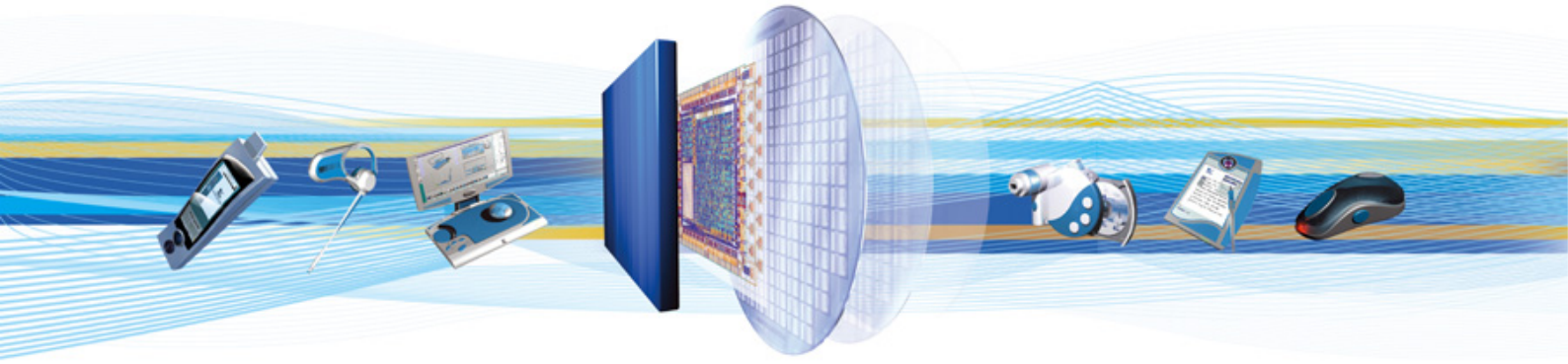
CSR Synergy Bluetooth 18.2.0

OBEX File Transfer Client

API Description

November 2011



Cambridge Silicon Radio Limited

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

Registered in England and Wales 3665875

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

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction and Scope

This document describes the message interface provided by the OBEX File Transfer Client (FTC). The FTC conforms to the client side of the File Transfer Profile, ref. [FTC].

1.2 Assumptions

The following assumptions and preconditions are made in the following:

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

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 browse an object store (file system) of another Bluetooth® device e.g. a mobile phone. Browsing involves viewing objects (files and folders) and navigating the folder hierarchy of another Bluetooth® device. For example, a PC browsing the file system of a mobile device
- A second usage is transfer objects (files and folders) between two Bluetooth® devices. For example, copying files from a PC to a mobile device
- A third usage is a Bluetooth® device to manipulate objects (files and folders) on another Bluetooth® device. This includes deleting objects, and creating new folders

The FTC provides the following services to the application:

- inquiry of devices
- screening of those
- connection handling
- OBEX protocol handling

The application is responsible for handling the requests and confirms from the FTC with correct data (object) as described in the IrOBEX specification. The FTC is not checking if the data is packed correctly with white spaces in the right places, for details see ref. [FTP] and [OBEX].

2.2 Reference Model

The FTC interfaces to the Connection Manager (CM).

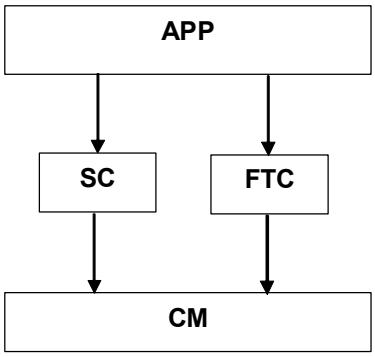


Figure 1: Reference model

2.3 Sequence Overview

If, in IDLE state, a connect request is received from the application, the FTC starts to connect to the specified device and the CONNECT state is entered, then the application receives a confirmation on the connect, the application can issue a request (get, put or setpath) to start the transfer for the object or folder browsing. The application can perform multiple gets/puts (one at the time) before disconnecting the connection. When the application disconnects the service, the FTC re-enters IDLE state.

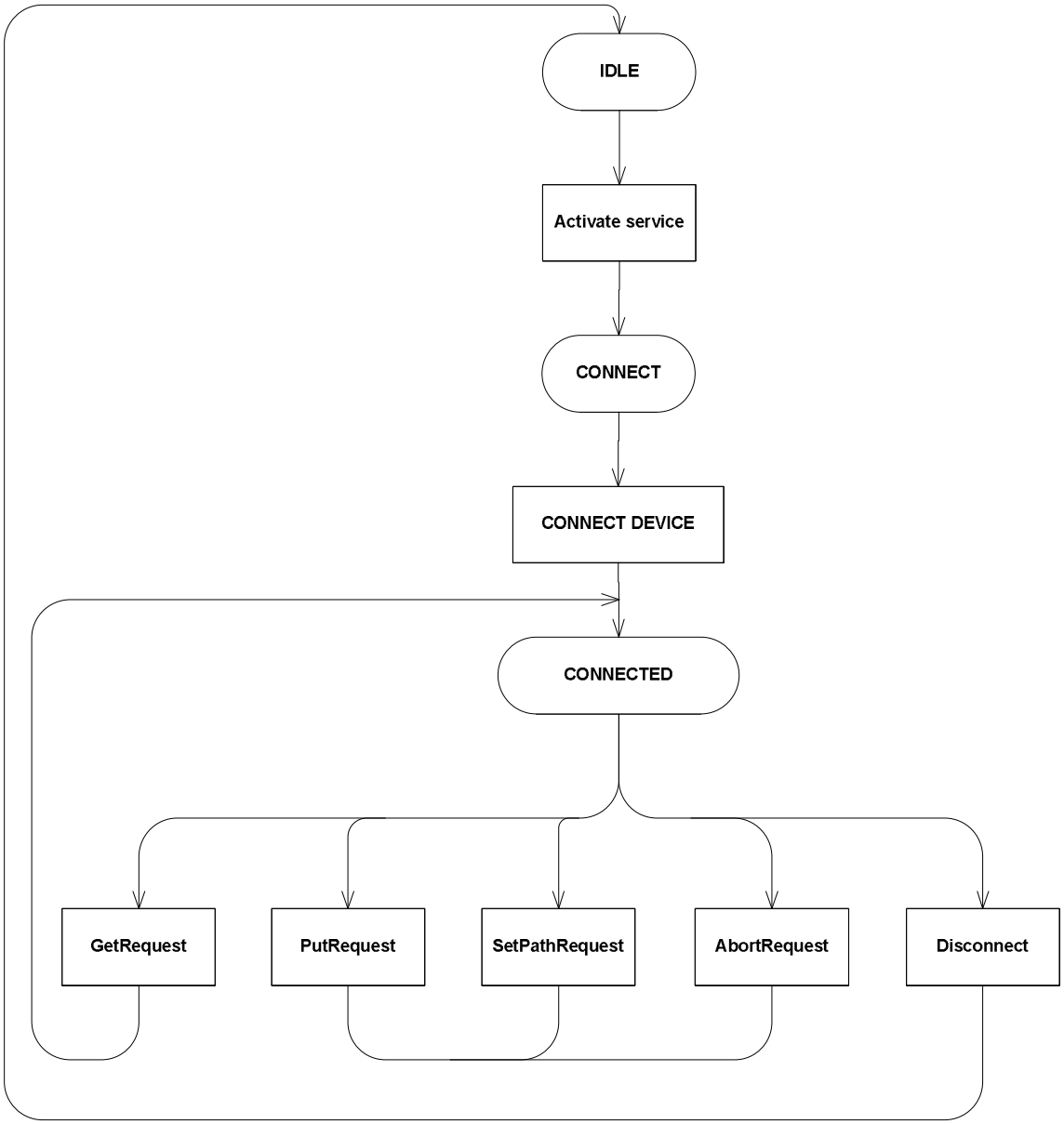


Figure 2: FTC state diagram

3 Interface Description

3.1 Connect

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

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

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

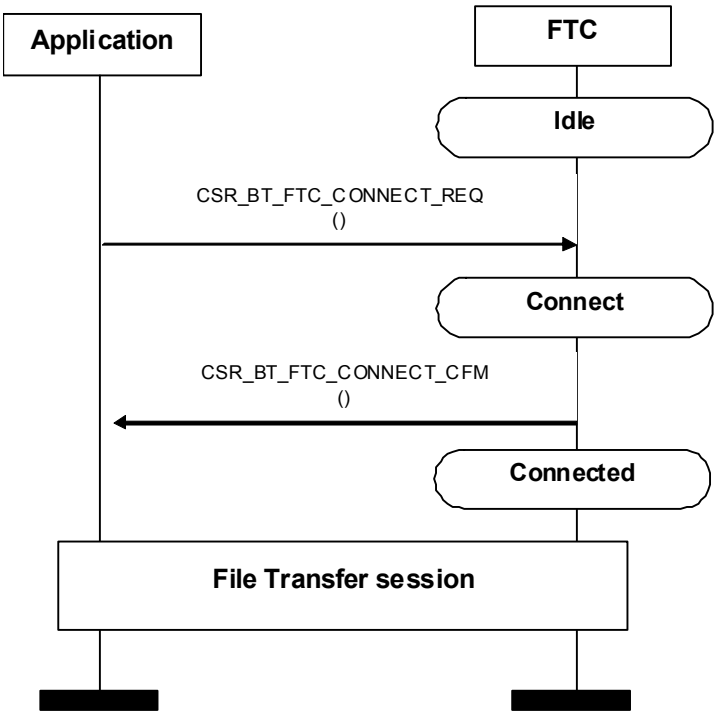


Figure 3: Connection handling

3.2 Cancel Connect

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

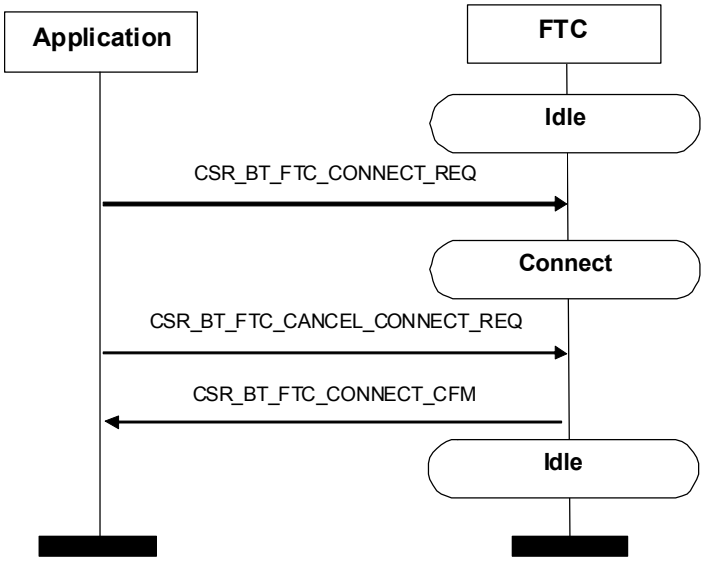


Figure 4: Cancel Connect I

Please note that if the application requests a CSR_BT_FTC_CANCEL_CONNECT_REQ while the FTC is sending a CSR_BT_FTC_CONNECT_CFM with the response code CSR_BT_OBEX_SUCCESS_RESPONSE_CODE to the application, then will FTC consider the CSR_BT_FTC_CANCEL_CONNECT_REQ as a CSR_BT_FTC_DISCONNECT_REQ and the application will receive a CSR_BT_FTC_DISCONNECT_IND.

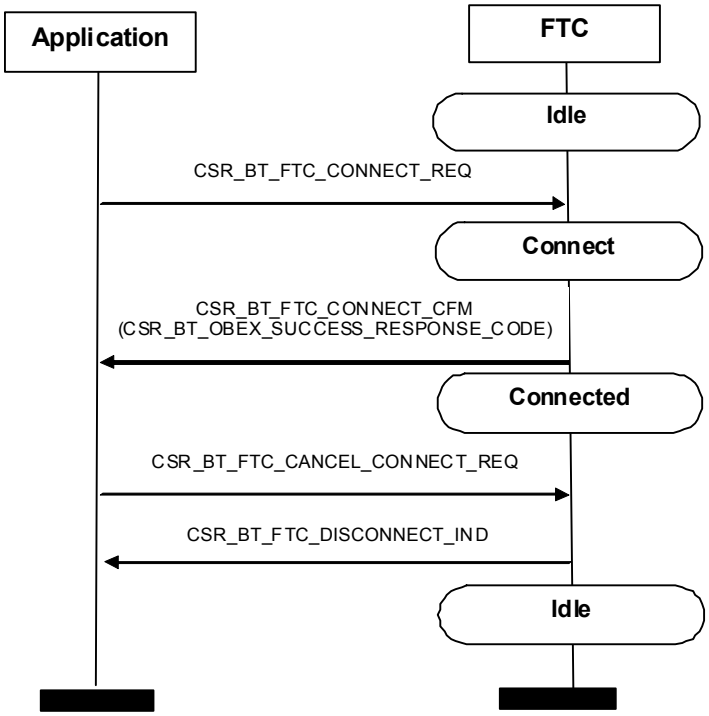


Figure 5: Cancel Connect II

3.3 Folder Browsing

Browsing an object store involves displaying folder contents and changing the 'current folder' forwards and backwards. The CSR_BT_FTC_SET_FOLDER_REQ is used for changing the current folder forward in the folder hierarchy. There are two ways to go back in the folder hierarchy – the first way is the CSR_BT_FTC_SET_ROOT_FOLDER_REQ changing the current folder to the root folder (the root folder is the start up folder after a connect). The second way is the CSR_BT_FTC_SET_FOLDER_BACK_REQ changing the current folder back to the parent folder of the current one. To display a folder hierarchy starting with the root folder, the client must read the folder content using CSR_BT_FTC_GET_LIST_FOLDER_REQ and the application responses with the folder list in the body of the CSR_BT_FTC_GET_LIST_FOLDER_CFM message. The fragmentation and use of finalFlag is described in sections 4.4 and 4.5.

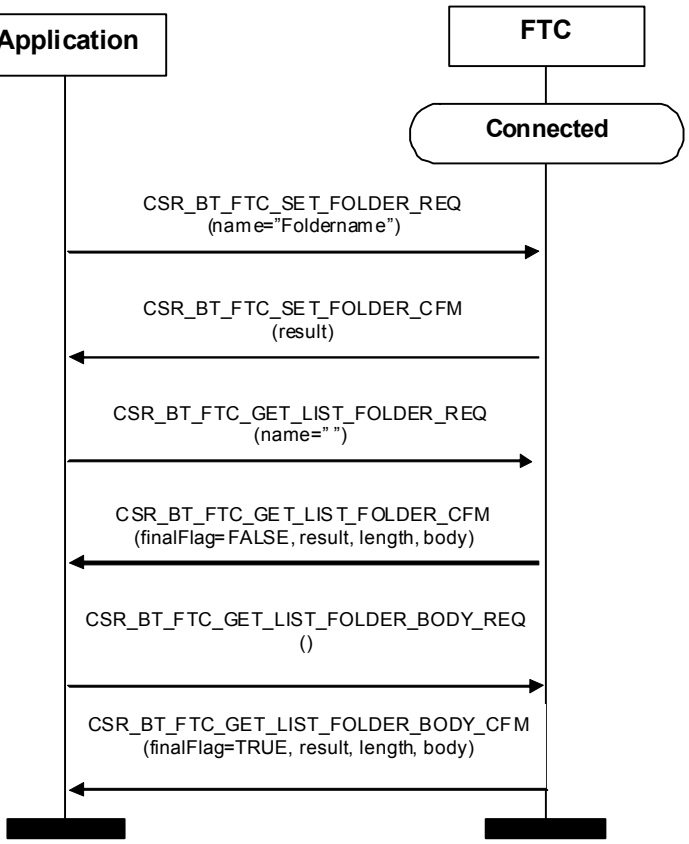


Figure 6: Folder browsing handling

3.4 Object Transfer

Objects are transmitted to the server by issuing a CSR_BT_FTC_PUT_OBJ_HEADER_REQ followed by a CSR_BT_FTC_PUT_OBJ_BODY_REQ. The server side responds with the result of the operation in a CSR_BT_FTC_PUT_OBJ_BODY_CFM signal. In case the application wants to fragment the body due to memory considerations it can do so by sending a CSR_BT_FTC_PUT_OBJ_BODY_REQ with finalFlag set to FALSE. On the confirm it can continue to send the next fragment. It can continue until it sets the finalFlag to TRUE.

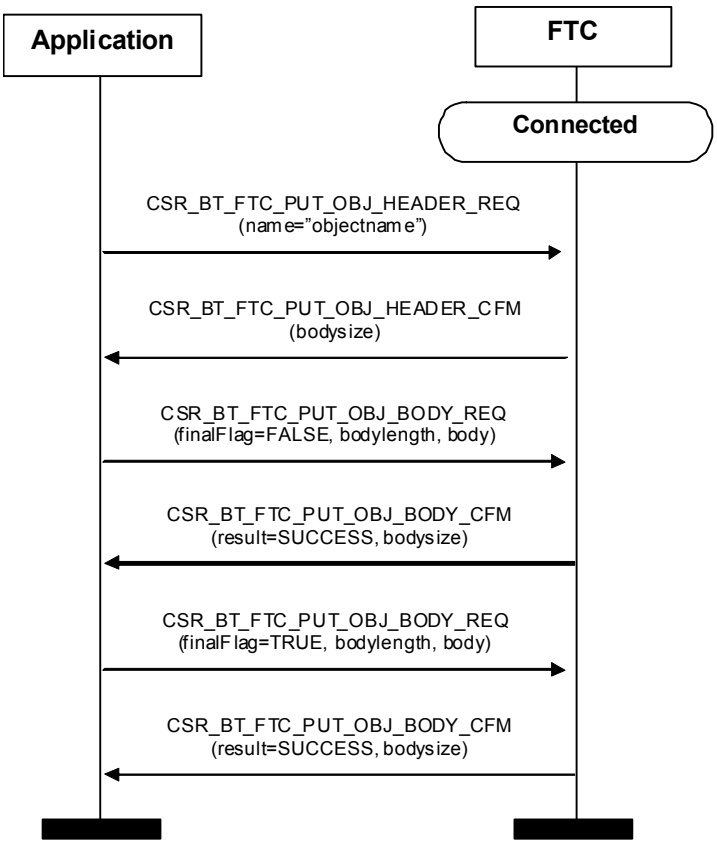


Figure 7: Put object

Objects can be pulled from the server by issuing a CSR_BT_FTC_GET_OBJ_REQ. The server responds with the result of the operation in a CSR_BT_FTC_GET_OBJ_CFM signal. If the server responds with multiple fragments, the first will be received by the application as a CSR_BT_FTC_GET_OBJ_CFM and the application has to send a CSR_BT_FTC_GET_OBJ_BODY_REQ to get the next fragment. This is the pattern until the finalFlag parameter in the confirm is set, this indicates that it is the last fragment.

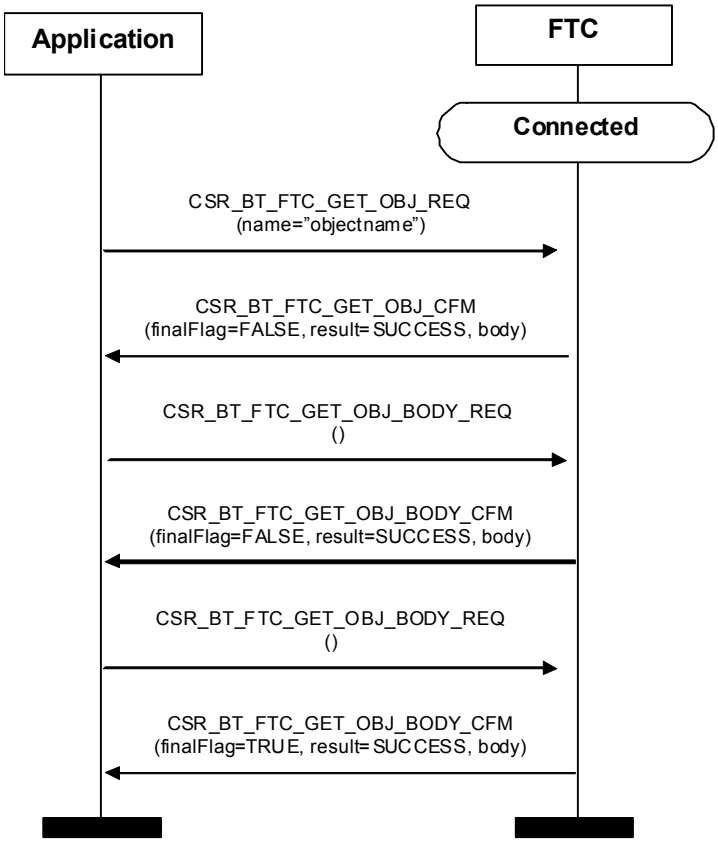


Figure 8: Get object

3.5 Objects Manipulating

Manipulating objects includes deleting objects and creating new folders. Creating a folder is done with the CSR_BT_FTC_SET_ADD_FOLDER_REQ. The new folder will be created as a sub folder for the current folder. If the creation of the folder is a CSR_BT_SUCCESS the current folder will be the new folder.

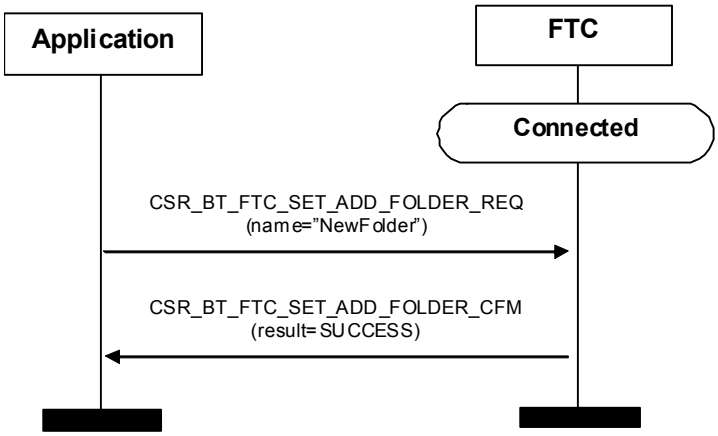


Figure 9: Create folder

Deleting an object is done with the CSR_BT_FTC_DEL_OBJ_REQ message - the object can either be a file or a folder. Some FTP servers may not allow to delete non-empty folders, the confirm will then have the result code CSR_BT_OBEX_PRECONDITION_FAILED_RESPONSE_CODE indicating that the folder is not empty. In this case the application will need to delete the contents before deleting the folder.

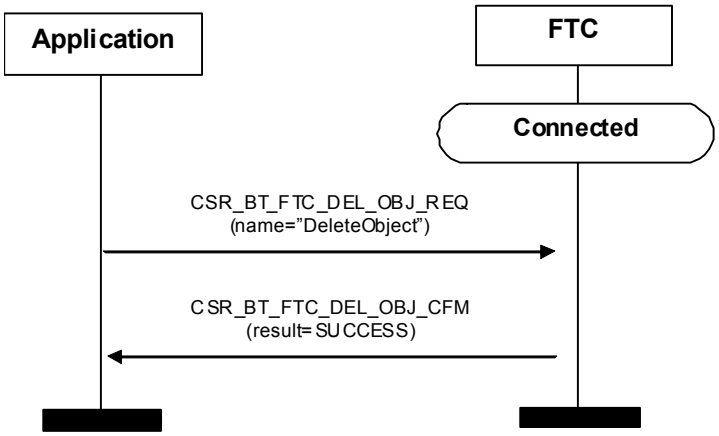


Figure 10: Delete object

3.6 Abort Operation

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

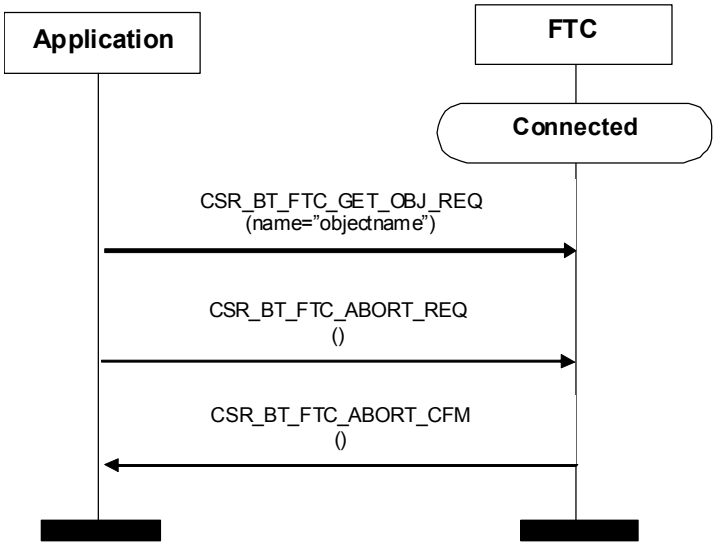


Figure 11: Abort operation

3.7 Obex Authentication

The application can authenticate the FTP server by setting the authorize and password parameters in the CSR_BT_FTC_CONNECT_REQ. The FTC will then authenticate the FTP server under the obex connection establishment. A FTP server can authenticate the FTC on every operation individually. For each application request it sends it can receive a CSR_BT_FTC_AUTHENTICATE_IND instead of the corresponding confirm message. If the application receives a CSR_BT_FTC_AUTHENTICATE_IND it must response with a CSR_BT_FTC_AUTHENTICATE_RES signal using the password or pin number that the FTP server requires. An example of the authenticate sequence is illustrated below.

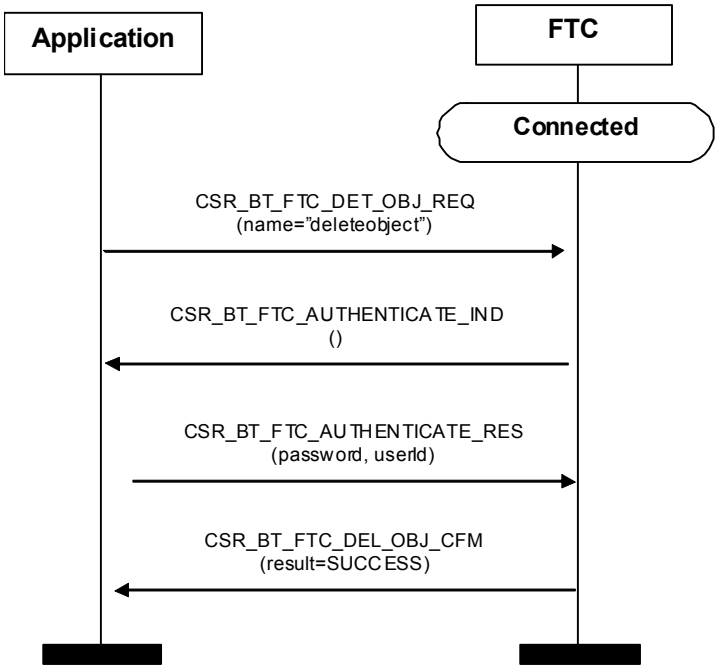


Figure 12: Authenticate delete object

3.8 Disconnect

Sending a CSR_BT_FTC_DISCONNECT_REQ to the FTC disconnects the current connection (if any). The disconnect may take some time and is confirmed with a CSR_BT_FTC_DISCONNECT_IND signal.

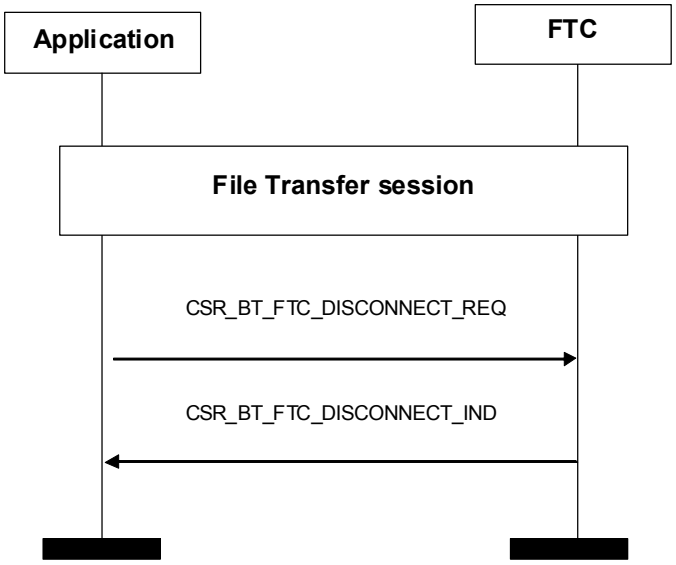


Figure 13: Normal disconnect

In case the peer side prematurely disconnects, the FTC sends a CSR_BT_FTC_DISCONNECT_IND to the application and enters IDLE state.

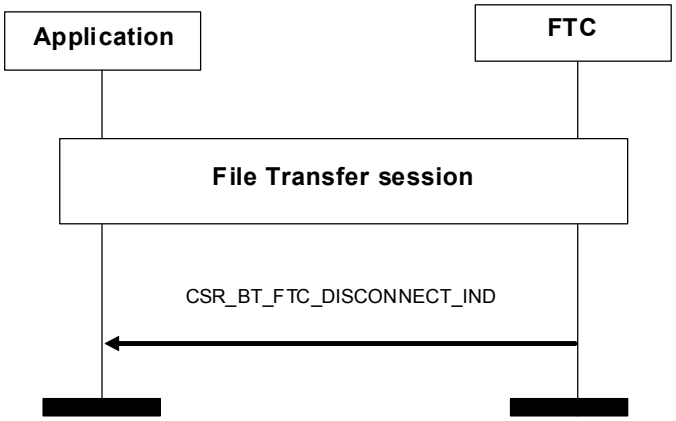


Figure 14: Abnormal disconnect

3.9 Payload Encapsulated Data

3.9.1 Using Offsets

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

```
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) it 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.

3.9.2 Payload Memory

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

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

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

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

4 OBEX File Transfer Client Primitives

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

4.1 List of All Primitives

Primitives:	Reference:
CSR_BT_FTC_CONNECT_REQ	See section 4.2
CSR_BT_FTC_CONNECT_CFM	See section 4.2
CSR_BT_FTC_AUTHENTICATE_IND	See section 4.3
CSR_BT_FTC_AUTENTICATE_RES	See section 4.3
CSR_BT_FTC_GET_LIST_FOLDER_REQ	See section 4.4
CSR_BT_FTC_GET_LIST_FOLDER_CFM	See section 4.4
CSR_BT_FTC_GET_LIST_FOLDER_BODY_REQ	See section 4.5
CSR_BT_FTC_GET_LIST_FOLDER_BODY_CFM	See section 4.5
CSR_BT_FTC_GET_OBJ_REQ	See section 4.6
CSR_BT_FTC_GET_OBJ_CFM	See section 4.6
CSR_BT_FTC_GET_OBJ_BODY_REQ	See section 0
CSR_BT_FTC_GET_OBJ_BODY_CFM	See section 0
CSR_BT_FTC_PUT_OBJ_HEADER_REQ	See section 4.8
CSR_BT_FTC_PUT_OBJ_HEADER_CFM	See section 4.8
CSR_BT_FTC_PUT_OBJ_BODY_REQ	See section 4.8
CSR_BT_FTC_PUT_OBJ_BODY_CFM	See section 4.8
CSR_BT_FTC_DEL_OBJ_REQ	See section 4.9
CSR_BT_FTC_DEL_OBJ_CFM	See section 4.9
CSR_BT_FTC_SET_FOLDER_REQ	See section 4.10
CSR_BT_FTC_SET_FOLDER_CFM	See section 4.10
CSR_BT_FTC_SET_BACK_FOLDER_REQ	See section 4.11
CSR_BT_FTC_SET_BACK_FOLDER_CFM	See section 4.11
CSR_BT_FTC_SET_ROOT_FOLDER_REQ	See section 4.12
CSR_BT_FTC_SET_ROOT_FOLDER_CFM	See section 4.12
CSR_BT_FTC_SET_ADD_FOLDER_REQ	See section 4.13
CSR_BT_FTC_SET_ADD_FOLDER_CFM	See section 4.13
CSR_BT_FTC_ABORT_REQ	See section 4.14
CSR_BT_FTC_ABORT_CFM	See section 4.14
CSR_BT_FTC_DISCONNECT_REQ	See section 4.15
CSR_BT_FTC_DISCONNECT_IND	See section 4.15
CSR_BT_FTC_CANCEL_CONNECT_REQ	See section 4.16
CSR_BT_FTC_SECURITY_OUT_REQ	See section 4.17
CSR_BT_FTC_SECURITY_OUT_CFM	See section 4.17
CSR_BT_FTC_COPYING_OBJ_REQ	See section 4.18
CSR_BT_FTC_COPYING_OBJ_CFM	See section 4.18
CSR_BT_FTC_MOVING_OBJ_REQ	See section 4.19
CSR_BT_FTC_MOVING_OBJ_CFM	See section 4.19
CSR_BT_FTC_SET_OBJ_PERMISSIONS_REQ	See section 4.20

CSR_BT_FTC_SET_OBJ_PERMISSIONS_CFM	See section 4.20

Table 1: List of all primitives

4.2 CSR_BT_FTC_CONNECT

Primitives \ Parameters	type	appHandle	maxPacketSize	destination	obexPeerMaxPacketSize	resultCode	resultSupplier	authorize	realmLength	*realm	*password	passwordLength	*userId	length	count	btConnId	windowSize	srmEnable
CSR_BT_FTC_CONNECT_REQ	✓	✓	✓	✓				✓	✓	✓	✓	✓	✓	✓	✓		✓	✓
CSR_BT_FTC_CONNECT_CFM	✓				✓	✓	✓									✓		

Table 2: CSR_BT_FTC_CONNECT Primitives

Description

To start an OBEX file transfer session against the FTP server, the application sends a CSR_BT_FTC_CONNECT_REQ with the wanted max packet size allowed to send from the server. The server responds with a CSR_BT_FTC_CONNECT_CFM. In case the result in the confirmation is CSR_BT_OBEX_SUCCESS_RESULT_CODE the connect is established. Any other value indicates a failure in the connection.

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

The function:

```
CsrBtFtcConnectReqSend (CsrSchedQid          theAppHandle,
                        CsrUint16           theMaxPacketSize,
                        CsrBtDeviceAddr     theDestination,
                        CsrBool             authorize,
                        CsrUint16           realmLength,
                        data_ptr_t          *realm,
                        CsrUint16           passwordLength,
                        CsrUint8            *password,
                        CsrCharString       *userId,
                        CsrUint32           length,
                        CsrUint32           count,
                        CsrUint16           windowSize,
                        CsrBool             srmEnable );
```

defined in csr_bt_ftc_lib.h, builds and sends the CSR_BT_FTC_CONNECT_REQ primitive to the FTC profile.

Parameters

type	Signal identity, CSR_BT_FTC_CONNECT_REQ/CFM.
appHandle	The identity of the calling process. It is possible to initiate the procedure by any higher layer process as the response is returned to appHandle.
maxPacketSize	The maximum obex packet size allowed sending to the client application.

destination	The Bluetooth® address of the device to connect to.
resultCode	The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h files or csr_bt_obex.h are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h
obexPeerMaxPacketSize	Indicates the maximum size OBEX packet that is allowed to send to the server
authorize	Is to control the OBEX authentication on connection to a FTP server. If TRUE the FTC will initiate the authentication against the server.
realmLength	Number of bytes in realm of type CsrUint16.
*realm	A displayable string indicating for the user which userid and/or password to use. The first byte of the string is the character set of the string. The table below shows the different values for character set. Note that this pointer must be pfree by the application, and that this pointer can be NULL because the realm field is optional to set by the peer device.

Char set Code	Meaning
0	ASCII
1	ISO-8859-1
2	ISO-8859-2
3	ISO-8859-3
4	ISO-8859-4
5	ISO-8859-5
6	ISO-8859-6
7	ISO-8859-7
8	ISO-8859-8
9	ISO-8859-9
0xFF = 255	UNICODE

passwordLength	The length of the response password.
*password	Containing the challenge password of the OBEX authentication. This is a pointer which shall be allocated by the application.
*userId	Zero terminated string (ASCII) containing the userId for the authentication. This is a pointer which shall be allocated by the application.
length	Length is use to express the approximate total length of the bodies of all the objects in the transaction. If set to 0 this header will not be include.
count	Count is use to indicate the number of objects that will be sent during this connection.

	If set to 0 this header will not be include
btConnId	Identifier which shall be used when using AMPM, for more information please refer to [AMPM].
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	TRUE enables local support for Single Response Mode (SRM). If SRM is enabled FTC allows that PUT and GET commands, multiple OBEX request packets (PUT) or OBEX response packet (GET), can be send immediately, without waiting for the remote device. Please note, SRM can only be enabled if both sides support it. For more information about SRM, please refer to [GOEP2.0].

4.3 CSR_BT_FTC_AUTHENTICATE

Primitives \ Parameters	type	options	realmLength	* realm	deviceAddr	*password	passwordLength	*userId
CSR_BT_FTC_AUTHENTICATE_IND	✓	✓	✓	✓	✓			
CSR_BT_FTC_AUTHENTICATE_RES	✓					✓	✓	✓

Table 3: CSR_BT_FTC_AUTHENTICATE Primitives

Description

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

Parameters

type — Signal identity, CSR_BT_FTC_AUTHENTICATE_IND/RES.

options — Challenge information of type CsrUint8.

Bit 0 controls the responding of a valid user Id.

If bit 0 is set it means that the application must response with a user Id in a CSR_BT_FTC_AUTHENTICATE_RES message. If bit 0 is not set the application can just set the userId to NULL.

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

Bit 2 - 7 is reserved.

realmLength — Number of bytes in realm of type CsrUint16.

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

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

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

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

deviceAddr	The Bluetooth address of the device that has initiated the OBEX authentication procedure
*password	Contains 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.

4.4 CSR_BT_FTC_GET_LIST_FOLDER

Primitives \ Parameters	type	*ucs2name	responseCode	lengthOfObject	finalFlag	bodyLength	bodyOffset	payloadLength	*payload	srmpOn
CSR_BT_FTC_GET_LIST_FOLDER_REQ	✓	✓								✓
CSR_BT_FTC_GET_LIST_FOLDER_CFM	✓		✓	✓	✓	✓	✓	✓	✓	

Table 4: CSR_BT_FTC_GET_LIST_FOLDER Primitives

Description

This signal is used for pulling the content of the current folder or a folder specified by the name parameter. The FTP server has the right to refuse to disclose the content of the folder by replying with a CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE result code or send a CSR_BT_FTC_AUTHENTICATE_IND to authenticate the client. If allowed, the content of the folder must be sent in the Folder Listing format specified in ref. [OBEX].

When a successful response for an object that fits entirely in one response packet is achieved the finalFlag is set, the whole packet is in the CSR_BT_FTC_GET_LIST_FOLDER_CFM signal and the packet is in the body parameter.

If the response is too large to fit into one packet the client needs to request for the rest of the packet with the CSR_BT_FTC_GET_LIST_FOLDER_BODY_REQ until the confirm signal (CSR_BT_FTC_GET_LIST_FOLDER_BODY_CFM) has the finalFlag set. In case the result is different from success (when the result is not CSR_BT_OBEX_SUCCESS_RESPONSE_CODE) the other parameters are invalid and not used.

Parameters

type	Signal identity, CSR_BT_FTC_GET_LIST_FOLDER_REQ/CFM.
*ucs2name	A null terminated 16 bit Unicode text string (UCS2) containing the (folder) name of the object. The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.
lengthOfObject	The total length of the object to receive.
finalFlag	Indicates that the body (object) fits the whole object or that it is the last part.
bodyLength	The length of the body (object).
bodyOffset	Offset relative to payload of the object itself.
payloadLength	Number of bytes in payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabledd, see section 4.2, the FTP Client can instruct the FTP Server to wait for the next request packet during a GET Operation by setting the

srmpOn parameter TRUE.

If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait; however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.5 CSR_BT_FTC_GET_LIST_FOLDER_BODY

Parameters / Primitives	type	responseCode	finalFlag	bodyLength	bodyOffset	payloadLength	*payload	srmpOn
CSR_BT_FTC_GET_LIST_FOLDER_BODY_REQ	✓							✓
CSR_BT_FTC_GET_LIST_FOLDER_BODY_CFM	✓	✓	✓	✓	✓	✓	✓	

Table 5: CSR_BT_FTC_GET_LIST_FOLDER_BODY Primitives

Description

This signal is used if the pulled content folder is too large to fit into one Obex packet. The first packet is sent in CSR_BT_FTC_GET_LIST_FOLDER_CFM, the next part is sent in CSR_BT_FTC_GET_LIST_FOLDER_BODY_CFM after sending the CSR_BT_FTC_GET_LIST_FOLDER_BODY_REQ signal. The last response has to set the parameter finalFlag.

Parameters

type	Signal identity, CSR_BT_FTC_GET_LIST_FOLDER_BODY_REQ/CFM.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.
finalFlag	Indicates that the body (object) fits the whole object or that it is the last part.
bodyLength	The length of the body (object).
bodyOffset	Payload relative offset for the object itself.
payloadLength	Number of bytes in payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabledd, see section 4.2, the FTP Client can instruct the FTP Server to wait for the next request packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait; however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.6 CSR_BT_FTC_GET_OBJ

Primitives \ Parameters	type	*ucs2name	responseCode	lengthOfObject	finalFlag	bodyLength	bodyOffset	payloadLength	*payload	srmpOn
CSR_BT_FTC_GET_OBJ_REQ	✓	✓								✓
CSR_BT_FTC_GET_OBJ_CFM	✓		✓	✓	✓	✓	✓	✓	✓	

Table 6: CSR_BT_FTC_GET_OBJ Primitives

Description

To retrieve an object from the FTP server the apps sends the CSR_BT_FTC_GET_OBJ_REQ to the FTC and the name parameter specifies which object the FTP client wants. The FTP server responses with a CSR_BT_FTC_GET_OBJ_CFM. 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 too large to fit into one packet the client needs to request for the rest of the object with the CSR_BT_FTC_GET_OBJ_BODY_REQ until the confirm signal (CSR_BT_FTC_GET_OBJ_BODY_CFM) has the finalFlag set. In case the result is different from success (when the result is different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE), the other parameters are invalid and not used.

Parameters

type	Signal identity, CSR_BT_FTC_GET_OBJ_REQ/CFM.
*ucs2name	A null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.
lengthOfObject	The total length of the object to receive.
finalFlag	Indicates that the body (object) fits the whole object or that it is the last part.
bodyLength	The length of the body (object).
bodyOffset	Offset of object relative to payload.
payloadLength	Number of bytes in payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabledd, see section 4.2, the FTP Client can instruct the FTP Server to wait for the next request packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait; however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.7 CSR_BT_FTC_GET_OBJ_BODY

Primitives \ Parameters	type	responseCode	finalFlag	bodyLength	bodyOffset	payloadLength	*payload	srmpOn
CSR_BT_FTC_GET_OBJ_BODY_REQ	✓							✓
CSR_BT_FTC_GET_OBJ_BODY_CFM	✓	✓	✓	✓	✓	✓	✓	

Table 7: CSR_BT_FTC_GET_OBJ_BODY Primitives

Description

This signal is used if the pulled object is too large to fit into one Obex packet. The first packet is sent in CSR_BT_FTC_GET_OBJ_CFM, the next part is sent in CSR_BT_FTC_GET_OBJ_BODY_CFM after sending the CSR_BT_FTC_GET_OBJ_BODY_REQ signal. The last response has to set the parameter finalFlag.

Parameters

type	Signal identity, CSR_BT_FTC_GET_OBJ_BODY_REQ/CFM.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.
finalFlag	Indicates that the body (object) fits the whole object or that it is the last part.
bodyLength	The length of the body (object).
bodyOffset	Offset relative to payload of the object body.
payloadLength	Number of bytes in payload.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabledd, see section 4.2, the FTP Client can instruct the FTP Server to wait for the next request packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait; however, once the srmpOn parameter is FALSE in a GET request, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.8 CSR_BT_FTC_PUT_OBJ_HEADER & CSR_BT_FTC_PUT_OBJ_BODY

Parameters / Primitives	type	*ucs2name	lengthOfObject	bodySize	finalFlag	bodyLength	*body	responseCode
CSR_BT_FTC_PUT_OBJ_HEADER_REQ	✓	✓	✓					
CSR_BT_FTC_PUT_OBJ_HEADER_CFM	✓			✓				✓
CSR_BT_FTC_PUT_OBJ_BODY_REQ	✓				✓	✓	✓	
CSR_BT_FTC_PUT_OBJ_BODY_CFM	✓			✓				✓

Table 8: CSR_BT_FTC_PUT_OBJ_HEADER & CSR_BT_FTC_PUT_OBJ_BODY Primitives

Description

To push an object to the file transfer server, send the CSR_BT_FTC_PUT_OBJ_HEADER_REQ to the FTC. The FTC then sends a confirm with the maximum size of the object to send in the next signal (CSR_BT_FTC_PUT_OBJ_BODY_REQ). If the body needs to be sent as multiple fragments, the apps can send a new CSR_BT_FTC_PUT_OBJ_BODY_REQ after receiving a CSR_BT_FTC_PUT_OBJ_BODY_CFM message. The finalFlag indicates the last part of the body (object). In case the result in the confirm is different from success, the other parameters are invalid and not used and the apps can stop sending more of this object.

Parameters

type	Signal identities, CSR_BT_FTC_PUT_OBJ_HEADER_REQ/CFM and CSR_BT_FTC_PUT_OBJ_BODY/REQ/CFM.
*ucs2name	A null terminated 16 bit Unicode text string (UCS2) containing the (file) name of the object. The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.
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.
bodySize	The maximum length of the body (object) to send in the next packet (CSR_BT_FTC_PUT_OBJ_BODY_REQ).
finalFlag	Indicates that the body (object) fits the whole object or that it is the last part.
bodyLength	The length of the body (object).
*body	The object itself. "body" is a CsrUint8 pointer to the object.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.

4.9 CSR_BT_FTC_DEL_OBJ

Primitives \ Parameters	type	*ucs2name
CSR_BT_FTC_DEL_OBJ_ REQ	✓	✓
CSR_BT_FTC_DEL_OBJ_ CFM	✓	

Table 9: CSR_BT_FTC_DEL_OBJ Primitives

Description

This signal CSR_BT_FTC_DEL_OBJ_REQ is used for deleting objects (files or folders) on the FTP server. The result of the delete operation is given to the apps with the confirm signal CSR_BT_FTC_DEL_OBJ_CFM. The result can contain error codes corresponding to the reason for failure or if the server does not permit this operation from the FTC. The apps can also receive a CSR_BT_FTC_AUTHENTICATE_IND before getting the confirm, this takes place if the other side (FTP server) wants to authenticate the client for this operation.

Parameters

type — Signal identity, CSR_BT_FTC_DEL_OBJ_REQ/CFM.

*ucs2name — A null terminated 16 bit Unicode text string (UCS2) containing the (file/directory) name of the object.

The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.

4.10 CSR_BT_FTC_SET_FOLDER

Primitives \ Parameters	type	*ucs2name	responseCode
CSR_BT_FTC_SET_FOLDER_ REQ	✓	✓	
CSR_BT_FTC_SET_FOLDER_CFM	✓		✓

Table 10: CSR_BT_FTC_SET_FOLDER Primitives

Description

This signal is used for changing the current folder on the server to a folder specified with the name parameter. This is used for navigating down in the directory hierarchy on the server. The result of the change folder operation is given in the confirm signal. The result can contain error codes corresponding to the reason for failure in case the folder does not exist or in case the server does not permit this operation. The apps can also receive a CSR_BT_FTC_AUTHENTICATE_IND before getting the confirm, this takes places if the other side (FTP server) wants to authenticate the client for this operation.

Parameters

type	Signal identity, CSR_BT_FTC_SET_FOLDER_REQ.
*ucs2name	A null terminated 16 bit Unicode text string (UCS2) containing the (directory) name of the object. The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.

4.11 CSR_BT_FTC_SET_BACK_FOLDER

Primitives \ Parameters	type	responseCode
CSR_BT_FTC_SET_BACK_FOLDER_REQ	✓	
CSR_BT_FTC_SET_BACK_FOLDER_CFM	✓	✓

Table 11: CSR_BT_FTC_SET_BACK_FOLDER Primitives

Description

This signal is used for setting the current folder back to the parent folder. The result of the operation is given in the confirm signal. If the current folder is the root folder the confirm will have the CSR_BT_OBEX_NOT_FOUND_RESPONSE_CODE result code.

Parameters

type	Signal identity, CSR_BT_FTC_SET_BACK_FOLDER_REQ/CFM.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.

4.12 CSR_BT_FTC_SET_ROOT_FOLDER

Primitives \ Parameters	type	responseCode
CSR_BT_FTC_SET_ROOT_FOLDER_REQ	✓	
CSR_BT_FTC_SET_ROOT_FOLDER_CFM	✓	✓

Table 12: CSR_BT_FTC_SET_ROOT_FOLDER Primitives

Description

This signal is used for setting the current folder back to the root folder. The result in the confirm message can contain error codes corresponding to the reason for the failure on the server, but normally this operation cannot fail.

Parameters

type	Signal identity, CSR_BT_FTC_SET_ROOT_FOLDER_REQ/CFM.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.

4.13 CSR_BT_FTC_SET_ADD_FOLDER

Primitives \ Parameters	type	*ucs2name	responseCode
CSR_BT_FTC_SET_ADD_FOLDER_REQ	✓	✓	
CSR_BT_FTC_SET_ADD_FOLDER_CFM	✓		✓

Table 13: CSR_BT_FTC_SET_ADD_FOLDER Primitives

Description

This signal is used for creating a new folder on the server. The new folder is specified with the name parameter and will be created as a sub folder for current folder. The result of the create folder operation is given in the confirm signal. The result can contain error codes corresponding to the reason for the failure on the server. The apps can also receive a CSR_BT_FTC_AUTHENTICATE_IND before getting the confirm, this takes place if the other side wants to authenticate the client for this operation.

Parameters

type	Signal identity, CSR_BT_FTC_SET_ADD_FOLDER_REQ/CFM.
*ucs2name	A null terminated 16 bit Unicode text string (UCS2) containing the (directory) name of the object. The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the confirm the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.

4.14 CSR_BT_FTC_ABORT

Primitives \ Parameters	type
CSR_BT_FTC_ABORT_REQ	✓
CSR_BT_FTC_ABORT_CFM	✓

Table 14: CSR_BT_FTC_ABORT Primitives

Description

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

Parameters

type Signal identity, CSR_BT_FTC_ABORT_REQ/CFM.

4.15 CSR_BT_FTC_DISCONNECT

Primitives	type	normalDisconnect	reasonCode	reasonSupplier
CSR_BT_FTC_DISCONNECT_REQ	✓	✓		
CSR_BT_FTC_DISCONNECT_IND	✓		✓	✓

Table 15: CSR_BT_FTC_DISCONNECT Primitives

Description

To disconnect a connection to a server (if any) send a CSR_BT_FTC_DISCONNECT_REQ to the FTC. When disconnected, the FTC will respond with a CSR_BT_FTC_DISCONNECT_IND. If the link is dropped in the middle of a session, the apps will receive a CSR_BT_FTC_DISCONNECT_IND indicating that the OBEX file transfer session is finished, and is ready for a new one.

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

Parameters

type	Signal identity, CSR_BT_FTC_DISCONNECT_REQ/IND.
normalDisconnect	FALSE defines an Abnormal disconnect sequence where the Bluetooth connection is released directly. TRUE defines a normal disconnect sequence where the OBEX connection is released before the Bluetooth connection.
reasonCode	The reason code of the operation. Possible values depend on the value of reasonSupplier. If e.g. the reasonSupplier == CSR_BT_SUPPLIER_CM then the possible reason codes can be found in csr_bt_cm_prim.h. 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

4.16 CSR_BT_FTC_CANCEL_CONNECT

Primitives \ Parameters	type
CSR_BT_FTC_CANCEL_CONNECT_REQ	✓

Table 16: CSR_BT_FTC_CANCEL_CONNECT Primitives

Description

The CSR_BT_FTC_CANCEL_CONNECT_REQ can be used the cancel an ongoing connect procedure. If the FTC succeeds in cancelling the ongoing connection attempt the application will receive a CSR_BT_FTC_CONNECT_CFM with a response code different from CSR_BT_OBEX_SUCCESS_RESPONSE_CODE.

Parameters

type Signal identity, CSR_BT_FTC_CANCEL_CONNECT_REQ

4.17 CSR_BT_FTC_SECURITY_OUT

Parameters / Primitives	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_FTC_SECURITY_OUT_REQ	✓	✓	✓		
CSR_BT_FTC_SECURITY_OUT_CFM	✓			✓	✓

Table 17: CSR_BT_FTC_SECURITY_OUT Primitives

Description

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

The *CSR_BT_SECURITY_OUT_REQ* signal sets up the security level for new outgoing connections. Already established and pending connections are not altered. Note that *authorisation* should not be used for outgoing connections as that may be confusing for the user – there is really no point in requesting an outgoing connection and afterwards having to authorise as they are both locally-only decided procedures.

Note, that any attempts to set security to a less secure level than the mandatory security level will be rejected. See csr_bt_profiles.h for mandatory security settings. The default settings used by CSR Synergy Bluetooth are set to require authentication and encryption.

Note that if MITM protection is requested and the remote device does not have the required IO capabilities, pairing/bonding will fail and connections to the remote device *cannot* be made. See [SC] for further details.

Parameters

type — Signal identity CSR_BT_FTC_SECURITY_OUT_REQ/CFM.

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

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

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

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

- CSR_BT_SEC_AUTHORISATION: Require authorisation
- CSR_BT_SEC_ AUTHENTICATION: Require authentication
- CSR_BT_SEC_SEC_ENCRYPTION: Require encryption (implies authentication)
- CSR_BT_SEC_MITM: Require MITM protection (implies encryption)

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

4.18 CSR_BT_FTC_ COPYING_OBJ

Primitives	type	*ucs2SrcName	*ucs2DestName	responseCode
CSR_BT_FTC_COPYING_OBJ_REQ	✓	✓	✓	
CSR_BT_FTC_COPYING_OBJ_CFM	✓			✓

Table 18: CSR_BT_FTC_COPYING_OBJ Primitives

Description

This command copies an object from one location to another on the FTP Server.

Parameters

type — Signal identity, CSR_BT_FTC_COPYING_OBJ_REQ/CFM.

*ucs2SrcName — A null terminated 16 bit Unicode text string (UCS2) containing the name of the file or folder on the Server that must be copied.

The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.

*ucs2DestName — A null terminated 16 bit Unicode text string (UCS2) containing the name of the destination object.

The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.

responseCode — The valid response codes are defined in csr_bt_obex.h. The responseCode CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates OBEX operation were a success, any other response code indicates a failure.

It is recommended that the following error response Code means:

CSR_BT_OBEX_NOT_FOUND_RESPONSE_CODE	Source object or destination folder does not exist.
CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE	Cannot read source object or create object in destination folder, permission denied.
CSR_BT_OBEX_DATABASE_FULL_RESPONSE_CODE	Cannot create object in destination folder, out of memory.
CSR_BT_OBEX_CONFLICT_RESPONSE_CODE	Cannot create object in destination folder, sharing violation, command reserved/busy.
CSR_BT_OBEX_NOT_IMPLEMENTED_RESPONSE_CODE	The CSR_BT_FTC_COPYING_OBJ_REQ command are not supported.
CSR_BT_OBEX_NOT_MODIFIED_RESPONSE_CODE	Cannot create folder/file, destination folder/file already exits.

4.19 CSR_BT_FTC_ MOVING_OBJ

Primitives	type	*ucs2SrcName	*ucs2DestName	responseCode
CSR_BT_FTC_MOVING_OBJ_REQ	✓	✓	✓	
CSR_BT_FTC_MOVING_OBJ_CFM	✓			✓

Table 19: CSR_BT_FTC_MOVING_OBJ Primitives

Description

This command moves an object from one location to another on the FTP Server. It can also be use to rename an object.

Parameters

type — Signal identity, CSR_BT_FTC_MOVING_OBJ_REQ/CFM.

*ucs2SrcName — A null terminated 16 bit Unicode text string (UCS2) containing the name of the file or folder on the Server that must be moved or renamed.

The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.

*ucs2DestName — A null terminated 16 bit Unicode text string (UCS2) containing the new name or destination for the object.

The function “CsrUtf82Ucs2String” can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.

responseCode — The valid response codes are defined in csr_bt_obex.h. The responseCode CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates OBEX operation were a success, any other response code indicates a failure.

It is recommended that the following error response Code means:

CSR_BT_OBEX_NOT_FOUND_RESPONSE_CODE	Source object or destination folder does not exist.
CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE	Cannot read source object or create object in destination folder, permission denied.
CSR_BT_OBEX_DATABASE_FULL_RESPONSE_CODE	Cannot create object in destination folder, out of memory.
CSR_BT_OBEX_CONFLICT_RESPONSE_CODE	Cannot create object in destination folder, sharing violation, command reserved/busy.
CSR_BT_OBEX_NOT_IMPLEMENTED_RESPONSE_CODE	The CSR_BT_FTC_MOVING_OBJ_REQ command are not supported.
CSR_BT_OBEX_NOT_MODIFIED_RESPONSE_CODE	Cannot create folder/file, destination folder/file already exits.

4.20 CSR_BT_FTC_SET_OBJ_PERMISSIONS

Primitives	type	*ucs2name	permissions	responseCode
CSR_BT_FTC_SET_OBJ_PERMISSIONS_REQ	✓	✓	✓	
CSR_BT_FTC_SET_OBJ_PERMISSIONS_CFM	✓			✓

Table 20: CSR_BT_FTC_SET_OBJ_PERMISSIONS Primitives

Description

This command sets the access permissions of an object or folder on the FTP Server. The 'ucs2name' parameter specifies the object and the 'permissions' parameter specifies the new permissions for this object. Please note when using this command for folders, it will set the permissions only for the folder; it does not affect the permissions of the contents of the folder.

Parameters

type — Signal identity, CSR_BT_FTC_SET_OBJ_PERMISSIONS_REQ/CFM.

*ucs2name — A null terminated 16 bit Unicode text string (UCS2) containing the name of the file or folder for which permissions must be set.

The function "CsrUtf82Ucs2String" can be used for converting a null terminated UTF8 text string into a null terminated UCS2 text string.

permissions — Permission is a 4-byte unsigned integer (CsrUint32) where the 4 bytes describe bit marks representing the various permission values. It is used for setting "Read", "Write", "Delete" and "Modify" permissions for files and folders. The permissions are applied to three different permissions levels, which are "User", "Group" and "Other" as illustrated below.

Byte 0	Byte 1	Byte 2	Bytes 3
Reserved (Should be set to 0)	User Permissions	Group Permissions	Other Permissions

The bits in each permissions byte have the following meanings:

Bit	Meaning
0	Read. When this bit is set to 1, reading permission is granted. Please note that the client needs both "Read" and "Delete" permission to the source file/folder in order to move it.
1	Write. When this bit is set to 1, writing permission is granted.
2	Delete. When this bit is set to 1, deletion permission is granted
3	Reserved for future use
4	Reserved for future use
5	Reserved for future use
6	Reserved for future use
7	Modify Permissions. When this bit is set to 1 the file access permissions can be changed

In csr_bt_obex.h the following bit masks are defined:

Name	Value
CSR_BT_OBEX_USER_PERMISSIONS_READ_MASK	0x00010000
CSR_BT_OBEX_USER_PERMISSIONS_WRITE_MASK	0x00020000
CSR_BT_OBEX_USER_PERMISSIONS_DELETE_MASK	0x00040000
CSR_BT_OBEX_USER_PERMISSIONS_MODIFY_MASK	0x00800000
CSR_BT_OBEX_GROUP_PERMISSIONS_READ_MASK	0x00000100
CSR_BT_OBEX_GROUP_PERMISSIONS_WRITE_MASK	0x00000200
CSR_BT_OBEX_GROUP_PERMISSIONS_DELETE_MASK	0x00000400
CSR_BT_OBEX_GROUP_PERMISSIONS_MODIFY_MASK	0x00008000
CSR_BT_OBEX_OTHER_PERMISSIONS_READ_MASK	0x00000001
CSR_BT_OBEX_OTHER_PERMISSIONS_WRITE_MASK	0x00000002
CSR_BT_OBEX_OTHER_PERMISSIONS_DELETE_MASK	0x00000004
CSR_BT_OBEX_OTHER_PERMISSIONS_MODIFY_MASK	0x00000080

responseCode

The valid response codes are defined in csr_bt_obex.h. The responseCode CSR_BT_OBEX_SUCCESS_RESPONSE_CODE indicates OBEX operation were a success, any other response code indicates a failure.

It is recommended that the following error response Code means:

CSR_BT_OBEX_NOT_FOUND_RESPONSE_CODE	Source object or destination folder does not exist.
CSR_BT_OBEX_FORBIDDEN_RESPONSE_CODE	Cannot modify the permissions of the destination object/folder, permission denied.
CSR_BT_OBEX_NOT_IMPLEMENTED_RESPONSE_CODE	The CSR_BT_FTC_SET_OBJ_PERMISSIONS_REQ command are not supported.
CSR_BT_OBEX_CONFLICT_RESPONSE_CODE	Cannot modify permissions, sharing violation, command busy.

5 Document References

Document	Reference
FILE TRANSFER PROFILE, Revision V12r00 26 August 2010	[FTP]
IrDA Object Exchange Protocol - IrOBEX Version 1.2 or Version 1.5	[OBEX]
GENERIC OBJECT EXCHANGE PROFILE Revision V20r00 26 August 2010	[GOEP2.0]
Specifications for Ir Mobile Communications (IrMC) Version 1.1 01 March 1999	[IRMC]
CSR Synergy Bluetooth, SC – Security Controller API Description, Document no. api-0102-sc	[SC]
CSR Synergy Bluetooth, AMPM – Alternate MAC and PHY Manager API Description, api-0148-ampm.pdf	[AMPM]
CSR Synergy Bluetooth. CM – Connection Manager API Description, doc. no. api-0101-cm	[CM]

Terms and Definitions

BlueCore®	Group term for CSR's range of Bluetooth wireless technology chips
Bluetooth®	Set of technologies providing audio and data transfer over short-range radio connections
CSR	Cambridge Silicon Radio
FTS	OBEX File Transfer Server
FTC	OBEX File Transfer Client
SRM	Single Response Mode
SRMP	Single Response Mode Parameters
GOEP	Generic Object Exchange Profile
SDS	Service Discovery Server
SIG	Special Interest Group
UniFi™	Group term for CSR's range of chips designed to meet IEEE 802.11 standards
AMPM	Alternate MAC and PHY Manager

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.