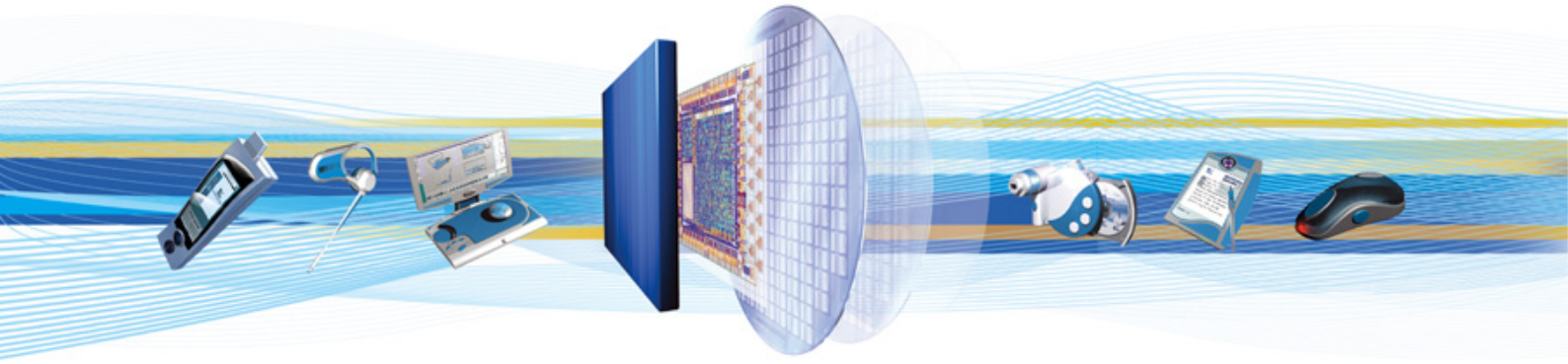
CSR Synergy Bluetooth 18.2.0

OBEX Sync Server API

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 Sync Server (SYNCS). The SYNCS conforms to the server side of the Synchronization Profile, ref. [SYNC].

1.2 Assumptions

The following assumptions and preconditions are made:

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

It is assumed that the reader has a thorough knowledge of the IrDA Object Exchange Protocol – IrOBEX specification and the Specifications for IrMobile Communications (IrMC).

2 Description

2.1 Introduction

The OBEX Sync Server (SYNCS) provides the following services to the application:

The SYNCS provides Service Discovery handling. It sets the Service Discovery Record according to the activation parameters.

The SYNCS is handling the interpretation of the OBEX packet.

The SYNCS is implemented with the specification from IrMC enabling IrMC Object exchange enabling easy exchange of:

- business cards
- calendar and to-do items
- text messages
- short notes

The SYNCS is responsible for interpretation of the information exchange operations based on the OBEX-commands. The SYNCS is implemented with the information exchange level 4 - this is primarily used for data synchronization operation.

The Application is responsible for handling the request from the SYNCS and sending the response with correct data (object) as described in the IrMC specification. The SYNCS is not checking if the data is packet correctly with white spaces in the right places, for details see the ref. [SYNC], [OBEX] and [IRMC].

2.2 Reference Model

The SYNCS interfaces to the Connection Manager (CM).

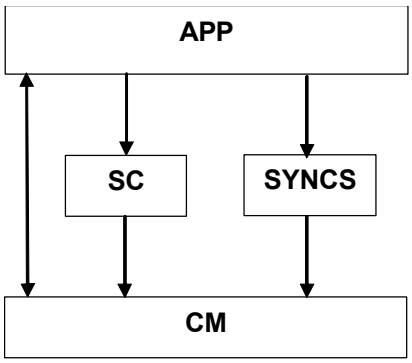


Figure 1: Reference model

2.3 Sequence Overview

The SYNCS starts up being in IDLE state. When the application activates SYNCS, the server enters ACTIVE 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.

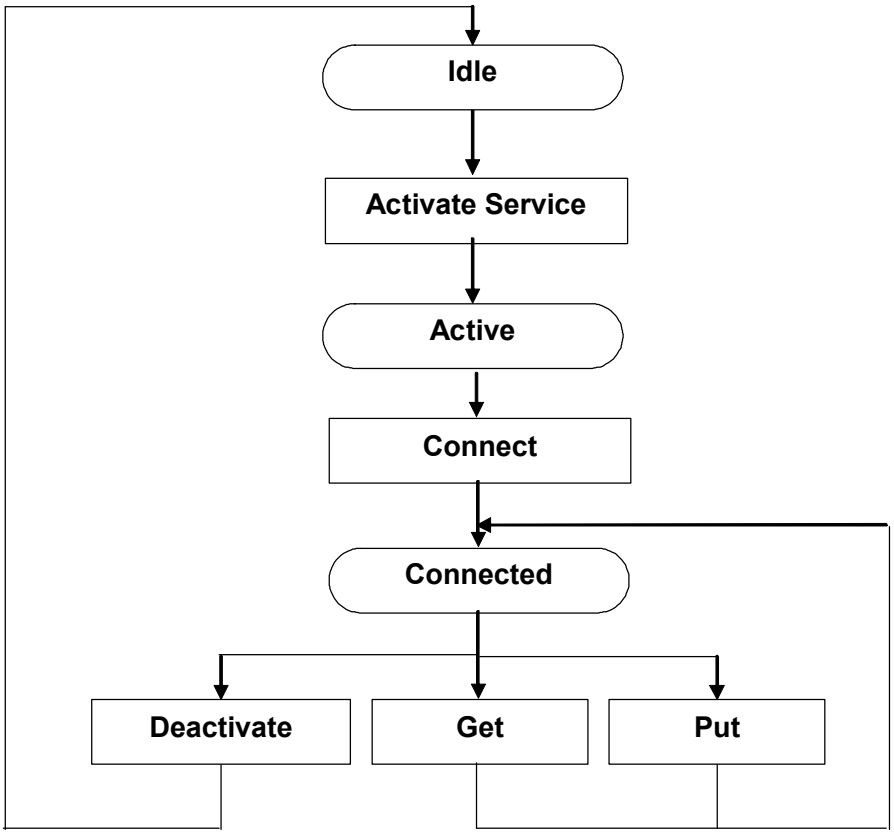


Figure 2: SYNCS state diagram

3 Interface Description

3.1 Activation and Deactivation

Sending a CSR_BT_SYNCS_ACTIVATE_REQ to the SYNCS activates the SYNCS. The SYNCS then registers a Service Record, which contains the supported formats list, in the Service Discovery Server. The SYNCS is now ready to handle incoming requests and making a synchronisation against a client.

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.

Sending a CSR_BT_SYNCS_DEACTIVATE_REQ to the SYNCS can deactivate the SYNCS. This procedure may take some time depending on the current SYNCS activity. When deactivated, the SYNCS confirm the deactivation with a CSR_BT_SYNCS_DEACTIVATE_CFM message.

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

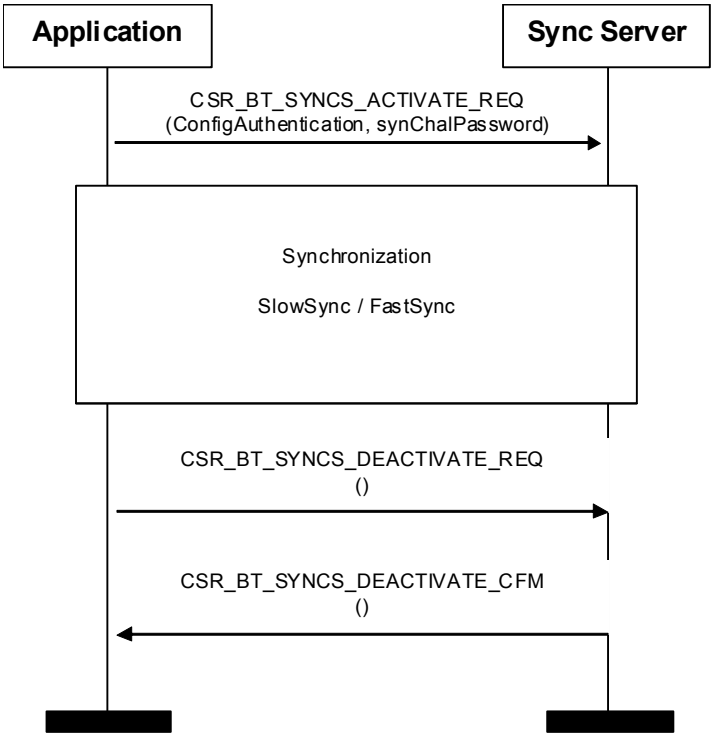


Figure 3: SYNCS activation and deactivation

3.2 Synchronization

The Bluetooth® Synchronization profile is based on the IrMC synchronization standard. IrMC defines a set of text objects representing synchronization information such as which records have been modified or deleted, and it defines how the actual record data is exchanged in a device independent manner. Using Bluetooth's Generic Object Exchange Profile, these IrMC objects are then exchanged between devices to synchronize their data.

There are two basic types of synchronization: Slow Sync and Fast Sync. There are variants, but this document concentrates on these two types.

"Slow Sync" is when two devices synchronize for the first time, or when the changes to one or both of the devices Object Stores are too numerous to be processed efficiently. The subsequent synchronization is Fast Sync if the change log describes all the changes that have occurred in the Object Store on the server.

This Sync server is implemented with the Change Counter as the Sync Anchor. The Change Counter is increased by 1 each time any change is made to an Object Store. A change is defined as an Add, Modify or Delete operation on a single object within the Object Store.

The following steps show the sequence of events and transfer of objects that happen when synchronizing two devices.

1. A Sync client connects to a Sync server.
2. The client gets the IrMC device info object to determine if it has ever synchronized with this server before.
3. The client gets the IrMC change counter object to find out if any changes have been made to the server's database.
4. The client gets the IrMC change log object to find out which records have changed in the database.
5. The client removes any deleted records from its copy of the database.
6. The client gets any modified/added records from the server and adds them to its database.
7. The client is running the sync engine, and puts the "result from the engine" modified records to the server.

3.2.1 Synchronization Example

A user with a Bluetooth® enabled device such as a PDA or cell phone (acting as an IrMC server) comes in range of a PC (the IrMC client). When the PC notices that the user's device is in range, it automatically starts the synchronization process.

First the PC establishes a Bluetooth® connection to the server (allowing for any authentication and bonding if needed) and then gets the *change log* (on the first sync the change log is number 0.log). If the PC does not recognize the Serial Number or Database ID then it gets the device info object from the server. This object tells the PC which object formats the server is supporting.

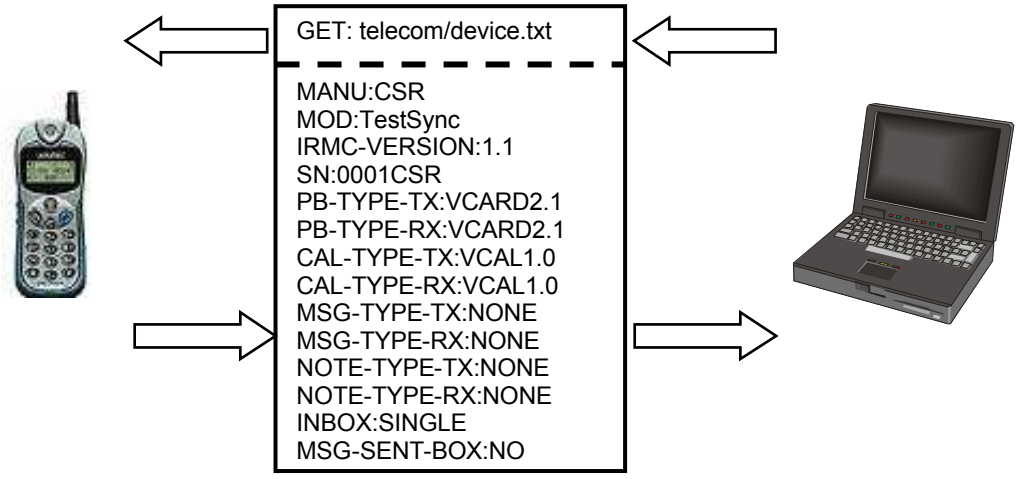


Figure 4: Example of a device info object, support vCard and vCalendar objects

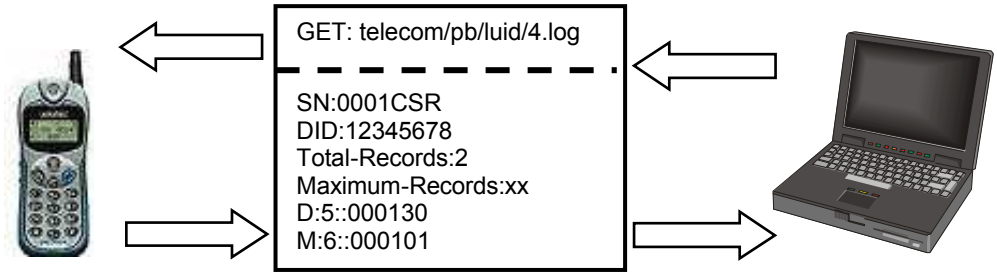


Figure 5: Get change log

Assuming that the PC has at one time been synchronized with this client, the next step is to get the IrMC server's *change counter object* to find out if there have been any changes since the last sync. The number in the change counter object is incremented each time there is a change to the event database. If the change counter was 4, during the latest synchronization, but this time it is 6 on the server, the device can tell that two new changes have been made.

This object (shown in Figure 5) tells the PC that object ID 000130 has been deleted, and that object ID 000101 has been modified. The PC now knows that it needs to get the new modified object by getting the record with ID 000101 from the server and then make the corresponding modification to it's internal version of the phonebook.

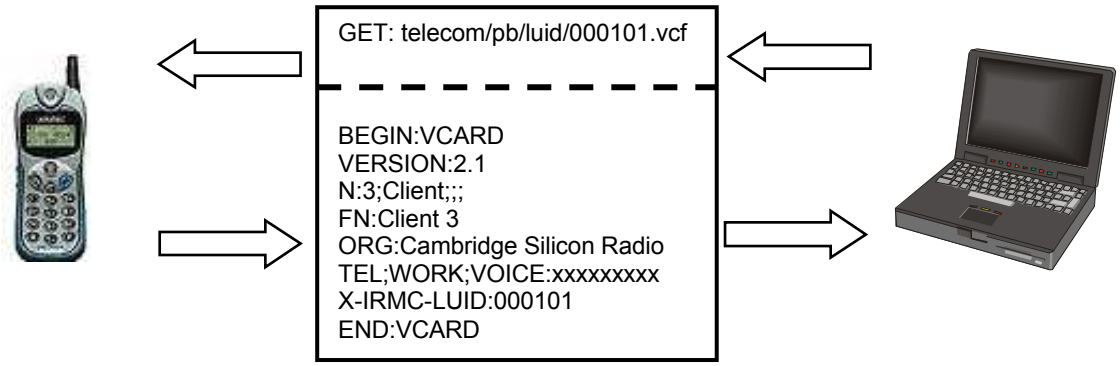


Figure 6: Get phonebook object

Now that the PC has the modifications, the only thing left to do is to delete object ID 000130 from its version of the phonebook. The PC then just stores away the current change counter value for the next synchronization and disconnects.

Figure 7 is an illustration of how the signal flow between the Sync server and the application will be with the synchronization example just described above.

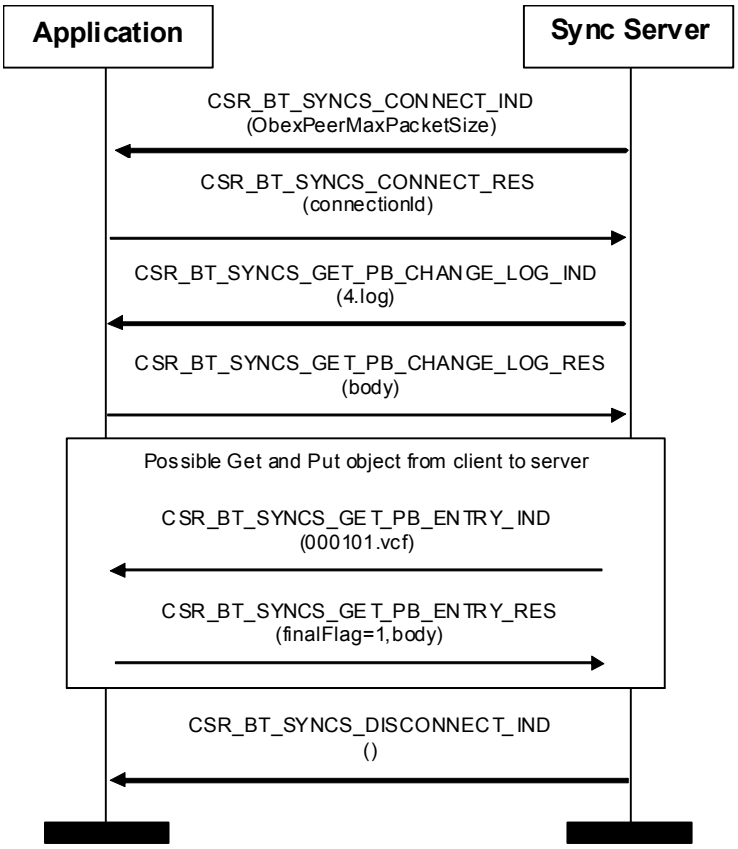


Figure 7: Signal flow on synchronization example

3.2.2 Interpretation of IrMC Requests

The following section shows the signal flow between the Sync server and the Application, there are four examples. This Sync server is implemented with the interpreter of the IrMC request (telecom/xxxxxxxx), the interpretation is made on the phonebook (vCard) and the Calendar objects. All other objects can also be synchronized with this Sync server, but the Application has to interpret the IrMC request. Two of the four figures in this section include phonebook interpretation, and the last two have no interpretation.

The first Sync example is a slow sync with support of a phonebook.

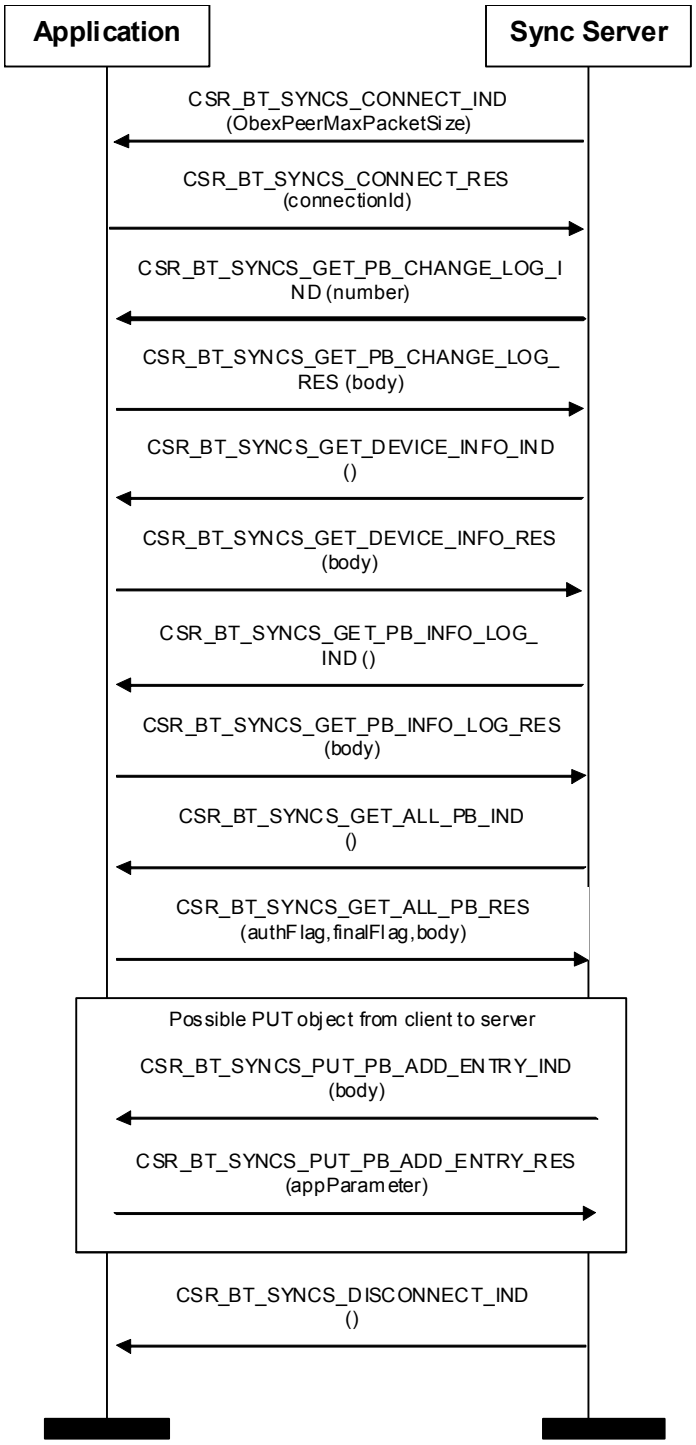


Figure 8: Slow Sync Phonebook signal flow

Figure 9 is showing the fast sync with phonebook interpretation.

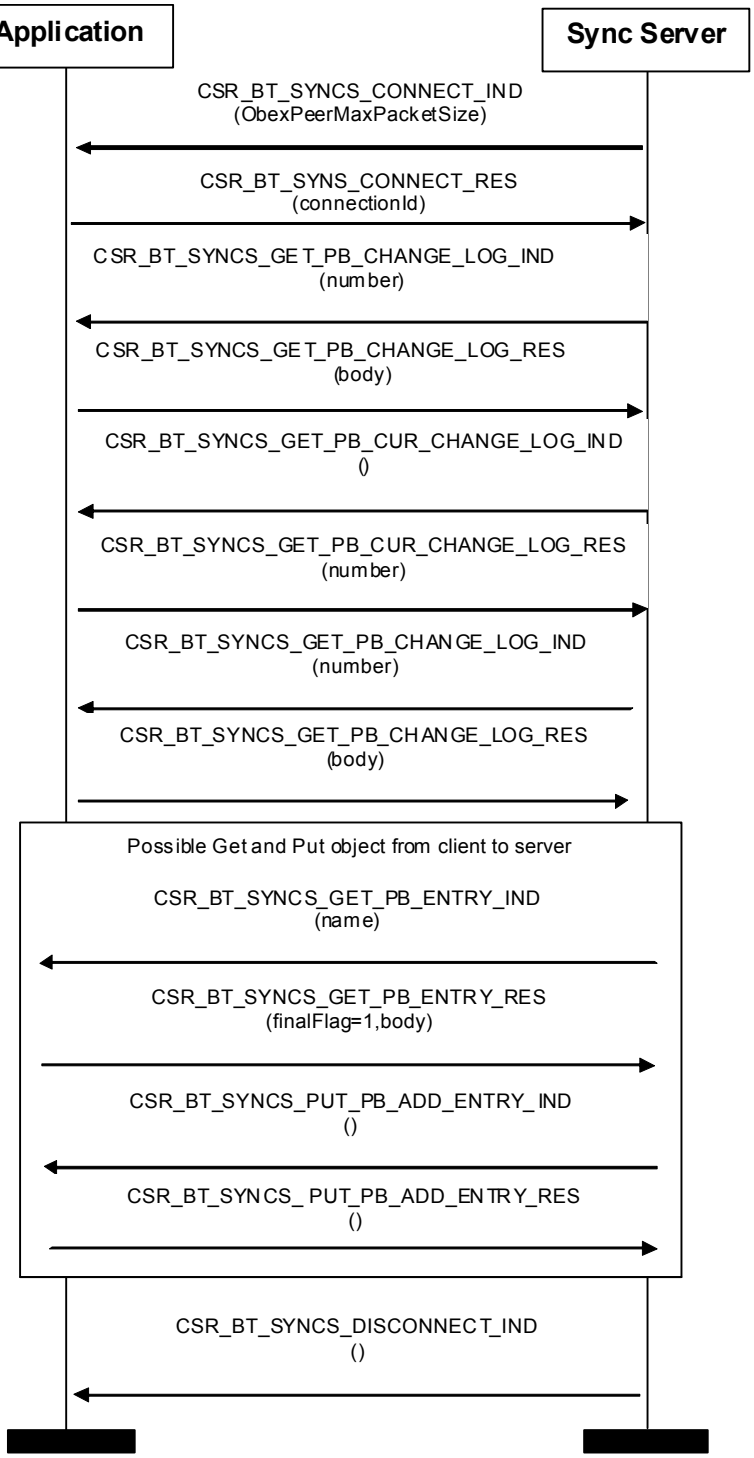


Figure 9: Fast Sync Phonebook signal flow

The signals Get and Put for a object as shown in the box on Figure 9, are also similar to the signals in Figure 10, where the objects are to large to be dealt with in one packet. For more details on the signals, see description of the specified primitives, see section 4.1.

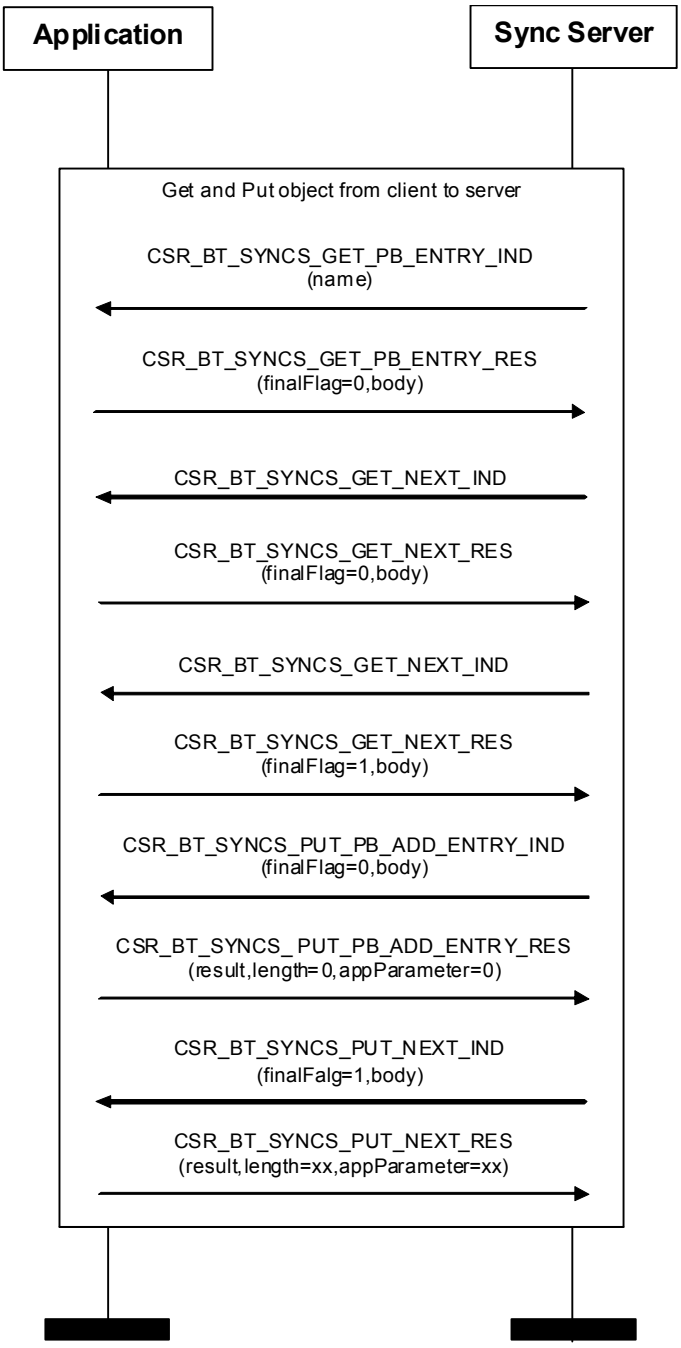


Figure 10: Get and Put large object

Figure 11 is showing the slow sync with Notes objects, the application has to interpret the IrMC “telecom/xxx commands. This example is also with OBEX authentication requested from the Sync client.

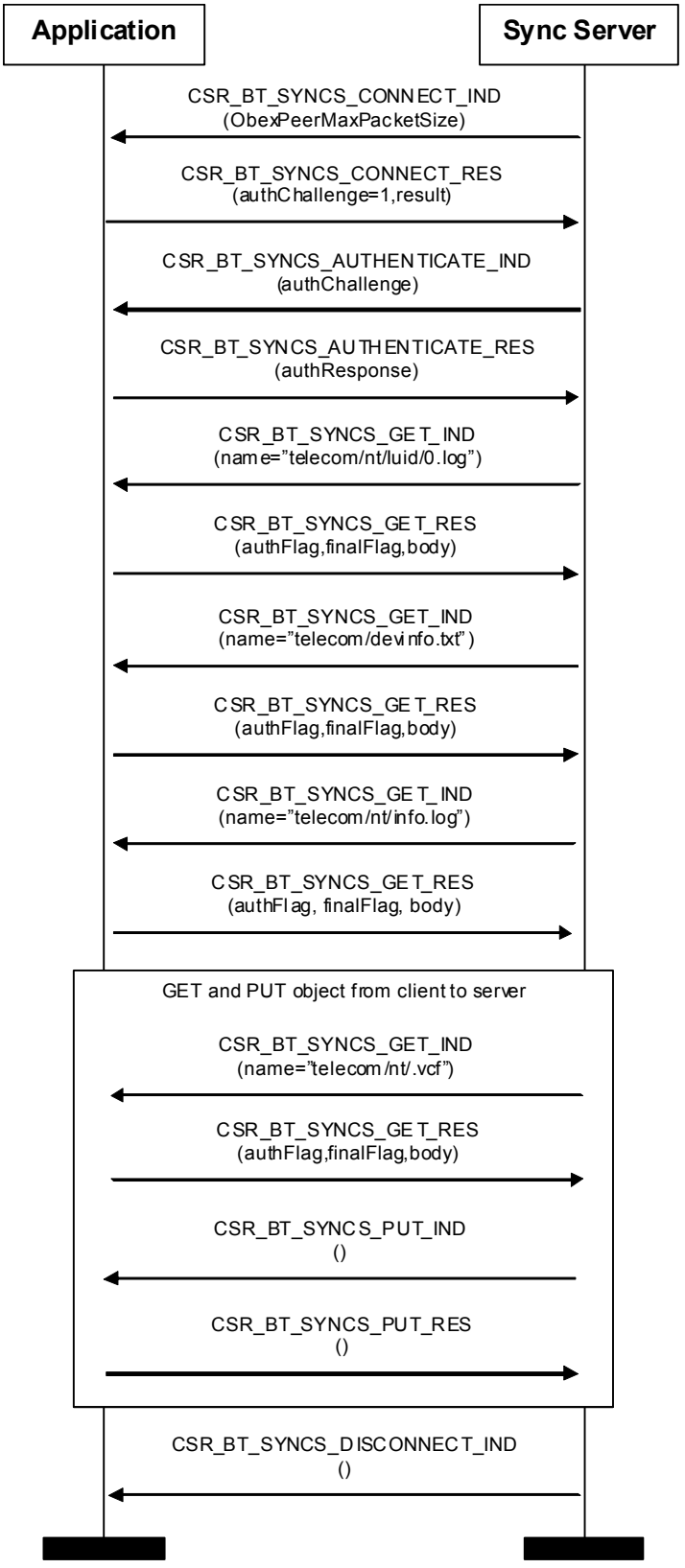


Figure 11: General Slow Sync with client authentication

Figure 12 is showing the fast sync with notes objects and no interpretation (for objects other than phonebook and calendar) from the Sync server.

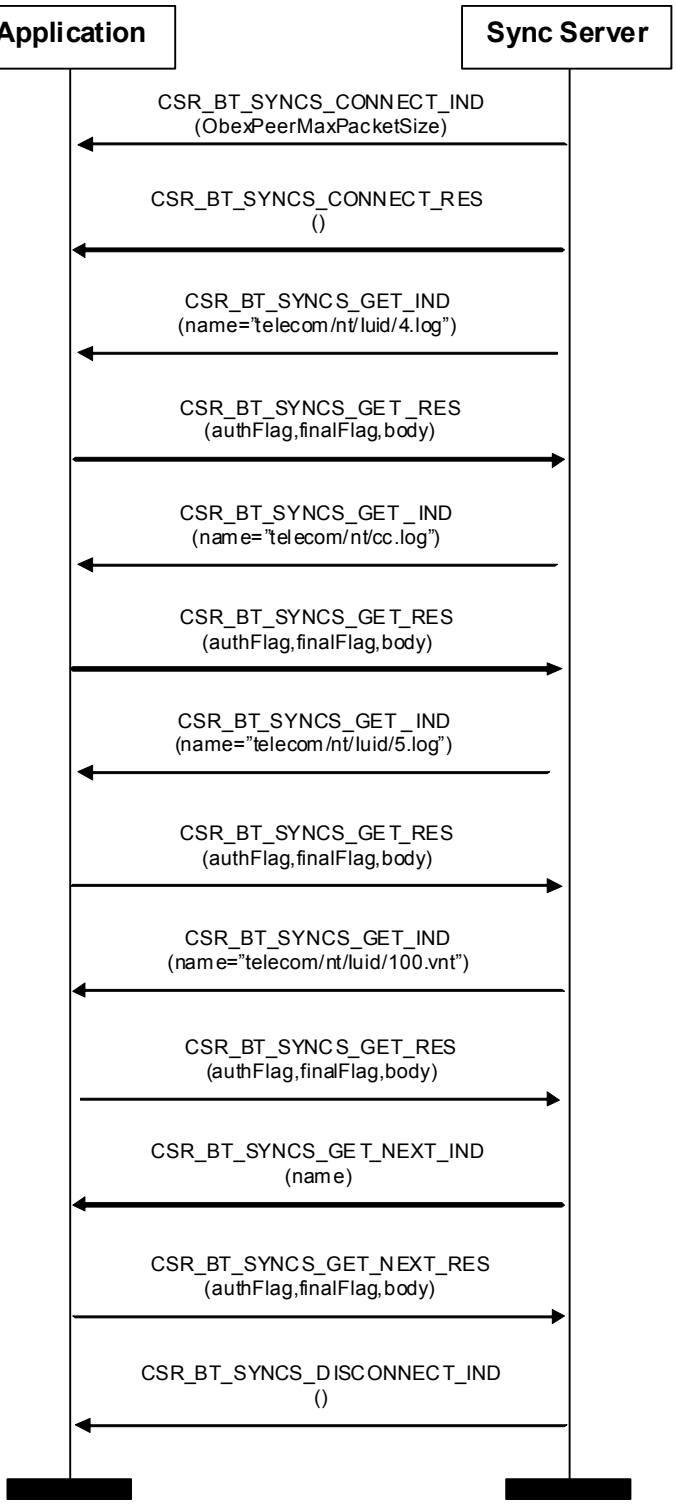


Figure 12: General Fast Sync

3.3 Payload Encapsulated Data

3.3.1 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:

```
CsrCommonPrim  type;
CsrUint8           result;
CsrUint16      ucs2nameOffset;
CsrUint16      bodyOffset;
CsrUint16      bodyLength;
CsrUint16      payloadLength;
CsrUint8       *payload;
```

In this example, two offset parameters can be found, namely *ucs2nameOffset* and *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  *ucs2name;
ucs2name = (CsrUint8*)(primitive->payload + primitive->ucs2nameOffset);
```

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:

```
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.

If the `bodyOffset` or the `bodyLength` is 0 (zero) this means that the signal does not contain any body. The same holds when the `payloadLength` is 0 (zero), which means that there is not payload.

3.3.2 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 Sync 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_syncs_prim.h file.

4.1 List of All Primitives

Primitives:	Reference:
CSR_BT_SYNCS_ACTIVATE_REQ	See section 4.2
CSR_BT_SYNCS_DEACTIVATE_REQ	See section 4.3
CSR_BT_SYNCS_DEACTIVATE_CFM	See section 4.3
CSR_BT_SYNCS_CONNECT_IND	See section 4.4
CSR_BT_SYNCS_CONNECT_RES	See section 4.4
CSR_BT_SYNCS_AUTHENTICATE_IND	See section 4.5
CSR_BT_SYNCS_AUTHENTICATE_RES	See section 4.5
CSR_BT_SYNCS_GET_IND	See section 4.6
CSR_BT_SYNCS_GET_RES	See section 4.6
CSR_BT_SYNCS_GET_NEXT_IND	See section 4.7
CSR_BT_SYNCS_GET_NEXT_RES	See section 4.7
CSR_BT_SYNCS_PUT_IND	See section 4.8
CSR_BT_SYNCS_PUT_RES	See section 4.8
CSR_BT_SYNCS_PUT_NEXT_IND	See section 4.9
CSR_BT_SYNCS_PUT_NEXT_RES	See section 4.9
CSR_BT_SYNCS_GET_DEVICE_INFO_IND	See section 4.10
CSR_BT_SYNCS_GET_DEVICE_INFO_RES	See section 4.10
CSR_BT_SYNCS_GET_PB_CHANGE_LOG_IND	See section 4.11
CSR_BT_SYNCS_GET_PB_CHANGE_LOG_RES	See section 4.11
CSR_BT_SYNCS_GET_PB_CUR_CHANGE_LOG_IND	See section 4.12
CSR_BT_SYNCS_GET_PB_CUR_CHANGE_LOG_RES	See section 4.12
CSR_BT_SYNCS_GET_PB_ INFO_LOG_IND	See section 4.13
CSR_BT_SYNCS_GET_PB_ INFO_LOG_RES	See section 4.13
CSR_BT_SYNCS_GET_PB_ ENTRY_IND	See section 4.14
CSR_BT_SYNCS_GET_PB_ ENTRY_RES	See section 4.14
CSR_BT_SYNCS_GET_PB_ ALL_IND	See section 4.15
CSR_BT_SYNCS_GET_PB_ ALL_RES	See section 4.15
CSR_BT_SYNCS_PUT_PB_ ENTRY_IND	See section 4.16
CSR_BT_SYNCS_PUT_PB_ ENTRY_RES	See section 4.16
CSR_BT_SYNCS_PUT_PB_ ADD_ENTRY_IND	See section 4.17
CSR_BT_SYNCS_PUT_PB_ ADD_ENTRY_RES	See section 4.17
CSR_BT_SYNCS_GET_CAL_CHANGE_LOG_IND	See section 4.18
CSR_BT_SYNCS_GET_CAL_CHANGE_LOG_RES	See section 4.18
CSR_BT_SYNCS_GET_CAL_CUR_CHANGE_LOG_IND	See section 4.19
CSR_BT_SYNCS_GET_CAL_CUR_CHANGE_LOG_RES	See section 4.19
CSR_BT_SYNCS_GET_CAL_INFO_LOG_IND	See section 4.20
CSR_BT_SYNCS_GET_CAL_INFO_LOG_RES	See section 4.20
CSR_BT_SYNCS_GET_CAL_ENTRY_IND	See section 4.21
CSR_BT_SYNCS_GET_CAL_ENTRY_RES	See section 4.21
CSR_BT_SYNCS_PUT_CAL_ENTRY_IND	See section 4.22
CSR_BT_SYNCS_PUT_CAL_ENTRY_RES	See section 4.22
CSR_BT_SYNCS_PUT_CAL_ADD_ENTRY_IND	See section 4.23
CSR_BT_SYNCS_PUT_CAL_ADD_ENTRY_RES	See section 4.23
CSR_BT_SYNCS_GET_CAL_ALL_IND	See section 4.24
CSR_BT_SYNCS_GET_CAL_ALL_RES	See section 4.24
CSR_BT_SYNCS_ABORT_IND	See section 4.25
CSR_BT_SYNCS_DISCONNECT_IND	See section 4.26
CSR_BT_SYNCS_SECURITY_IN_REQ	See section 4.27
CSR_BT_SYNCS_SECURITY_IN_CFM	See section 4.27

Table 1: List of all primitives

4.2 CSR_BT_SYNCS_ACTIVATE

Primitives \ Parameters	type	appHandle	obexMaxPacketSize	windowSize	srmEnable
CSR_BT_SYNCS_ACTIVATE_REQ	✓	✓	✓	✓	✓

Table 2: CSR_BT_SYNCS_ACTIVATE Primitives

Description

This signal is used for activating the SYNCS and make it connectable. The process includes:

1. Register the OBEX Sync Server service in the service discovery database.
2. Enabling page scan.

The SYNCS will remain activated until a CSR_BT_SYNCS_DEACTIVATE_REQ is received.

Parameters

type	Signal identity, CSR_BT_SYNCS_ACTIVATE_REQ.
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.
supportedFormats	The formats being supported in this application. This applies both to incoming and outgoing objects. The following values are possible (defined in csr_bt_syncs_prim.h): • VCARD_2_1_SUPPORT • VCARD_3_0_SUPPORT • VCAL_1_0_SUPPORT • ICAL_2_0_SUPPORT • VNOTE_SUPPORT • VMESSAGE_SUPPORT • ANY_TYPE_SUPPORT Multiple formats can be combined by means of binary OR'ing values.
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_SYNCS_DEACTIVATE

Primitives \ Parameters	type
CSR_BT_SYNCS_DEACTIVATE_REQ	✓
CSR_BT_SYNCS_DEACTIVATE_CFM	✓

Table 3: CSR_BT_SYNCS_DEACTIVATE Primitives

Description

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

The service will no longer be connectable. The OBEX Sync Server service is removed from the service discovery database.

The signal will stop any ongoing transaction.

Parameters

type Signal identity, CSR_BT_SYNCS_DEACTIVATE_REQ/CFM.

4.4 CSR_BT_SYNCS_CONNECT

Primitives \ Parameters	type	connectionId	obexPeerMaxPacketSize	deviceAddr	responseCode	length	count	btConnId
CSR_BT_SYNCS_CONNECT_IND	✓	✓	✓	✓		✓	✓	✓
CSR_BT_SYNCS_CONNECT_RES	✓	✓			✓			

Table 4: CSR_BT_SYNCS_CONNECT Primitives

Description

This signal is indicating that a SYNC client is starting synchronization. The application can then either accept or deny with the result and has to return the connectionId received in the indication. With the parameter authFlag it is possible to control if the application wants to use OBEX authentication against the client.

Parameters

type	Signal identity, CSR_BT_SYNCS_CONNECT_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
obexPeerMaxPacketSize	The maximum obex packet size allowed for sending to the client application.
deviceAddr	The Bluetooth address which is connected to the device
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
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_SYNCS_AUTHENTICATE

Parameters / Primitives	type	options	realmLength	* realm	deviceAddr	*password	passwordLength	*userId
CSR_BT_SYNCS_AUTHENTICATE_IND	✓	✓	✓	✓	✓			
CSR_BT_SYNCS_AUTHENTICATE_RES	✓					✓	✓	✓
CSR_BT_SYNCS_AUTHENTICATE_REQ	✓		✓	✓		✓	✓	✓
CSR_BT_SYNCS_AUTHENTICATE_CFM	✓							

Table 5: CSR_BT_SYNCS_AUTHENTICATE Primitives

Description

This signal is used when the SYNC client wants to OBEX authenticate the SYNC server. The application has to response with a password or pin number in the syncsResponsePassword and userId for the client to identify the proper password.

Parameters

type — Signal identity, CSR_BT_SYNCS_AUTHENTICATE_IND/RES.

options — Challenge information of type CsrUint8.

If bit 0 is set it means that the application must response with a user Id in a CSR_BT_SYNCS_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 is in the CSR_BT_SYNCS_AUTHENTICATE_IND always set to 0x0000 and it is right now not used in the CSR_BT_SYNCS_AUTHENTICATE_REQ

* 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 is always set to NULL in the CSR_BT_SYNCS_AUTHENTICATE_IND and it is not used in the CSR_BT_SYNCS_AUTHENTICATE_REQ.

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. This is a pointer, which shall be allocated by the application.
passwordLength	The length of the response password.
*userId	Pointer to a zero terminated string (ASCII) containing the userId for the authentication. This is a pointer, which shall be allocated by the application.

4.6 CSR_BT_SYNCS_GET

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	totalObjectSize	ucs2nameOffset	bodyLength	*body	payloadLength	*payload	smpOn
CSR_BT_SYNCS_GET_IND	✓	✓				✓			✓	✓	
CSR_BT_SYNCS_GET_RES	✓	✓	✓	✓	✓		✓	✓			✓

Table 6: CSR_BT_SYNCS_GET Primitives

Description

To retrieve an object from the server specified by the name parameter in the CSR_BT_SYNCS_GET_IND signal, the server responses with a CSR_BT_SYNCS_GET_RES. When a successful response for an object that fits entirely in one response packet is achieved the finalFlag is set, followed by the object body. If the response is large enough to require multiple requests (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application, but remember the request (name) until the last response (CSR_BT_SYNCS_GET_NEXT_RES). In case the result is different from success, the other parameters are invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	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.
ucs2nameOffset	Offset to a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyLength	The length of the body (object).

*body	The object itself. "body" is a CsrUint8 pointer to the object.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX payload. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.7 CSR_BT_SYNCS_GET_NEXT

Parameters / Primitives	type	connectionId	finalFlag	responseCode	bodyLength	*body	SMOoN
CSR_BT_SYNCS_GET_NEXT_IND	✓	✓					
CSR_BT_SYNCS_GET_NEXT_RES	✓	✓	✓	✓	✓	✓	✓

Table 7: CSR_BT_SYNCS_GET_NEXT Primitives

Description

To retrieve multiple objects from the server, the first packet is the CSR_BT_SYNCS_GET_RES, the next packet is sent to the CSR_BT_SYNCS_GET_NEXT_RES after receiving the CSR_BT_SYNCS_GET_NEXT_IND signal. The last response has to set the parameter finalFlag. The application must remember the name parameter in the first CSR_BT_SYNCS_GET_IND, then dealing with multiple objects.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_NEXT_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
bodyLength	The length of the body (object).
*body	The object itself "body" is a CsrUint8 pointer to the object.
smpOn	Reserved for future use. Set to FALSE.

4.8 CSR_BT_SYNCS_PUT

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	lengthOfObject	ucs2nameOffset	bodyOffset	bodyLength	appParameterLength	*appParameter	payloadLengbth	*payload	smpOn
CSR_BT_SYNCS_PUT_IND	✓	✓	✓		✓	✓	✓	✓			✓	✓	
CSR_BT_SYNCS_PUT_RES	✓	✓		✓					✓	✓			✓

Table 8: CSR_BT_SYNCS_PUT Primitives

Description

The SYNCS passes incoming objects on to the application with the CSR_BT_SYNCS_PUT_IND signal. The application can then store the objects. The result of the store operation is given to the SYNCS with the CSR_BT_SYNCS_PUT_RES signal. The result can contain error codes corresponding to the reason for failure.

The parameter name indicates the object that has been modified. If the name is present but the length is CSR_BT_NO_BODY_HEADER and finalFlag is set, then the body is empty and that's indicate that the object is deleted from the client. If the name parameter is empty then it is a new object from the client. The appParameter is only used if the finalFlag was set in the indication, the appParameter must be packed as the IrOBEX specification specifies.

Parameters

type	Signal identity, CSR_BT_SYNCS_PUT_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (object) fits the whole object or that it is the last part.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
lengthOfObject	The total length of the object to send.
ucs2nameOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function "CsrUsc2String2Utf8" can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyLength	The length of the body.
bodyOffset	Offset relative to payload where the actual "body" data can be retrieved.

appParameterLength	The length of the appParameter (object).
*appParameter	The application response parameter is a CsrUint8 pointer to the parameter.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX data payload. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.9 CSR_BT_SYNCS_PUT_NEXT

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	bodyLength	bodyOffset	appParameterLength	*appParameter	payloadLength	*payload	smpOn
CSR_BT_SYNCS_PUT_NEXT_IND	✓	✓	✓		✓	✓			✓	✓	
CSR_BT_SYNCS_PUT_NEXT_RES	✓	✓		✓			✓	✓			✓

Table 9: CSR_BT_SYNCS_PUT_NEXT Primitives

Description

The SYNCS passes incoming objects on to the application with the CSR_BT_SYNCS_PUT_IND signal. In case the object is too large to fit into one OBEX packet, the first part is in the CSR_BT_SYNCS_PUT_IND and the next part of the object will appear in the CSR_BT_SYNCS_PUT_NEXT_IND until the finalFlag parameter is set. When the last part of the object is received the response must give the client information about the object in the appParameter, the appParameter must be packed as the IrOBEX specification specifies.

Parameters

type	Signal identity, CSR_BT_SYNCS_PUT_NEXT_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (object) fits the whole object or that it is the last part.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
bodyLength	The length of the body (object).
bodyOffset	Offset relative to payload of the object itself.
appParameterLength	The length of the appParameter (object).
*appParameter	The application response parameter is a CsrUint8 pointer to the parameter.
payloadLength	Number of bytes in the payload parameter.
*paylaod	OBEX data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.10 CSR_BT_SYNCS_GET_DEVICE_INFO

Parameters / Primitives	type	connectionId	finalFlag	responseCode	totalObjectSize	bodyLength	*body	smpOn
CSR_BT_SYNCS_GET_DEVICE_INFO_IND	✓	✓						
CSR_BT_SYNCS_GET_DEVICE_INFO_RES	✓	✓	✓	✓	✓	✓	✓	✓

Table 10: CSR_BT_SYNCS_GET_DEVICE_INFO Primitives

Description

The requesting for device information for the device is interpretative into the signal CSR_BT_SYNCS_GET_DEVICE_INFO. The application is returning the device info object in the body. If a successful response for a log that fits entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_DEVICE_INFO_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_DEVICE_INFO_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	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 object itself. "body" is a CsrUint8 pointer to the object.
smpOn	Reserved for future use. Set to FALSE.

4.11 CSR_BT_SYNCS_GET_PB_CHANGE_LOG

Parameters / Primitives	type	connectionId	finalFlag	responseCode	totalObjectSize	ucs2nameOffset	bodyLength	*body	payloadLength	*payload	smpOn
CSR_BT_SYNCS_GET_PB_CHANGE_LOG_IND	✓	✓				✓			✓	✓	
CSR_BT_SYNCS_GET_PB_CHANGE_LOG_RES	✓	✓	✓	✓	✓		✓	✓			✓

Table 11: CSR_BT_SYNCS_GET_PB_CHANGE_LOG Primitives

Description

The requesting for the change log for the phonebook is interpretative into the signal CSR_BT_SYNCS_GET_PB_CHANGE_LOG_IND. The name parameter is indicating which log the client is requesting. The application is returning the change log in the body. If a successful response for a log that fits entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_PB_CHANGE_LOG_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_PB_CHANGE_LOG_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	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.
ucs2nameOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyLength	The length of the body (object).

body	The object itself. "body" is a CsrUint8 pointer to the object.
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.

4.12 CSR_BT_SYNCS_GET_PB_CUR_CHANGE_LOG

Primitives \ Parameters	type	connectionId	responseCode	totalObjectSize	changeCounterLength	*changeCounter	smpOn
CSR_BT_SYNCS_GET_PB_CUR_CHANGE_LOG_IND	✓	✓					
CSR_BT_SYNCS_GET_PB_CUR_CHANGE_LOG_RES	✓	✓	✓	✓	✓	✓	✓

Table 12: CSR_BT_SYNCS_GET_PB_CUR_CHANGE_LOG Primitives

Description

The request for the current change counter for the phonebook is interpretative into the signal CSR_BT_SYNCS_GET_PB_CUR_CHANGE_LOG. The application is returning the current change counter in the body. In case the result is different from success, the parameter (changeCounter) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_PB_CUR_CHANGE_LOG_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	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.
changeCounterLength	The length of the changeCounter (object).
*changeCounter	The object itself. "changeCounter" is a CsrUint8 pointer to the object.
smpOn	Reserved for future use. Set to FALSE.

4.13 CSR_BT_SYNCS_GET_PB_INFO_LOG

Parameters / Primitives	type	connectionId	finalFlag	responseCode	totalObjectSize	bodyLength	*body	smpOn
CSR_BT_SYNCS_GET_PB_INFO_ LOG_IND	✓	✓						
CSR_BT_SYNCS_GET_PB_INFO_ LOG_RES	✓	✓	✓	✓	✓	✓	✓	✓

Table 13: CSR_BT_SYNCS_GET_PB_INFO_LOG Primitives

Description

The requesting for phone book information log is interpretative into the signal CSR_BT_SYNCS_GET_PB_INFO_LOG. The application is returning the information log object in the body. If a successful response for a log that fits entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_PB_INFO_LOG_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_PB_INFO_LOG_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	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 object itself. "body" is a CsrUint8 pointer to the object.
smpOn	Reserved for future use. Set to FALSE.

4.14 CSR_BT_SYNCS_GET_PB_ENTRY

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	totalObjectSize	ucs2nameOffset	bodyLength	*body	payloadLength	*payload	smpOn
CSR_BT_SYNCS_GET_PB_ENTRY_IND	✓	✓				✓			✓	✓	
CSR_BT_SYNCS_GET_PB_ENTRY_RES	✓	✓	✓	✓	✓		✓	✓			✓

Table 14: CSR_BT_SYNCS_GET_PB_ENTRY Primitives

Description

Requesting for a specific Vcard (phone book) entry is interpretative into the signal CSR_BT_SYNCS_GET_PB_ENTRY. The application is returning the object requested with the name parameter in the body. If a successful response for an entry fitting entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_PB_ENTRY_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_PB_ENTRY_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	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.
ucs2nameOffset	Offset relative to payload. A null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyLength	The length of the body (object).
*body	The object itself. "body" is a CsrUint8 pointer to the object.

payloadLength	Number of bytes in the payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.15 CSR_BT_SYNCS_GET_PB_ALL

Parameters / Primitives	type	connectionId	finalFlag	responseCode	bodyLength	*body	smpOn
CSR_BT_SYNCS_GET_PB_ALL_IND	✓	✓					
CSR_BT_SYNCS_GET_PB_ALL_RES	✓	✓	✓	✓	✓	✓	✓

Table 15: CSR_BT_SYNCS_GET_PB_ALL Primitives

Description

The requesting for all the Vcard entries in the (phone book) is interpretative into the signal CSR_BT_SYNCS_GET_PB_ALL. The application is returning the whole phone book in the body. If a successful response for all the entries that fits entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_PB_ALL_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_PB_ALL_IND/RES.
connectionId	The connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
bodyLength	The length of the body (object).
*body	The object itself. "body" is a CsrUint8 pointer to the object.
smpOn	Reserved for future use. Set to FALSE.

4.16 CSR_BT_SYNCS_PUT_PB_ENTRY

Primitives \ Parameters	type	connectionId	finalFlag	lengthOfObject	responseCode	ucs2nameOffset	bodyLength	bodyOffset	appParameterLength	*appParameter	payloadLength	*payload	smpOn
CSR_BT_SYNCS_PUT_PB_ENTRY_IND	✓	✓	✓	✓		✓	✓	✓			✓	✓	
CSR_BT_SYNCS_PUT_PB_ENTRY_RES	✓	✓			✓				✓	✓			✓

Table 16: CSR_BT_SYNCS_PUT_PB_ENTRY Primitives

Description

The incoming of a phonebook object that is known is interpretative into the signal CSR_BT_SYNCS_PUT_PB_ENTRY. The application can then store the object. If the finalFlag is set the whole object is in the body otherwise the application has to remember which object to store and add the incoming body part from the CSR_BT_SYNCS_PUT_NEXT_IND until the finalFlag is set. The result of the store operation is given with the CSR_BT_SYNCS_PUT_PB_ENTRY_RES or CSR_BT_SYNCS_PUT_NEXT_RES. If the object is too large for one packet, the result can contain error codes corresponding to the reason for failure.

The parameter name indicates the object that has been modified. If the length is CSR_BT_NO_BODY_HEADER and finalFlag is set, then the body is empty and that indicates that the object is deleted from the client.
The appParameter is only used if the finalFlag is set in the indication, the appParameter must be packed as the IrOBEX specification specifies.

Parameters

type	Signal identity, CSR_BT_SYNCS_PUT_PB_ENTRY_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (object) fits the whole object or that it's the last part.
lengthOfObject	The total length of the object to send.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
ucs2nameOffset	Offset for a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function "CsrUcs2ByteString2Utf8" can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyLength	The length of the body (object).
bodyOffset	Offset relative to payload of the object itself.

appParameterLength	The length of the body or appParameter (object).
*appParameter	The application response parameter, is a CsrUint8 pointer to the parameter.
payloadLength	Number of bytes in the payload
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.17 CSR_BT_SYNCS_PUT_PB_ADD_ENTRY

Parameters / Primitives	type	connectionId	finalFlag	lengthOfObject	responseCode	ucs2nameOffset	bodyLength	bodyOffset	appParameterLength	*appParameter	payloadLength	*payload	smpOn
CSR_BT_SYNCS_PUT_PB_ADD_ENTRY_IND	✓	✓	✓	✓		✓	✓	✓			✓	✓	
CSR_BT_SYNCS_PUT_PB_ADD_ENTRY_RES	✓	✓			✓				✓	✓			✓

Table 17: CSR_BT_SYNCS_PUT_PB_ADD_ENTRY Primitives

Description

The incoming of a phonebook object that is new is interpretative into the signal CSR_BT_SYNCS_PUT_PB_ADD_ENTRY. The application can then store the object. If the finalFlag is set the whole object is in the body otherwise the application has to remember which object to store and add the incoming body part from the CSR_BT_SYNCS_PUT_NEXT_IND until the finalFlag is set. The result of the store operation is given with the CSR_BT_SYNCS_PUT_PB_ENTRY_ADD_RES or CSR_BT_SYNCS_PUT_NEXT_RES. If the object is too large for one packet, the result can contain error codes corresponding to the reason for failure.

The appParameter is only used if the finalFlag is set in the indication. The appParameter must be packed as the IrOBEX specification specifies.

Parameters

Type	Signal identity, CSR_BT_SYNCS_PUT_PB_ADD_ENTRY_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (object) fits the whole object or that it's the last part.
lengthOfObject	The total length of the object to send.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
ucs2nameOffset	Offset for a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function "CsrUcs2ByteString2Utf8" can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyLength	The length of the body (object).
bodyOffset	Offset relative to payload for the object itself.
appParameterLength	The length of the appParameter (object).

*appParameter	The application response parameter, is a CsrUint8 pointer to the parameter.
payloadLength	Number of bytes in the payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.18 CSR_BT_SYNCS_GET_CAL_CHANGE_LOG

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	ucs2nameOffset	totalObjectSize	bodyLength	*body	payloadLength	*payload	smpOn
CSR_BT_SYNCS_GET_CAL_CHANGE_LOG_IND	✓	✓			✓				✓	✓	
CSR_BT_SYNCS_GET_CAL_CHANGE_LOG_RES	✓	✓	✓	✓		✓	✓	✓			✓

Table 18: CSR_BT_SYNCS_GET_CAL_CHANGE_LOG Primitives

Description

The requesting for the change log for the calendar objects is interpretative into the signal CSR_BT_SYNCS_GET_CAL_CHANGE_LOG_IND, the name parameter is indicating which log the client is requesting. The application is returning the change log in the body. If a successful response for a log that fits entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_CAL_CHANGE_LOG_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_CAL_CHANGE_LOG_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
ucs2nameOffset	Offset for a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
totalObjectSize	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 object itself. "body" is a CsrUint8 pointer to the object.

payloadLength	Number of bytes in the payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.19 CSR_BT_SYNCS_GET_CAL_CUR_CHANGE_LOG

Primitives \ Parameters	type	connectionId	responseCode	totalObjectSize	changeCounterLength	*changeCounter	smpOn
CSR_BT_SYNCS_GET_CAL_CUR_CHANGE_ LOG _IND	✓	✓					
CSR_BT_SYNCS_GET_CAL_CUR_CHANGE_LOG _RES	✓	✓	✓	✓	✓	✓	✓

Table 19: CSR_BT_SYNCS_GET_CAL_CUR_CHANGE_LOG Primitives

Description

The request for the current change counter for the calendar objects is interpretative into the signal CSR_BT_SYNCS_GET_CAL_CUR_CHANGE_LOG. The application is returning the current change counter in the body. In case the result is different from success, the parameter (changeCounter) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_CAL_CUR_CHANGE_LOG_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	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.
changeCounter Length	The length of the changeCounter (object).
changeCounter	The object itself. "changeCounter" is a CsrUint8 pointer to the object.
smpOn	Reserved for future use. Set to FALSE.

4.20 CSR_BT_SYNCS_GET_CAL_INFO_LOG

Parameters / Primitives	type	connectionId	finalFlag	responseCode	ltotalObjectSize	bodyLength	*body	smpOn
CSR_BT_SYNCS_GET_CAL_INFO_LOG_IND	✓	✓						
CSR_BT_SYNCS_GET_CAL_INFO_LOG_RES	✓	✓	✓	✓	✓	✓	✓	✓

Table 20: CSR_BT_SYNCS_GET_CAL_INFO_LOG Primitives

Description

The requesting for calendar information log is interpretative into the signal CSR_BT_SYNCS_GET_CAL_INFO_LOG. The application is returning the information log object in the body. If a successful response for a log that fits entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_CAL_INFO_LOG_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_CAL_INFO_LOG_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	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 object itself. "body" is a CsrUint8 pointer to the object.
smpOn	Reserved for future use. Set to FALSE.

4.21 CSR_BT_SYNCS_GET_CAL_ENTRY

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	ucs2nameOffset	totalObjectSize	bodyLength	*body	payloadLength	*payload	smpOn
CSR_BT_SYNCS_GET_CAL_ENTRY_IND	✓	✓			✓				✓	✓	
CSR_BT_SYNCS_GET_CAL_ENTRY_RES	✓	✓	✓	✓		✓	✓	✓			✓

Table 21: CSR_BT_SYNCS_GET_CAL_ENTRY Primitives

Description

The requesting for a specific Vcalendar entry is interpretative into the signal CSR_BT_SYNCS_GET_CAL_ENTRY. The application is returning the object requested with the name parameter in the body. If a successful response for an entry that fits entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND). Only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_CAL_ENTRY_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_CAL_ENTRY_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
ucs2nameOffset	Offset relative to payload of a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
totalObjectSize	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 object itself. "body" is a CsrUint8 pointer to the object.
payloadLength	Number of bytes in payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.22 CSR_BT_SYNCS_PUT_CAL_ENTRY

Parameters / Primitives	type	connectionId	finalFlag	responseCode	totalObjectSize	ucs2nameOffset	bodyOffset	bodyLength	appParameterLength	*appParameter	payloadLength	*payload	smpOn
CSR_BT_SYNCS_PUT_CAL_ENTRY_IND	✓	✓	✓		✓	✓	✓	✓			✓	✓	
CSR_BT_SYNCS_PUT_CAL_ENTRY_RES	✓	✓		✓					✓	✓			✓

Table 22: CSR_BT_SYNCS_PUT_CAL_ENTRY Primitives

Description

The incoming of a calendar object that is known is interpretative into the signal CSR_BT_SYNCS_PUT_CAL_ENTRY. The application can then store the object. If the finalFlag is set the whole object is in the body otherwise the application has to remember which object to store and add the incoming body part from the CSR_BT_SYNCS_PUT_NEXT_IND until the finalFlag is set. The result of the store operation is given with the CSR_BT_SYNCS_PUT_CAL_ENTRY_RES or CSR_BT_SYNCS_PUT_NEXT_RES. If the object was too large for one packet, the result can contain error codes corresponding to the reason for failure.

The parameter name indicates the object that has been modified. If the length is CSR_BT_NO_BODY_HEADER and finalFlag is set, then the body is empty and that indicates that the object is deleted from the client.
The appParameter is only used if the finalFlag is set in the indication. The appParameter must be packed as the IrOBEX specification specifies.

Parameters

type	Signal identity, CSR_BT_SYNCS_PUT_CAL_ENTRY_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (object) fits the whole object or that it is the last part.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	The total length of the object to send.
ucs2nameOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyLength	The length of the body (object).
bodyOffset	Offset for the object itself.

appParameterLength	The length of the appParameter (object).
*appParameter	The application response parameter, is a CsrUint8 pointer to the parameter.
payloadLength	Number of bytes in the payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE.

4.23 CSR_BT_SYNCS_PUT_CAL_ADD_ENTRY

Parameters / Primitives	type	connectionId	finalFlag	responseCode	totalObjectSize	ucs2nameOffset	bodyLength	bodyOffset	*appParameter	appParameterLength	payloadLength	*payload	smpOn
CSR_BT_SYNCS_PUT_CAL_ADD_ENTRY_IND	✓	✓	✓		✓	✓	✓	✓			✓	✓	
CSR_BT_SYNCS_PUT_CAL_ADD_ENTRY_RES	✓	✓		✓					✓	✓			✓

Table 23: CSR_BT_SYNCS_PUT_CAL_ADD_ENTRY Primitives

Description

The incoming of a calendar object that is new is interpretative into the signal CSR_BT_SYNCS_PUT_CAL_ADD_ENTRY. The application can then store the object. If the finalFlag is set the whole object is in the body otherwise the application has to remember which object to store and add the incoming body part from the CSR_BT_SYNCS_PUT_NEXT_IND until the finalFlag is set. The result of the store operation is given with the CSR_BT_SYNCS_PUT_CAL_ENTRY_ADD_RES or CSR_BT_SYNCS_PUT_NEXT_RES. If the object is too large for one packet, the result can contain error codes corresponding to the reason for failure.

The appParameter is only used if the finalFlag is set in the indication. The appParameter must be packed as the IrOBEX specification specifies.

Parameters

type	Signal identity, CSR_BT_SYNCS_PUT_CAL_ADD_ENTRY_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (object) fits the whole object or that it is the last part.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
totalObjectSize	The total length of the object to send.
ucs2nameOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
bodyLength	The length of the body (object).
bodyOffset	Payload relative offset of the object itself.
*appParameter	The application response parameter, is a CsrUint8 pointer to the parameter.

appParameterLength	The length of the appParameter (object).
payloadLength	Number of bytes in the payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
smpOn	Reserved for future use. Set to FALSE

4.24 CSR_BT_SYNCS_GET_CAL_ALL

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	bodyLength	*body	smpOn
CSR_BT_SYNCS_GET_CAL_ALL_IND	✓	✓					
CSR_BT_SYNCS_GET_CAL_ALL_RES	✓	✓	✓	✓	✓	✓	✓

Table 24: CSR_BT_SYNCS_GET_CAL_ALL Primitives

Description

The requesting for all the calendar object entries is interpretative into the signal CSR_BT_SYNCS_GET_CAL_ALL_IND. The application is returning the all the calendar object in the body. If a successful response for all the entries that fits entirely in one response packet is achieved the finalFlag is set. If the response is too large to fit into one packet then multiple requests are used (CSR_BT_SYNCS_GET_NEXT_IND), only the last response has the finalFlag set and the application has to remember the first request (CSR_BT_SYNCS_GET_CAL_ALL_IND) until the last response. In case the result is different from success, the parameter (body) is invalid and not used.

Parameters

type	Signal identity, CSR_BT_SYNCS_GET_CAL_ALL_IND/RES.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
finalFlag	Indicate that the body (object) fits the whole object or that it is the last part.
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. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
bodyLength	The length of the body (object).
*body	The object itself. "body" is a CsrUint8 pointer to the object.
smpOn	Reserved for future use. Set to FALSE.

4.25 CSR_BT_SYNCS_ABORT

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

Table 25: CSR_BT_SYNCS_ABORT Primitives

Description

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

Parameters

type	Signal identity, CSR_BT_SYNCS_ABORT_IND.
connectionId	Is the connection Id for this session, the SYNC client 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 description field.
payloadLength	Length of the payload in bytes.
*payload	OBEX data payload. Offsets are relative to this pointer.

4.26 CSR_BT_SYNCS_DISCONNECT

Primitives \ Parameters	type	connectionId	reasonCode	reasonSupplier
CSR_BT_SYNCS_DISCONNECT_IND	✓	✓	✓	✓

Table 26: CSR_BT_SYNCS_DISCONNECT Primitives

Description

This signal is indicating that the OBEX synchronization session is finished, and is ready for a new one.

Parameters

type	Signal identity, CSR_BT_SYNCS_DISCONNECT_IND.
connectionId	Is the connection Id for this session, the SYNC client will use this Id in the request.
reasonCode	The reason code of the operation. Possible values depends on the value of reasonSupplier. If eg. 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

4.27 CSR_BT_SYNCS_SECURITY_IN

Primitives \ Parameters	type	appHandle	secLevel	responseCode
CSR_BT_SYNCS_SECURITY_IN_REQ	✓	✓	✓	
CSR_BT_SYNCS_SECURITY_IN_CFM	✓			✓

Table 27: CSR_BT_SYNCS_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_SYNCS_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)

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.

The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.

5 Document References

Document	Reference
OBJECT PUSH PROFILE Version 1.1 Section K:11 22 February 2001	[OPP]
SYNCHRONIZATION PROFILE Version 1.1 Section K:13 22 February 2001	[SYNC]
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®	A set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
OPS	OBEX Push Server
OPC	OBEX Push Client
SYNCS	OBEX Sync Server
SIG	Special Interest Group
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.