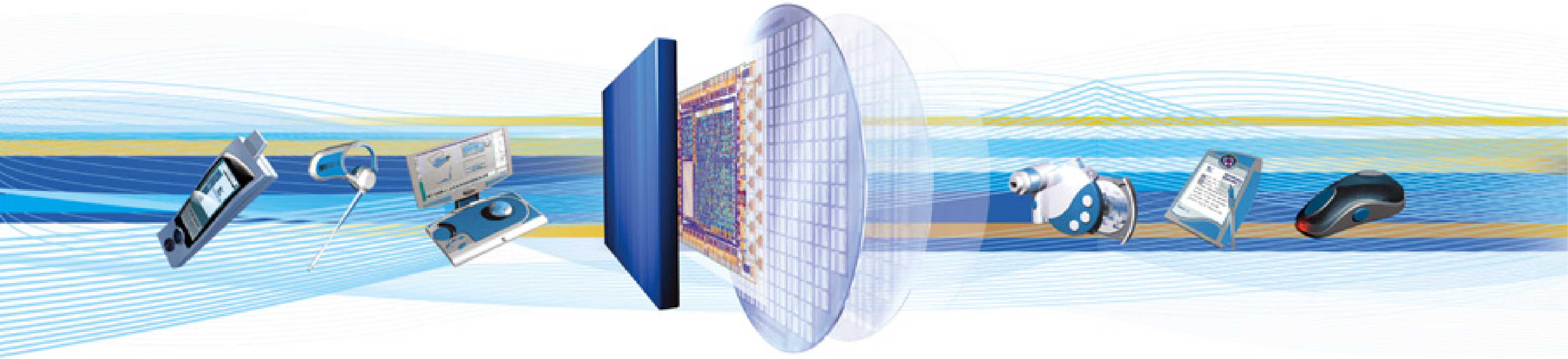
CSR Synergy®

CSR Synergy Bluetooth 18.2.0

OBEX SyncML Server

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 OBEX SyncML Server side (SMLS). The SMLS conforms to the server side of the OBEX SyncML binding description ref. [SMLOBEXBINDING]. The OBEX-link functions as OBEX-server-side in this present version.

1.2 Assumptions

The following assumptions and preconditions are made in the following:

- There is a secure and reliable transport between the OBEX-Syncml-server "profile, i.e. SMLS and the application
- The SMLS shall only handle one request at the time
- Bonding (pairing) is NOT handled by the SMLS

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 ref: [SMLREPPROT]

The OBEX SyncML Server (SMLS) must be activated by the application. When it is activated it is able to provide the application with incoming SyncML-msg-object (put-requests from the SyncML-client) and enables the application to send outgoing SyncML-msg-objects (get-requests from the SyncML-client).

- The SMLS provides Service Discovery handling
- The SMLS is handling the interpretation of the OBEX packet.

The application is responsible for handling the indications from the SMLS and sending the correct responses. The response codes used are described in the IrOBEX Specification [OBEX]. The SMLS does not check and verify the data in the responses. Thus, it is the responsibility of the application to make sure that data follows the appropriate standards and formats. For further details on this subject please consult ref. [SMLOBEXBINDING], [SMLREPPROT] and [OBEX].

2.2 Reference Model

The SMLS interfaces to the Connection Manager (CM) and to the application.

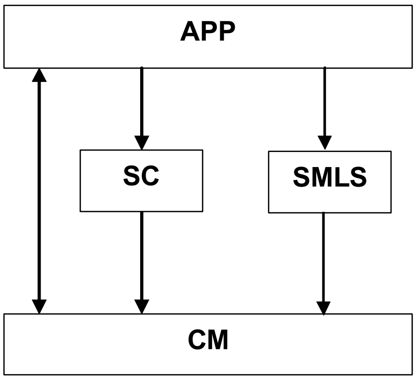


Figure 1: Reference model

2.3 Sequence Overview

The SMLS starts up being in IDLE state. When the application activates SMLS, the server enters ACTIVATE state and is ready to handle incoming requests. The server remains in this state until deactivated by the application. When deactivated it re-enters IDLE state.

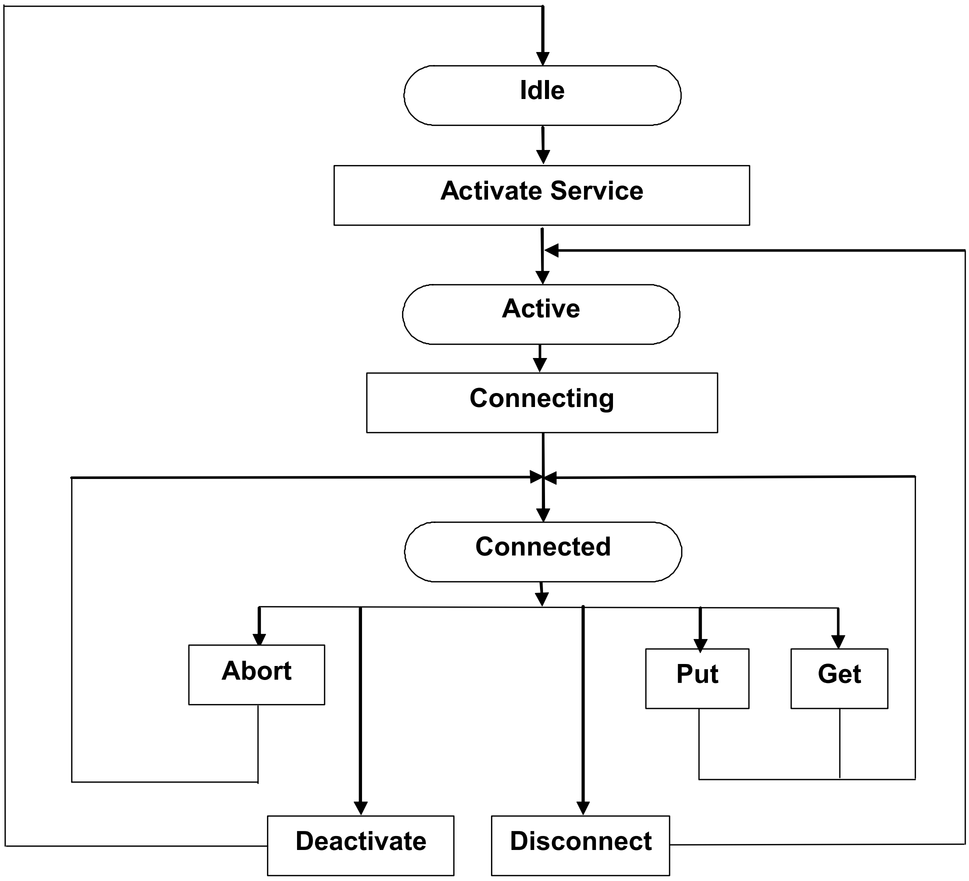


Figure 2: SMLS state diagram

3 Interface Description

3.1 Activation

Sending a CSR_BT_SMLS_ACTIVATE_REQ to the SMLS activates the SMLS. The SMLS then registers a Service Record, in the Service Discovery Server, and makes it connectable. After this the SMLS sends back a CSR_BT_SMLS_ACTIVATE_CFM to the application and the SMLS is now ready to handle incoming requests.

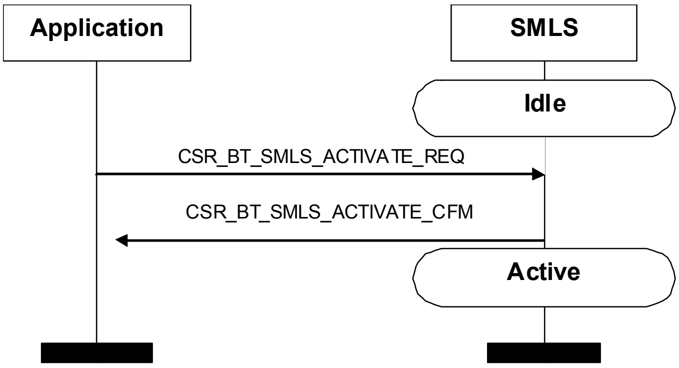


Figure 3: SMLS activation

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.

3.2 Deactivation

Sending a CSR_BT_SMLS_DEACTIVATION_REQ to the SMLS can deactivate the SMLS. This procedure can take some time depending on the current SMLS activity. When deactivated, the SMLS confirms the deactivation with a CSR_BT_SMLS_DEACTIVATE_CFM message.

Any transaction in progress will be terminated immediately when this message is received by the SMLS.

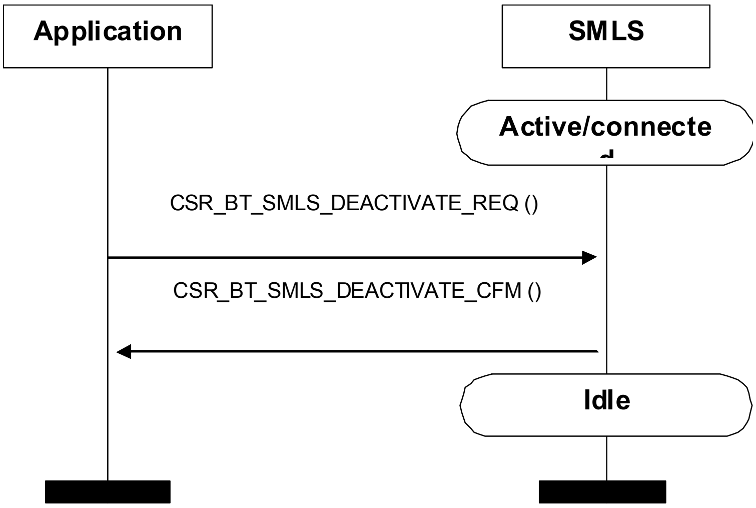


Figure 4: SMLS deactivation

3.3 Connect

When the SyncML client is making a connect against the server the first message the application receives is CSR_BT_SMLS_CONNECT_IND, this message has an “obexPeerMaxPacketSize” parameter indicating the maximum Obex packet size (bodysize) which the application can send down to the SMLS in the body of one message response. Another important parameter “targetService” is indicating what kind of Sync-service the client asks for and the Application-Server has to response on this request information when sending the connect-response.

The application responses with a CSR_BT_SMLS_CONNECT_RES message with an appropriate result code. This message has the parameter obexMaxPacketSize being the maximum packet (body) size that the application wants to receive from the client. There is a defined CSR_BT_MAX_OBEX_SIGNAL_LENGHT and the application must use this in the response. This value is calculated from the defined CSR_BT_MAX_OBEX_SIGNAL_LENGTH and both defines are placed in the file csr_bt_obex.h. The value can be between 255 bytes – 64K bytes – 1, see definition in ref. [OBEX]. If the packet size is large it’s optimizing for faster data transfer, but the disadvantage will be use of big memory blocks. The memory use will increase with the packet size. The CSR_BT_SMLS_CONNECT_RES also has to hold the acknowledge information for the targetService (who) which has to be the same as the one placed in the clients request.

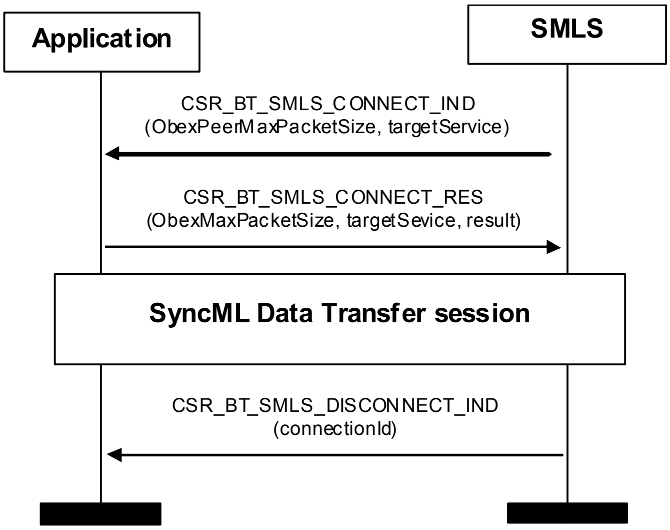

Application
SMLS
CSR_BT_SMLS_CONNECT_IND
(ObexPeerMaxPacketSize, targetService)
CSR_BT_SMLS_CONNECT_RES
(ObexMaxPacketSize, targetSevice, result)
SyncML Data Transfer session
CSR_BT_SMLS_DISCONNECT_IND
(connectionId)

Figure 5: Connection handling

3.4 Pushing SyncML Message Objects

When SyncML message objects are received by the SMLS, it passes them on to the application in a CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND message. The application responds with a CSR_BT_SMLS_PUT_SML_MSG_OBJ_RES, which contains the result of the “put”. If the client side sends the body part fragmented the SMLS sends additional indications (CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_IND’s) until the finalFlag parameter is set. This indicates end of body to the application.

A SyncML packet can hold one or more SyncML messages. The application has to look for the (SyncML-packet end code) in a SyncML message received and first when this parameter is found the complete SyncML-packet has been received. This means if no (SyncML-packet-end) code is found yet another PUT/PUT_NEXT will be initiated by the Client-side to send the next SyncML message to complete the SyncML-packet.

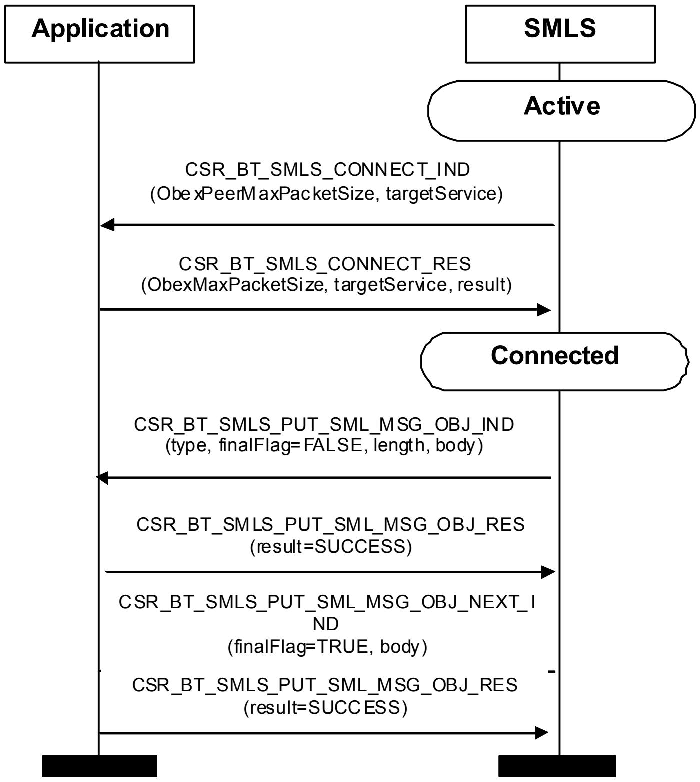

Application
SMLS
Active
CSR_BT_SMLS_CONNECT_IND
(ObexPeerMaxPacketSize, targetService)
CSR_BT_SMLS_CONNECT_RES
(ObexMaxPacketSize, targetService, result)
Connected
CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND
(type, finalFlag=FALSE, length, body)
CSR_BT_SMLS_PUT_SML_MSG_OBJ_RES
(result=SUCCESS)
CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_IND
(finalFlag=TRUE, body)
CSR_BT_SMLS_PUT_SML_MSG_OBJ_RES
(result=SUCCESS)

Figure 6: Incoming SyncML message handling

3.5 Pulling SyncML Message Objects

When the SMLS receives a request to send a SyncML-packet to the client side, it sends a CSR_BT_SMLS_GET_OBJ_IND message to the application with the mimeType parameter set to the requested one by the client-side. The application responds with a CSR_BT_SMLS_GET_OBJ_RES with the appropriate result code. If the application wants to fragment the “body” information due to memory considerations it can set the finalFlag to FALSE and will hence receive CSR_BT_SMLS_GET_OBJ_NEXT_INDs until the finalFlag is set to TRUE in the following CSR_BT_SMLS_GET_OBJ_NEXT_RES.

A SyncML-packet can contain one or more SyncML message(s). The client side looks for the “SyncML-packet end code”. When a GET/GET_NEXT session has ended (transferring one SyncML message) the client-side application has to determine if the complete SyncML packet has been transferred or just yet a SyncML message inside a packet has been received. When finding the “SyncML-packet-end-code” the Client-application side knows it has received all packet-data and stops starting yet a GET/GET_NEXT session for the next SyncML message. But as long as this code is missing the Client-side will start yet a GET/GET_NEXT session to get the next SyncML message following in the SyncML packet.

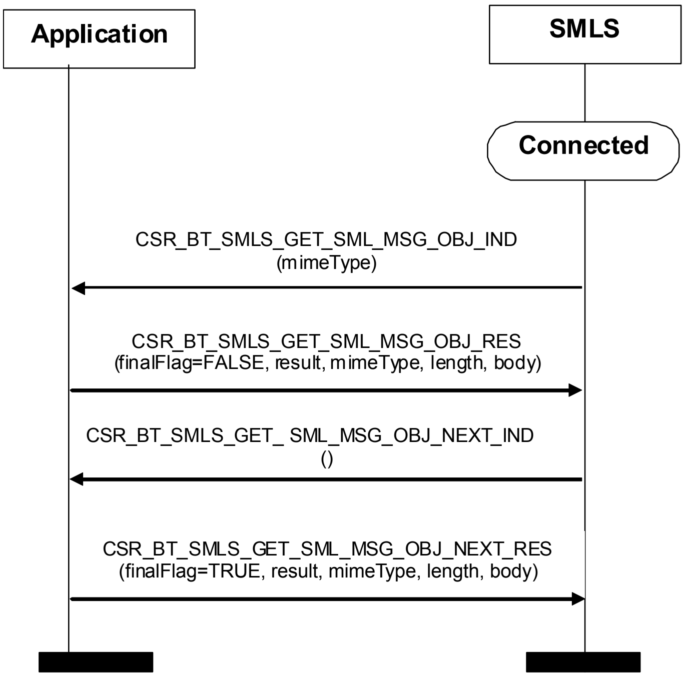

Application
SMLS
Connected
CSR_BT_SMLS_GET_SML_MSG_OBJ_IND
(mimeType)
CSR_BT_SMLS_GET_SML_MSG_OBJ_RES
(finalFlag=FALSE, result, mimeType, length, body)
CSR_BT_SMLS_GET_ SML_MSG_OBJ_NEXT_IND
()
CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_RES
(finalFlag=TRUE, result, mimeType, length, body)

Figure 7: Outgoing SyncML message handling

3.6 Aborting a SyncML Message Transfer Session

When the SMLS receives a request to abort the current session, it sends the CSR_BT_SMLS_ABORT_IND to the application. When the SyncML-server (application) receives a CSR_BT_SMLS_ABORT_IND, it shall discontinue the on-going session clearing the present received/transmitted data and reset the status of the data-compartment and then just wait for the next action. Now it is up to the SyncML-client-side to decide what to do. It can pickup the aborted transferring session again or it may just disconnect the OBEX-connection and may re-initiate the complete (disrupted) SyncML-session over OBEX.

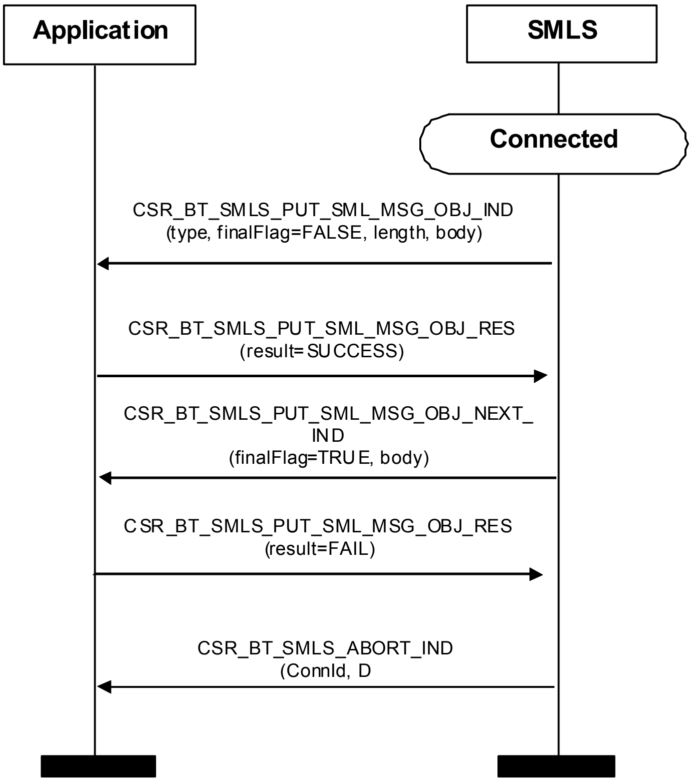

Application
SMLS
Connected
CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND
(type, finalFlag=FALSE, length, body)
CSR_BT_SMLS_PUT_SML_MSG_OBJ_RES
(result= SUCCESS)
CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_IND
(finalFlag=TRUE, body)
CSR_BT_SMLS_PUT_SML_MSG_OBJ_RES
(result=FAIL)
CSR_BT_SMLS_ABORT_IND
(ConnId, D

Figure 8: Aborting an in-coming SyncML message

3.7 Disconnecting

When the SMLS receives a request to close down the OBEX-connection it sends the CSR_BT_SMLS_DISCONNECT_IND to the application. When the application receives the CSR_BT_SMLS_DISCONNECT_IND the complete data-transfer-session has ended.

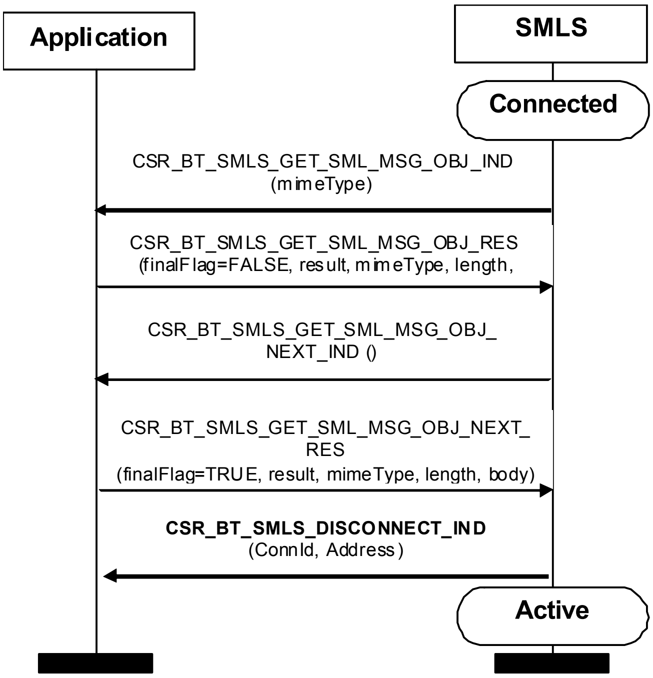

Application
SMLS
Connected
CSR_BT_SMLS_GET_SML_MSG_OBJ_IND
(mimeType)
CSR_BT_SMLS_GET_SML_MSG_OBJ_RES
(finalFlag=FALSE, result, mimeType, length,
CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_IND ()
CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_RES
(finalFlag=TRUE, result, mimeType, length, body)
CSR_BT_SMLS_DISCONNECT_IND
(ConnId, Address)
Active

Figure 9: Outgoing SyncML message handling

3.8 Payload Encapsulated Data

3.8.1 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. As can be seen, the offset contains the number of bytes within the *payload* where the information begins. Similarly, the body data can be retrieved using the following:

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

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

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

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

3.8.2 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 Server Primitives

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

4.1 List of All Primitives

Primitives:	Reference:
CSR_BT_SMLS_ACTIVATE_REQ	See section 4.2
CSR_BT_SMLS_ACTIVATE_CFM	See section 4.2
CSR_BT_SMLS_DEACTIVATE_REQ	See section 4.3
CSR_BT_SMLS_DEACTIVATE_CFM	See section 4.3
CSR_BT_SMLS_CONNECT_IND	See section 4.4
CSR_BT_SMLS_CONNECT_RES	See section 4.4
CSR_BT_SMLS_AUTHENTICATE_REQ	See section 4.5
CSR_BT_SMLS_AUTHENTICATE_CFM	See section 4.5
CSR_BT_SMLS_AUTHENTICATE_IND	See section 4.5
CSR_BT_SMLS_AUTHENTICATE_RES	See section 4.5
CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND	See section 4.6
CSR_BT_SMLS_PUT_SML_MSG_OBJ_RES	See section 4.6
CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_IND	See section 4.7
CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_RES	See section 4.7
CSR_BT_SMLS_GET_SML_MSG_OBJ_IND	See section 0
CSR_BT_SMLS_GET_SML_MSG_OBJ_RES	See section 0
CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_IND	See section 4.9
CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_RES	See section 4.9
CSR_BT_SMLS_ABORT_IND	See section 4.10
CSR_BT_SMLS_DISCONNECT_IND	See section 4.11
CSR_BT_SMLS_SECURITY_IN_REQ	See section 4.12
CSR_BT_SMLS_SECURITY_IN_CFM	See section 4.12

Table 1: List of all primitives

4.2 CSR_BT_SMLS_ACTIVATE

Parameters Primitives	type	appHandle	resultCode	resultSupplier	obexMaxPacketSize	windowSize	srmEnable
CSR_BT_SMLS_ACTIVATE_REQ	✓	✓				✓	✓
CSR_BT_SMLS_ACTIVATE_CFM	✓		✓	✓	✓		

Table 2: CSR_BT_SMLS_ACTIVATE Primitives

Description

This signal is used for activating the SMLS and making it accessible from a remote device. The process includes:

1. Register the OBEX SyncML Server service in the service discovery database.
2. Enabling page scan.
3. Sending activate confirm to inform caller of activation.

The SMLS is ready for application requests when the CSR_BT_SMLS_ACTIVATE_CFM has been sent, so the application has to wait for this signal before any other requests are send. The SMLS will remain activated until a CSR_BT_SMLS_DEACTIVATE_REQ is received.

Parameters

type — Signal identity, CSR_BT_SMLS_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.

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

obexMaxPacketSize — To control the maximum allowed obex packet size the application can receive. There is a define CSR_BT_MAX_OBEX_SIGNAL_LENGHT (in csr_bt_obex.h) to be used for this value, the max allowed value is 64K bytes – 1.

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_SMLS_DEACTIVATE

Parameters Primitives	type
CSR_BT_SMLS_DEACTIVATE_REQ	✓
CSR_BT_SMLS_DEACTIVATE_CFM	✓

Table 3: CSR_BT_SMLS_DEACTIVATE Primitives

Description

This signal deactivates the SMLS. The service cannot be re-activated until after the application has received a CSR_BT_SMLS_DEACTIVATE_CFM.

The service will no longer be visible to inquire devices and the inquiry and page scan may be stopped (depending on the fact if other services are available or not). The OBEX SyncML Server service is removed from the service discovery database.

The signal also stops any ongoing transaction.

Parameters

type	Signal identity, CSR_BT_SMLS_DEACTIVATE_REQ/CFM.

4.4 CSR_BT_SMLS_CONNECT

Parameters / Primitives	type	connectionId	targetService	obexPeerMaxPacketSize	deviceAddr	responseCode	length	count	btConnId
CSR_BT_SMLS_CONNECT_IND	✓	✓	✓	✓	✓		✓	✓	✓
CSR_BT_SMLS_CONNECT_RES	✓	✓	✓			✓			

Table 4: CSR_BT_SMLS_CONNECT Primitives

Description

This signal is indicating that a SyncML client is starting a SyncML session. The application can then accept or deny the session by the result and has also to return the connectionId and the targetService-codes received in the indication.

Parameters

type — Signal identity, CSR_BT_SMLS_CONNECT_IND/RES.

connectionId — Is the connection Id for this session, The Syncml client side will use this Id in the next –coming requests.

obexPeerMaxPacketSize — The maximum OBEX packet size being allowed to send to the client application.

deviceAddr — The Bluetooth address which is connected to the device

targetService — Parameter indicating what service the connection is to support: Either SyncML synchronization service OR SyncML device management service. The valid codes are defined (in csr_bt_smls_prim.h)

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.

length — The length parameter contains the length in bytes of the bodies of all the objects that the sender plans to send. Note this length cannot be guarantee correct, so while the value may be useful for status indicators and resource reservations, the application should not die if the length is not correct.
If 0 this parameter were not included in the received OBEX Connect Request packet.

count — Count is use to indicate the number of objects that will be sent by the sender during this connection.
If 0 this parameter were not included in the received OBEX Connect Request packet.

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

4.5 CSR_BT_SMLS_AUTHENTICATE

Parameters Primitives	type	options	realmLength	* realm	deviceAddr	*password	passwordLength	*userId
CSR_BT_SMLS_AUTHENTICATE_REQ	✓		✓	✓		✓	✓	✓
CSR_BT_SMLS_AUTHENTICATE_CFM	✓							
CSR_BT_SMLS_AUTHENTICATE_IND	✓	✓	✓	✓	✓			
CSR_BT_SMLS_AUTHENTICATE_RES	✓					✓	✓	✓

Table 5: CSR_BT_SMLS_AUTHENTICATE Primitives

Description

The request signal is used when the SyncML server side wants to OBEX authenticate the client. The application has to send a password or pin number in the *smlsChalPassword to authenticate the client with. The authentication of the client is only a success if the application receives a CSR_BT_SMLS_AUTHENTICATE_CFM.

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

Parameters

type

Signal identity, CSR_BT_SMLS_AUTHENTICATE_REQ/-CFM/-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_SMLS_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

Note in this release version the 'realmLength' parameter in the CSR_BT_SMLS_AUTHENTICATE_IND is always set to 0x0000 and in the CSR_BT_SMLS_AUTHENTICATE_REQ the 'realmLength' 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 CsrPfree 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 in the CSR_BT_SMLS_AUTHENTICATE_IND is always set to NULL and in the CSR_BT_SMLS_AUTHENTICATE_REQ the ‘realm’ is ignored right now.

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

For CSR_BT_SMLS_AUTHENTICATE_REQ it is challenge password of the OBEX authentication.
For CSR_BT_SMLS_AUTHENTICATE_RES it is response password of the OBEX authentication.

passwordLength

The actual length of the challenge password.

*userId

Pointer to zero terminated (ASCII)-data containing the userId for the authentication.

Note in the CSR_BT_SMLS_AUTHENTICATE_REQ the ‘userId’ is ignored right now.

4.6 CSR_BT_SMLS_PUT_SML_MSG_OBJ

Parameters / Primitives	type	connectionId	finalFlag	lengthOfObject	mimeType	bodyLength	bodyOffset	*payload	payloadLength	responseCode	smpOn
CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND	✓	✓	✓	✓	✓	✓	✓	✓	✓		
CSR_BT_SMLS_PUT_SML_MSG_OBJ_RES	✓	✓								✓	✓

Table 6: CSR_BT_SMLS_PUT_SML_MSG_OBJ Primitives

Description

The SMLS passes incoming SyncML-message-objects on to the application with the CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND signal. The application then has to gather and store the SyncML-message-objects where appropriate. The result of the store operation is given to the SMLS with the CSR_BT_SMLS_PUT_SML_MSG_OBJ_RES signal. The result can contain error codes corresponding to the reason for failure.

The application can also authenticate the client-side before accepting the operation, which is done with the CSR_BT_SMLS_AUTHENTICATE_REQ.

Parameters

type	Signal identity, CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND/-RES.
connectionId	The connection Id for this session, the SyncML client-side will use this Id in the request.
finalFlag	Indicates that the body (object) fits the whole object or that it's the last part.
lengthOfObject	The total length of the object to receive.
mimeType	Indicates what kind of mimeType the body-data refers to. See (csr_bt_smls_prim.h) for possible selections
bodyLength	The length of the body (object).
bodyOffset	Offset relative to the payload of the object data itself.
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.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

CSR Synergy Bluetooth 18.2.0 OBEX SyncML Server API

4.7 CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT

Primitives \ Parameters	type	connectionId	finalFlag	mimeType	bodyLength	bodyOffset	*payload	payloadLength	responseCode	smpOn
CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_IND	✓	✓	✓	✓	✓	✓	✓	✓		
CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_RES	✓	✓							✓	✓

Table 7: CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT Primitives

Description

The SMLS passes incoming SyncML-message-objects on to the application with the CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND signal. In case the SyncML-message-object is too large to fit into one OBEX packet, the first part is in the CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND and the next part(s) of the object will appear in the CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_IND(s) until the finalFlag parameter is set. When the finalFlag parameter is set ONE SINGLE complete SyncML-message has been received.

If a SyncML-package contains more SyncML-messages the above described session/sequence (starting with CSR_BT_SMLS_PUT_SML_MSG_OBJ_IND followed by CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_IND(s)) is repeated for each individual SyncML-message contained by the complete SyncML-package. The application can determine whether it is the last SyncML-message (or whether there are more SyncML-messages to receive) in the SyncML-package by looking for the “Final Element” in the SyncML-message, each time a complete SyncML-message has been received. If the “Final element” is present in the SyncML-message it is the last SyncML-message in the SyncML-package to receive.

Parameters

type
Signal identity, CSR_BT_SMLS_PUT_SML_MSG_OBJ_NEXT_IND/-RES.

connectionId
Is the connection Id for this session, the SyncML client-side will use this Id in the requests.

finalFlag
Indicates that the body (SyncML-message-object) fits the whole object or that it is the last part.

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

bodyLength
The length of the body (SyncML-message-object).

bodyOffset
Offset relative to payload for the actual body data.

*payload
OBEX payload data. Offsets are relative to this pointer.

payloadLength
Number of bytes in the payload structure.

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.

smpOn Reserved for future use. Set to FALSE.

4.8 CSR_BT_SMLS_GET_SML_MSG_OBJ

Parameters / Primitives	type	connectionId	mimeType	finalFlag	responseCode	lengthOfObject	bodyLength	*body	smpOn
CSR_BT_SMLS_GET_SML_MSG_OBJ_IND	✓	✓	✓						
CSR_BT_SMLS_GET_SML_MSG_OBJ_RES	✓	✓		✓	✓	✓	✓	✓	✓

Table 8: CSR_BT_SMLS_GET_SML_MSG_OBJ Primitives

Description

For the SyncML-client side to retrieve a SyncML-message-object from the SyncML server-side which type is specified by the mimeType parameter in the CSR_BT_SMLS_GET_SML_MSG_OBJ_IND signal, the application responses with a CSR_BT_SMLS_GET_SML_MSG_OBJ_RES. When a successful response for a SyncML-message of mimeType that fits entirely in one response packet is achieved the finalFlag is set, followed by the SyncML-message-object itself in the body. If the SyncML-message in the response is too large to fit in one body (which means it requires multiple requests: (CSR_BT_SMLS_GET_SMLMSG_NEXT_IND)), only the last response has the finalFlag set.

The application, must remember the SyncML-message to transmit until the last response (CSR_BT_SMLS_GET_SMLMSG_NEXT_RES) has been sent. In case the result is different from success, the other parameters are invalid and not used.

If the SyncML-package to be sent in the response contains more SyncML-messages the application must remember these SyncML-messages and send them subsequently/individually in yet the next-coming get-session(s) (CSR_BT_SMLS_GET_SMLMSG_IND - CSR_BT_SMLS_GET_SMLMSG_NEXT_IND(s)) still initiated by the client-side. As long as a send (responded) SyncML-message does NOT hold the “Final element” yet a SyncML-message-object still has to be transferred and therefore the application should wait to be requested by the client-side. When a SyncML-message responded holds the SyncML-packet-“Final Element”, it is the last SyncML-message to be transferred to complete the SyncML-package transfer.

The application can also authenticate the client before accepting the operation, which is done with the CSR_BT_SMLS_AUTHENTICATE_REQ.

Parameters

type Signal identity, CSR_BT_SMLS_GET_SML_MSG_OBJ_IND/-RES.

connectionId The connection Id for this session, the SyncML client side will use this Id in the request.

mimeType Parameter indicating what kind of mimeType the body-data to respond with has to refer too see csr_bt_smls_prim.h for possible selections

finalFlag Indicate that the body (SyncML-message object) fits in one response packet or the last part of multiple responses.

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 object to send.

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.

bodyLength — The length of the body (object).

*body — The SyncML-message object itself. "body" is a pointer to the object (or part of the object).

smpOn — Reserved for future use. Set to FALSE.

4.9 CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT

Parameters / Primitives	type	connectionId	mimeType	finalFlag	responseCode	bodyLength	*body	smpOn
CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_IND	✓	✓	✓					
CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_RES	✓	✓		✓	✓	✓	✓	✓

Table 9: CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT Primitives

Description

To retrieve multiple parts of a SyncML-message-object from the server, the first obex-packet is the CSR_BT_SMLS_GET_SML_MSG_OBJ_RES, the next obex-packet is the CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_RES after receiving the CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_IND signal. The last response; i.e. all data in the object syncml-msg (CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_RES) has to set the parameter finalFlag. The application must remember the next-coming parts of the SyncML-message here dealing with multiple chunks of the SyncML-message object.

If a SyncML-package containing more SyncML-messages is to be retrieved the above described get; get-next session is to be repeated until all contained SyncML-messages have been transferred. To identify the last SyncML-message in a SyncML-package look for the SyncML-“Final element”. If the Final-element is present in a SyncML-message, it is the last SyncML-message in the SyncML-package.

Parameters

type — Signal identity, CSR_BT_SMLS_GET_SML_MSG_OBJ_NEXT_IND/-RES.

connectionId — Is the connection Id for this session, the SyncML client side will use this Id in the request.

finalFlag — Indicate that the body (SyncML-message-object) fits in one response packet or the last part of multiple responses.

mimeType — Parameter indicating what kind of mimeType the body-data to respond with has to refer too see csr_bt_smls_prim.h for possible selections

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.

bodyLength — The length of the body (Chunk of SyncML-message-object).

*body — The SyncML-message-object itself. "body" is a pointer to the object. (or part of the object)

smpOn — Reserved for future use. Set to FALSE.

4.10 CSR_BT_SMLS_ABORT

Primitives \ Parameters	type	connectionId	descriptionOffset	descriptionLength	payloadLength	*payload
CSR_BT_SMLS_ABORT_IND	✓	✓	✓	✓	✓	✓

Table 10: CSR_BT_SMLS_ABORT Primitives

Description

This signal is indicating that the SyncML OBEX client has terminated an operation (such as PUT or GET), before it would normally end the session.

Parameters

type	Signal identity, CSR_BT_SMLS_ABORT_IND.
connectionId	The connection Id for this session, the SyncML client side will use this Id in the request.
descriptionOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the reason for the abort. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string
descriptionLength	Length of the abort description string.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX payload data. Offsets are relative to this pointer.

4.11 CSR_BT_SMLS_DISCONNECT

Primitives \ Parameters	type	connectionId	deviceAddr	reasonCode	reasonSupplier
CSR_BT_SMLS_DISCONNECT_IND	✓	✓	✓	✓	✓

Table 11: CSR_BT_SMLS_DISCONNECT Primitives

Description

This signal is indicating that the SyncML transfer session via OBEX is finished.

Parameters

type — Signal identity, CSR_BT_SMLS_DISCONNECT_IND.

connectionId — The connection Id for this session, the SyncML client side will use this Id in the request.

deviceAddr — The Bluetooth address which refers to the device that disconnects.

reasonCode — The reason code of the operation. Possible values depends 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 are the respective prim.h files or csr_bt_obex.h is 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

4.12 CSR_BT_SMLS_SECURITY_IN

Parameters / Primitives	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_SMLS_SECURITY_IN_REQ	✓	✓	✓		
CSR_BT_SMLS_SECURITY_IN_CFM	✓			✓	✓

Table 12: CSR_BT_SMLS_SECURITY_IN Primitives

Description

Applications that wish to change the enforcement to a specific profile security level, i.e. authentication, encryption and/or authorisation, can use this API to set up the security level for *new* connections. Note that this API is for the local device only and can be used from within any state.

The *CSR_BT_SECURITY_IN_REQ* signal sets up the security level for new incoming connections. Already established or pending connections are not altered.

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_SMLS_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. CM – Connection Manager API Description, doc. no. api-0101-cm	[CM]
CSR Synergy Bluetooth, SC – Security Controller API Description, Document no. api-0102-sc	[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
SMLC	OBEX SyncML Client
SMLS	OBEX SyncML Server
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.