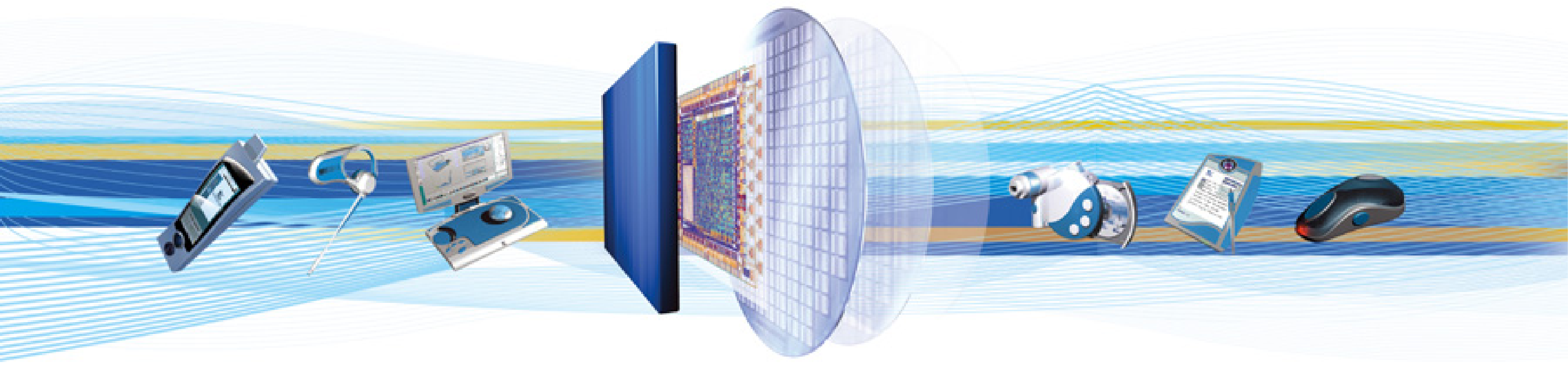
CSR Synergy Bluetooth 18.2.0

OBEX File Transfer Server

API Description

November 2011



Cambridge Silicon Radio Limited

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

Registered in England and Wales 3665875

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

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 Server (FTS). The FTS conforms to the server side of the File Transfer Profile, ref. [FTS].

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

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 OBEX File Transfer Server (FTS) must be activated by the application. When it is activated it is able to provide the application with incoming objects and provide the application the ability to send objects. Furthermore, the server makes available the communication necessary for the client to perform folder browsing.

The FTS provides Service Discovery handling.

The FTS is handling the interpretation of the OBEX packet.

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

2.2 Reference Model

The FTS interfaces to the Connection Manager (CM) and to the Service Discovery Server (SDS).

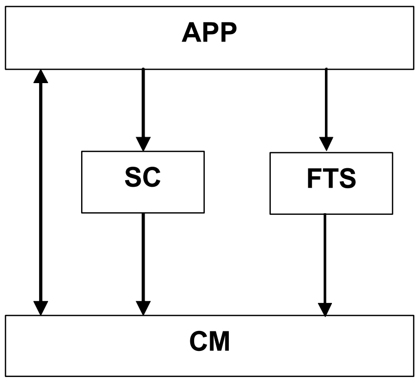


Figure 1: Reference model

2.3 Sequence Overview

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

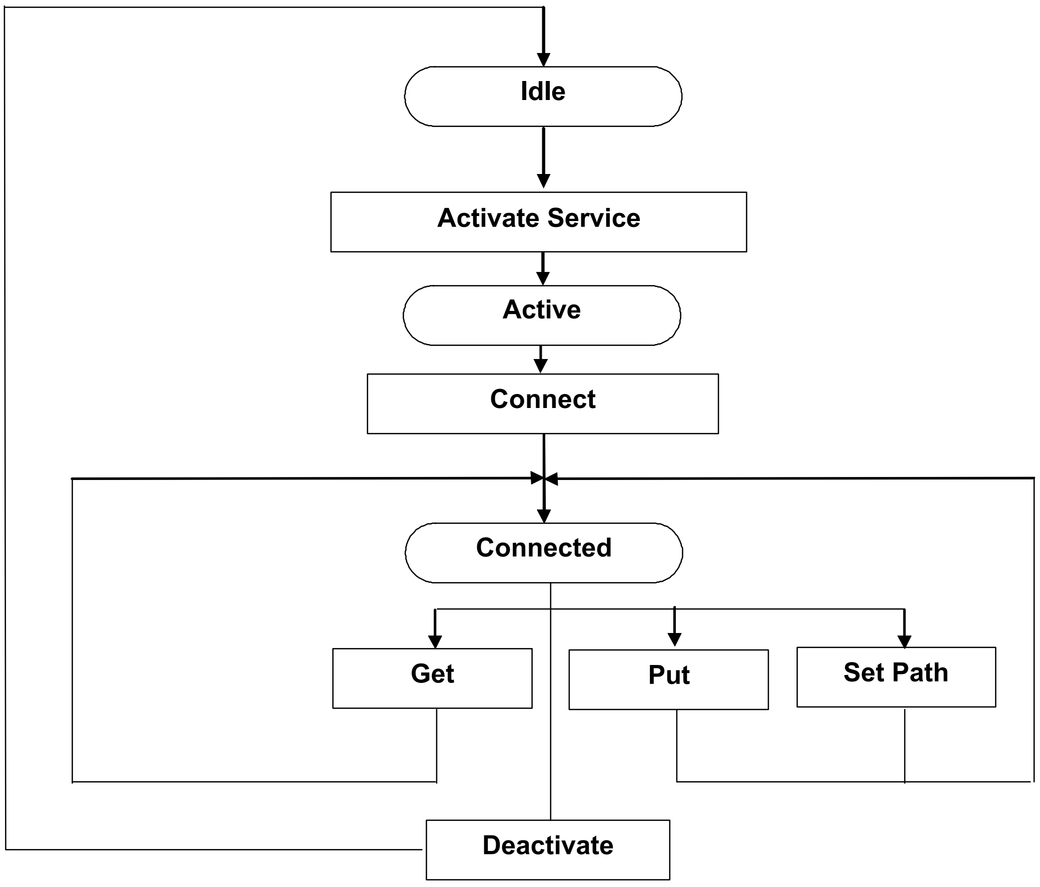


Figure 2: FTS state diagram

3 Interface Description

3.1 Activation

Sending a CSR_BT_FTS_ACTIVATE_REQ to the FTS activates the FTS. The FTS then registers a Service Record, in the Service Discovery Server, and make it connectable. The FTS is now ready to handle incoming requests.

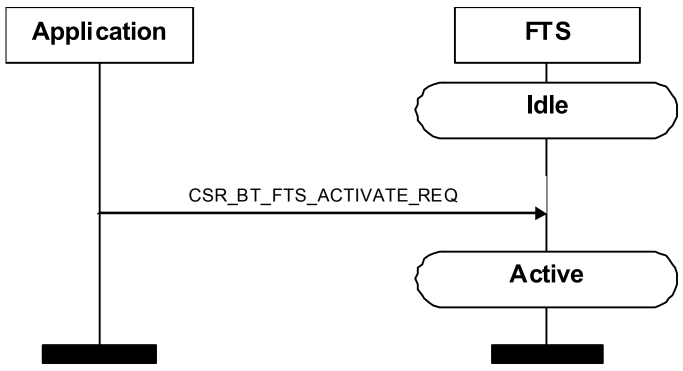


Figure 3: FTS activation

Please note that whether or not the Bluetooth device will be discoverable, i.e. can be found by other Bluetooth devices, it must be controlled by the application. For more information, please refer to [CM]. After initialization of CSR Synergy Bluetooth the Bluetooth® device is set up to be discoverable.

3.2 Connect

When the client is making a connect against the server the first message the application receives is CSR_BT_FTS_CONNECT_IND, this message has a parameter ObexPeerMaxPacketSize indicating the maximum Obex packet size which the application can send down to the FTS in the body in one message response.

The application responses with a CSR_BT_FTS_CONNECT_RES message with the appropriate result code. This message has the parameter ObexMaxPacketSize being the maximum packet (body) size that the application wants to receive from the client. There is a defined CSR_BT_ MAX_OBEX_SIGNAL_LENGHT and the application must use this in the response. This value is calculated from the defined CSR_BT_MAX_OBEX_SIGNAL_LENGTH and both defines are placed in the file csr_bt_obex.h. The value can be between 255 bytes – 64K bytes – 1, see definition in ref. [OBEX]. If the packet size is large it is optimizing for quick file transfer, but the disadvantage will be use for big memory block. The memory use will increase with the packet size.

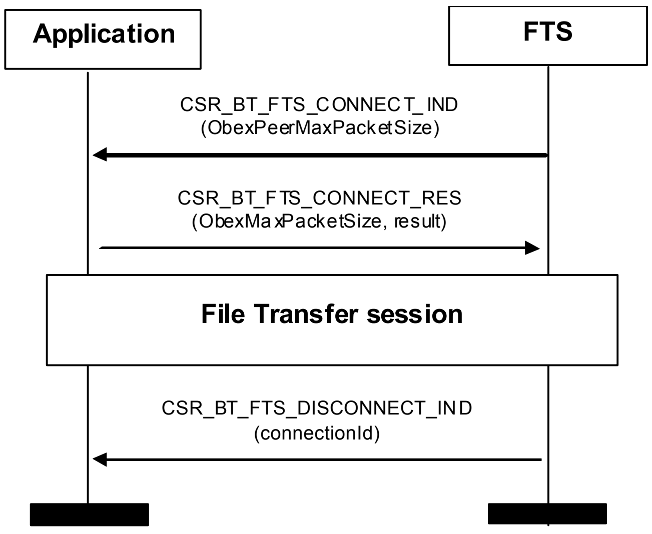


Figure 4: Connection handling

3.3 Browsing Folders

Browsing an object store involves displaying folder contents and setting the ‘current folder’. The CSR_BT_FTS_SET_FOLDER_IND is used for changing the current folder. To display a folder hierarchy starting with the root folder, the client must read the folder content using CSR_BT_FTS_GET_LIST_FOLDER_IND and the application responses with the folder list in the body of the CSR_BT_FTS_GET_LIST_FOLDER_RES message. The fragmentation and use of finalFlag is described under the section 3.5.

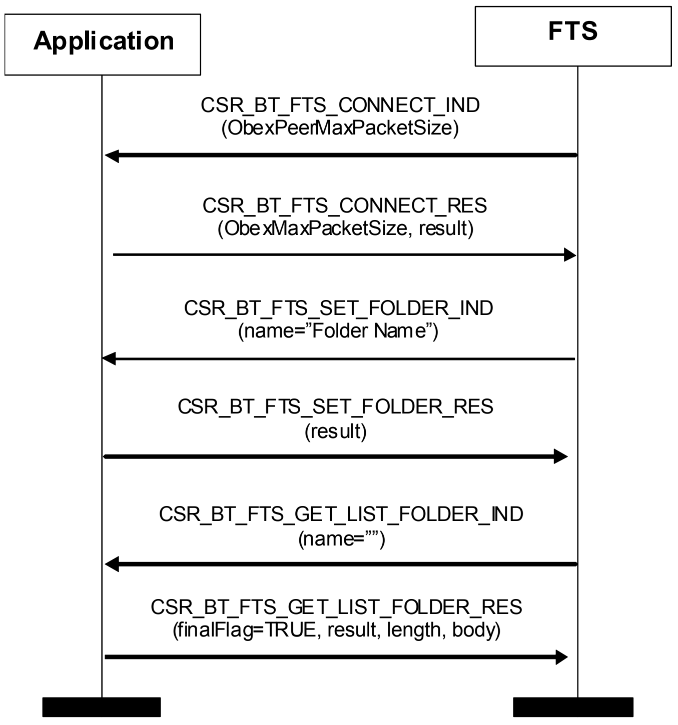

Application
FTS
CSR_BT_FTS_CONNECT_IND
(ObexPeerMaxPacketSize)
CSR_BT_FTS_CONNECT_RES
(ObexMaxPacketSize, result)
CSR_BT_FTS_SET_FOLDER_IND
(name=”Folder Name”)
CSR_BT_FTS_SET_FOLDER_RES
(result)
CSR_BT_FTS_GET_LIST_FOLDER_IND
(name=””)
CSR_BT_FTS_GET_LIST_FOLDER_RES
(finalFlag=TRUE, result, length, body)

Figure 5: Folder browsing handling

3.4 Pushing Objects

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

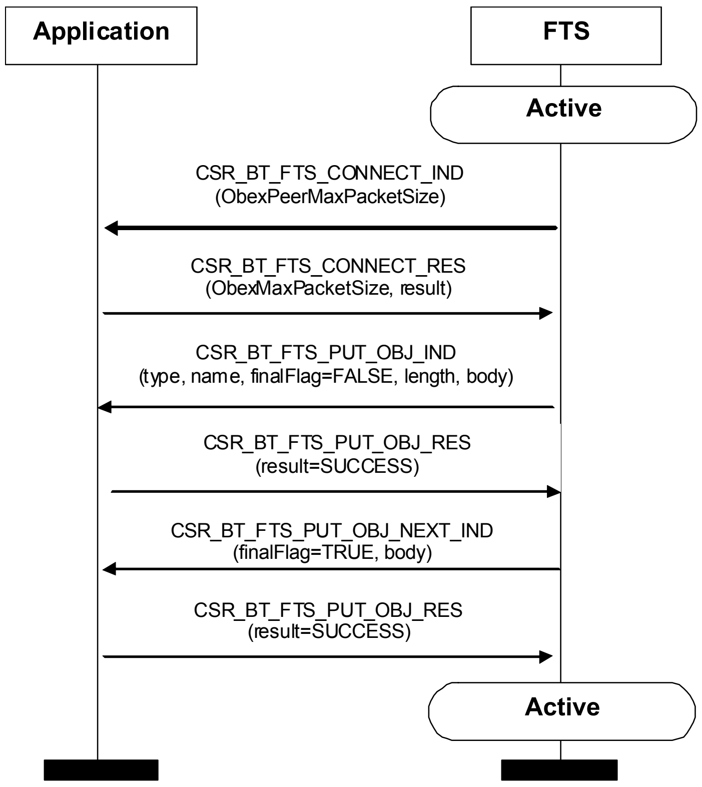

Application
FTS
Active
CSR_BT_FTS_CONNECT_IND (ObexPeerMaxPacketSize)
CSR_BT_FTS_CONNECT_RES (ObexMaxPacketSize, result)
CSR_BT_FTS_PUT_OBJ_IND (type, name, finalFlag=FALSE, length, body)
CSR_BT_FTS_PUT_OBJ_RES (result=SUCCESS)
CSR_BT_FTS_PUT_OBJ_NEXT_IND (finalFlag=TRUE, body)
CSR_BT_FTS_PUT_OBJ_RES (result=SUCCESS)
Active

Figure 6: Incoming message handling

3.5 Pulling Objects

When the FTS receives a request to send an object to the client side, it sends a CSR_BT_FTS_GET_OBJ_IND message to the application with the name parameter set to the requested name of the object. The application responds with a CSR_BT_FTS_GET_OBJ_RES with the appropriate result code. If the application wants to fragment the "body" information due to memory considerations it can set the finalFlag to FALSE and will hence receive CSR_BT_FTS_GET_OBJ_NEXT_INDs until the finalFlag is set to TRUE in the following CSR_BT_FTS_GET_OBJ_NEXT_RES.

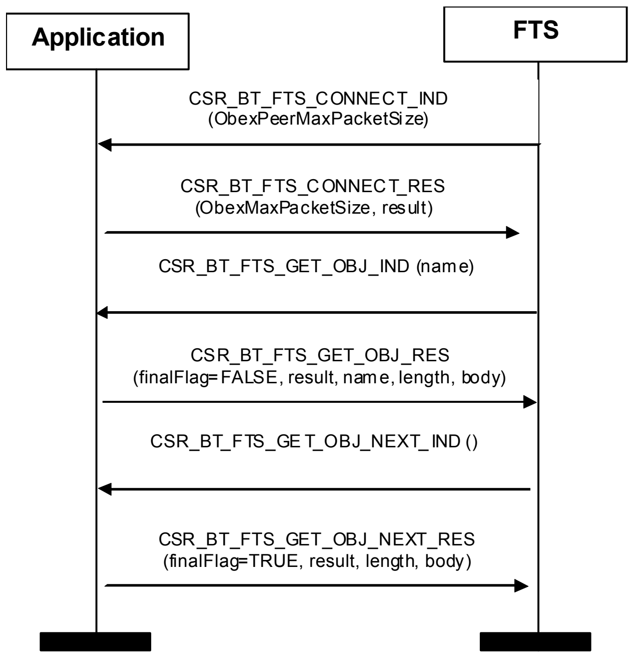

Application
FTS
CSR_BT_FTS_CONNECT_IND (ObexPeerMaxPacketSize)
CSR_BT_FTS_CONNECT_RES (ObexMaxPacketSize, result)
CSR_BT_FTS_GET_OBJ_IND (name)
CSR_BT_FTS_GET_OBJ_RES (finalFlag=FALSE, result, name, length, body)
CSR_BT_FTS_GET_OBJ_NEXT_IND ()
CSR_BT_FTS_GET_OBJ_NEXT_RES (finalFlag=TRUE, result, length, body)

Figure 7: Outgoing message handling

3.6 Deactivation

Sending a CSR_BT_FTS_DEACTIVATION_REQ to the FTS can deactivate the FTS. This procedure can take some time depending on the current FTS activity. When deactivated, the FTS confirms the deactivation with a CSR_BT_FTS_DEACTIVATE_CFM message.

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

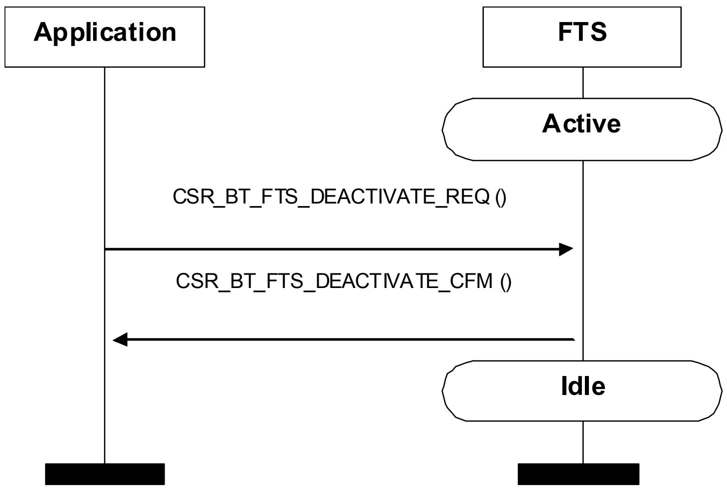


Figure 8: FTS deactivation

3.7 Payload Encapsulated Data

3.7.1 Using Offsets

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

```
CsrBtCommonPrim    type;
CsrUint8               result;
CsrUint16          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.

3.7.2 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 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_fts_prim.h file.

4.1 List of All Primitives

Primitives:	**Reference:**
CSR_BT_FTS_ACTIVATE_REQ	See section 4.2
CSR_BT_FTS_DEACTIVATE_REQ	See section 4.3
CSR_BT_FTS_DEACTIVATE_CFM	See section 4.3
CSR_BT_FTS_CONNECT_IND	See section 4.4
CSR_BT_FTS_CONNECT_RES	See section 4.4
CSR_BT_FTS_AUTHENTICATE_REQ	See section 4.5
CSR_BT_FTS_AUTHENTICATE_CFM	See section 4.5
CSR_BT_FTS_AUTHENTICATE_IND	See section 4.5
CSR_BT_FTS_AUTHENTICATE_RES	See section 4.5
CSR_BT_FTS_PUT_OBJ_IND	See section 4.6
CSR_BT_FTS_PUT_OBJ_RES	See section 4.6
CSR_BT_FTS_PUT_OBJ_NEXT_IND	See section 4.7
CSR_BT_FTS_PUT_OBJ_NEXT_RES	See section 4.7
CSR_BT_FTS_DEL_OBJ_IND	See section 4.8
CSR_BT_FTS_DEL_OBJ_RES	See section 4.8
CSR_BT_FTS_GET_OBJ_IND	See section 4.9
CSR_BT_FTS_GET_OBJ_RES	See section 4.9
CSR_BT_FTS_GET_OBJ_NEXT_IND	See section 4.10
CSR_BT_FTS_GET_OBJ_NEXT_RES	See section 4.10
CSR_BT_FTS_GET_LIST_FOLDER_IND	See section 4.11
CSR_BT_FTS_GET_LIST_FOLDER_RES	See section 4.11
CSR_BT_FTS_GET_LIST_FOLDER_NEXT_IND	See section 4.12
CSR_BT_FTS_GET_LIST_FOLDER_NEXT_RES	See section 4.12
CSR_BT_FTS_SET_FOLDER_IND	See section 4.13
CSR_BT_FTS_SET_FOLDER_RES	See section 4.13
CSR_BT_FTS_SET_BACK_FOLDER_IND	See section 4.14
CSR_BT_FTS_SET_BACK_FOLDER_RES	See section 4.14
CSR_BT_FTS_SET_ROOT_FOLDER_IND	See section 4.15
CSR_BT_FTS_SET_ROOT_FOLDER_RES	See section 4.15
CSR_BT_FTS_SET_ADD_FOLDER_IND	See section 4.16
CSR_BT_FTS_SET_ADD_FOLDER_RES	See section 4.16
CSR_BT_FTS_ABORT_IND	See section 4.17
CSR_BT_FTS_DISCONNECT_IND	See section 4.18
CSR_BT_FTS_SECURITY_IN_REQ	See section 4.19
CSR_BT_FTS_SECURITY_IN_CFM	See section 4.19
CSR_BT_FTS_COPY_OBJ_IND	See section 4.20
CSR_BT_FTS_COPY_OBJ_RES	See section 4.20
CSR_BT_FTS_MOVE_OBJ_IND	See section 4.21

Primitives:	Reference:
CSR_BT_FTS_MOVE_OBJ_RES	See section 4.21
CSR_BT_FTS_SET_OBJ_PERMISSIONS_IND	See section 4.22
CSR_BT_FTS_SET_OBJ_PERMISSIONS_RES	See section 4.22

Table 1: List of all primitives

4.2 CSR_BT_FTS_ACTIVATE

Parameters / Primitives	type	appHandle	obexMacPacketSize	windowSize	srmEnable
CSR_BT_FTS_ACTIVATE_REQ	✓	✓	✓	✓	✓

Table 2: CSR_BT_FTS_ACTIVATE Primitives

Description

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

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

The FTS will remain activated until a CSR_BT_FTS_DEACTIVATE_REQ is received.

Parameters

type	Signal identity, CSR_BT_FTS_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.
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	TRUE enables local support for Single Response Mode (SRM). If SRM is enabled FTS 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_FTS_DEACTIVATE

Parameters / Primitives	type
CSR_BT_FTS_DEACTIVATE_REQ	✓
CSR_BT_FTS_DEACTIVATE_CFM	✓

Table 3: CSR_BT_FTS_DEACTIVATE Primitives

Description

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

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

The signal will stop any ongoing transaction.

Parameters

type	Signal identity, CSR_BT_FTS_DEACTIVATE_REQ/CFM.

4.4 CSR_BT_FTS_CONNECT

Primitives \ Parameters	type	connectionId	obexPeerMaxPacketSize	deviceAddr	responseCode	length	count	btConnId
CSR_BT_FTS_CONNECT_IND	✓	✓	✓	✓		✓	✓	✓
CSR_BT_FTS_CONNECT_RES	✓	✓			✓			

Table 4: CSR_BT_FTS_CONNECT Primitives

Description

This signal is indicating that a FTP client is starting file transfer session. The application can then accept or deny the result and has to return the connectionId received in the indication.

Parameters

type	Signal identity, CSR_BT_FTS_CONNECT_IND/RES.
connectionId	Is the connection Id for this session, the FTP client will use this Id in the request.
obexPeerMaxPacketSize	The maximum OBEX packet size being allowed to send to the client application.
deviceAddr	The Bluetooth address which is connected to the device
responseCode	The valid reponse codes are defined (in csr_bt_obex.h). For accepting a connection the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure to make a connection. 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, FTS 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 which shall be used when using AMPM, for more information please refer to [AMPM].

4.5 CSR_BT_FTS_AUTHENTICATE

Primitives \ Parameters	type	options	realmLength	*realm	deviceAddr	*password	passwordLength	*userId
CSR_BT_FTS_AUTHENTICATE_REQ	✓		✓	✓		✓	✓	✓
CSR_BT_FTS_AUTHENTICATE_CFM	✓							
CSR_BT_FTS_AUTHENTICATE_IND	✓	✓	✓	✓	✓			
CSR_BT_FTS_AUTHENTICATE_RES	✓					✓	✓	✓

Table 5: CSR_BT_FTS_AUTHENTICATE Primitives

Description

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

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

Parameters

type	Signal identity, CSR_BT_FTS_AUTHENTICATE_REQ/CFM/IND/RES.
options	Challenge information of type CsrUint8. Bit 0 controls the responding of a valid user Id. If bit 0 is set it means that the application must response with a user Id in a CSR_BT_FTS_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 always set to 0x0000 for the CSR_BT_FTS_AUTHENTICATE_IND and in CSR_BT_FTS_AUTHENTICATE_REQ the 'realmLength is ignored.
* 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 for the CSR_BT_FTS_AUTHENTICATE_IND and in CSR_BT_FTS_AUTHENTICATE_REQ the contents of the ‘realm’ is ignored right now.

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

deviceAddr	The Bluetooth address of the device that has initiated the OBEX authentication procedure
*password	Containing the challenge password of the OBEX authentication. This is a pointer which shall be allocated by the application.
passwordLength	The length of the challenge password.
*userId	Zero terminated string (ASCII) containing the userId for the authentication. This is a pointer which shall be allocated by the application. Note in CSR_BT_FTS_AUTHENTICATE_REQ the userid is ignored right now.

4.6 CSR_BT_FTS_PUT_OBJ

Primitives \ Parameters	type	connectionId	finalFlag	lengthOfObject	ucs2nameOffset	bodyLength	bodyOffset	responseCode	payloadLength	*payload	srmpOn
CSR_BT_FTS_PUT_OBJ_IND	✓	✓	✓	✓	✓	✓	✓		✓	✓	
CSR_BT_FTS_PUT_OBJ_RES	✓	✓						✓			✓

Table 6: CSR_BT_FTS_PUT_OBJ Primitives

Description

The FTS passes incoming objects on to the application with the CSR_BT_FTS_PUT_OBJ_IND signal. The application can then store the objects in the current folder. The result of the store operation is given to the FTS with the CSR_BT_FTS_PUT_OBJ_RES signal. The result can contain error codes corresponding to the reason for failure.

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

Parameters

type	Signal identity, CSR_BT_FTS_PUT_OBJ_IND/RES.
connectionId	The connection Id for this session, the FTP client will use this Id in the request.
finalFlag	Indicates that the body (object) fits the whole object or that it's the last part.
lengthOfObject	The total length of the object to receive.
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 relative to the payload of the object data itself.
reponseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabled, see section 4.2, the FTP server can instruct the FTP Client to wait for the next response packet during a PUT Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first PUT response and may be used in consecutive PUT response packets to cause the Client to continue its wait;

however, once the srmpOn parameter is FALSE in a PUT response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.7 CSR_BT_FTS_PUT_OBJ_NEXT

Primitives \ Parameters	type	connectionId	finalFlag	bodyLength	bodyOffset	responseCode	payloadLength	*payload	srmpOn
CSR_BT_FTS_PUT_OBJ_NEXT_IND	✓	✓	✓	✓	✓		✓	✓	
CSR_BT_FTS_PUT_OBJ_NEXT_RES	✓	✓				✓			✓

Table 7: CSR_BT_FTS_PUT_OBJ_NEXT Primitives

Description

The FTS passes incoming objects on to the application with the CSR_BT_FTS_PUT_OBJ_IND signal. In case the object is too large to fit into one OBEX packet, the first part is in the CSR_BT_FTS_PUT_OBJ_IND and the next part of the object will appear in the CSR_BT_FTS_PUT_OBJ_NEXT_IND until the finalFlag parameter is set.

Parameters

type	Signal identity, CSR_BT_FTS_PUT_OBJ_NEXT_IND/RES.
connectionId	Is the connection Id for this session, the FTP 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.
bodyLength	The length of the body (object).
bodyOffset	Offset relative to payload for the actual body data.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure.´ The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabled, see section 4.2, the FTP server can instruct the FTP Client to wait for the next response packet during a PUT Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first PUT response and may be used in consecutive PUT response packets to cause the Client to continue its wait; however, once the srmpOn parameter is FALSE in a PUT response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.8 CSR_BT_FTS_DEL_OBJ

Primitives \ Parameters	type	connectionId	ucs2nameOffset	responseCode	payloadLength	*payload	srmpOn
CSR_BT_FTS_DEL_OBJ_IND	✓	✓	✓		✓	✓	
CSR_BT_FTS_DEL_OBJ_RES	✓	✓		✓			✓

Table 8: CSR_BT_FTS_DEL_OBJ Primitives

Description

This signal is used for deleting objects (files or folders) on the FTS. The application can then delete the object in the current folder. The result of the delete operation is given to the FTS with the response signal. The result can contain error codes corresponding to the reason for failure or if the application does not permit this operation from the client. The application can also authenticate the client before accepting the operation, is done with the CSR_BT_FTS_AUTHENTICATE_REQ.

Parameters

type	Signal identity, CSR_BT_FTS_DEL_OBJ_IND/RES.
connectionId	The connection Id for this session, the FTP client will use this Id in the request.
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
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	Reserved for future use. Set to FALSE.

4.9 CSR_BT_FTS_GET_OBJ

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	lengthOfObject	ucs2nameOffset	bodyLength	*body	payloadLength	*payload	srmpOn
CSR_BT_FTS_GET_OBJ_IND	✓	✓				✓			✓	✓	
CSR_BT_FTS_GET_OBJ_RES	✓	✓	✓	✓	✓		✓	✓			✓

Table 9: CSR_BT_FTS_GET_OBJ Primitives

Description

To retrieve an object from the FTP server specified by the name parameter in the CSR_BT_FTS_GET_OBJ_IND signal, the server responses with a CSR_BT_FTS_GET_OBJ_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_FTS_GET_NEXT_IND), only the last response has the finalFlag set and the application, must remember the request (name) until the last response (CSR_BT_FTS_GET_NEXT_RES). In case the result is different from success, the other parameters are invalid and not used.

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

Parameters

type	Signal identity, CSR_BT_FTS_GET_OBJ_IND/RES.
connectionId	The connection Id for this session, the FTP 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 valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure. 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. 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 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 **Note:** This value can be 0 in the CSR_BT_FTS_GET_OBJ_IND while the offset is valid (contrary to what the text in section 3.7 states). Use the *payloadLength* to check if the name is valid.
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. In the CSR_BT_FTS_GET_OBJ_IND, this parameter will be greater than 0 if the ucs2nameOffset is valid.
*payload	OBEX payload data. Offsets are relative to this pointer.
srmpOn	If Single Response Mode is enabled, see section 4.2, the FTP server can instruct the FTP Client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause the Client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.10 CSR_BT_FTS_GET_OBJ_NEXT

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	bodyLength	*body	srmpOn
CSR_BT_FTS_GET_OBJ_NEXT_IND	✓	✓					
CSR_BT_FTS_GET_OBJ_NEXT_RES	✓	✓	✓	✓	✓	✓	✓

Table 10: CSR_BT_FTS_GET_OBJ_NEXT Primitives

Description

To retrieve multiple parts of an object from the server, the first packet is the CSR_BT_FTS_GET_OBJ_RES, the next packet is sent to the CSR_BT_FTS_GET_OBJ_RES after receiving the CSR_BT_FTS_GET_OBJ_IND signal. The last response has to set the parameter finalFlag. The application must remember the name parameter in the first CSR_BT_FTS_GET_OBJ_IND, then dealing with multiple objects.

Parameters

type	Signal identity, CSR_BT_FTS_GET_OBJ_NEXT_IND/RES.
connectionId	Is the connection Id for this session, the FTP 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 valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure. 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.
srmpOn	If Single Response Mode is enabled, see section 4.2, the FTP server can instruct the FTP Client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause the Client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.11 CSR_BT_FTS_GET_LIST_FOLDER

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	lengthOfObject	ucs2nameOffset	bodyLength	*body	payloadLength	*payload	srmpOn
CSR_BT_FTS_GET_LIST_FOLDER_IND	✓	✓				✓			✓	✓	
CSR_BT_FTS_GET_LIST_FOLDER_RES	✓	✓	✓	✓	✓		✓	✓			✓

Table 11: CSR_BT_FTS_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 application 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_FTS_AUTHENTICATE_REQ 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. If the response is large enough to require multiple requests (CSR_BT_FTS_GET_LIST_FOLDER_NEXT_IND), only the last response has the finalFlag set and the application, but remember the request (name) until the last response (CSR_BT_FTS_GET_LIST_FOLDER_NEXT_RES). In case the result is different from success, the other parameters except for connectionId are invalid and not used.

Parameters

type	Signal identity, CSR_BT_FTS_GET_LIST_FOLDER_IND/RES.
connectionId	The connection Id for this session, the FTP 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 valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure. 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. 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 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. **Note:** This value can be 0 in the CSR_BT_FTS_GET_OBJ_IND while the offset is valid (contrary to what the text in section 3.7 states). Use the *payloadLength* to check if the name is valid.
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.
srmpOn	If Single Response Mode is enabled, see section 4.2, the FTP server can instruct the FTP Client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause the Client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.12 CSR_BT_FTS_GET_LIST_FOLDER_NEXT

Primitives \ Parameters	type	connectionId	finalFlag	responseCode	bodyLength	*body	srmpOn
CSR_BT_FTS_GET_LIST_FOLDER_NEXT_IND	✓	✓					
CSR_BT_FTS_GET_LIST_FOLDER_NEXT_RES	✓	✓	✓	✓	✓	✓	✓

Table 12: CSR_BT_FTS_GET_LIST_FOLDER_NEXT Primitives

Description

This signal is used if the pull content of folder is too large to fit into one OBEX packet. The first packet is sent in CSR_BT_FTS_GET_LIST_FOLDER_RES, the next part is sent in CSR_BT_FTS_GET_LIST_FOLDER_NEXT_RES after receiving the CSR_BT_FTS_GET_LIST_FOLDER_NEXT_IND signal. The last response has to set the parameter finalFlag.

Parameters

type	Signal identity, CSR_BT_FTS_GET_LIST_FOLDER_NEXT_IND/RES.
connectionId	Is the connection Id for this session, the FTP 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 valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure. 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.
srmpOn	If Single Response Mode is enabled, see section 4.2, the FTP server can instruct the FTP Client to wait for the next response packet during a GET Operation by setting the srmpOn parameter TRUE. If used, the srmpOn parameter shall be TRUE in the first GET response, and may be used in consecutive GET response packets to cause the Client to continue its wait; however, once the srmpOn parameter is FALSE in a GET response, the srmpOn parameter are consider to be FALSE for the duration of the operation.

4.13 CSR_BT_FTS_SET_FOLDER

Primitives \ Parameters	type	connectionId	ucs2nameOffset	responseCode	payloadLength	*payload
CSR_BT_FTS_SET_FOLDER_IND	✓	✓	✓		✓	✓
CSR_BT_FTS_SET_FOLDER_RES	✓	✓		✓		

Table 13: CSR_BT_FTS_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. The result of the change folder operation is given in the response signal. The result can contain error codes corresponding to the reason for failure, the folder does not exist, or if the application does not permit this operation from the client. The application can also authenticate the client before accepting the operation, is done with CSR_BT_FTS_AUTHENTICATE_REQ.

Parameters

type	Signal identity, CSR_BT_FTS_SET_FOLDER_IND/RES.
connectionId	The connection Id for this session, the FTP client will use this Id in the request.
ucs2nameOffset	Payload relative 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
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX payload data. Offsets are relative to this pointer.

4.14 CSR_BT_FTS_SET_BACK_FOLDER

Primitives \ Parameters	type	connectionId	responseCode
CSR_BT_FTS_SET_BACK_FOLDER_IND	✓	✓	
CSR_BT_FTS_SET_BACK_FOLDER_RES	✓	✓	✓

Table 14: CSR_BT_FTS_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 response signal. If the current folder is the root folder the response is the CSR_BT_OBEX_NOT_FOUND_RESPONSE_CODE result code.

Parameters

type	Signal identity, CSR_BT_FTS_SET_BACK_FOLDER_IND/RES.
connectionId	The connection Id for this session. The FTP client will use this Id in the request.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.

4.15 CSR_BT_FTS_SET_ROOT_FOLDER

Parameters / Primitives	type	connectionId
CSR_BT_FTS_SET_ROOT_FOLDER_IND	✓	✓
CSR_BT_FTS_SET_ROOT_FOLDER_RES	✓	✓

Table 15: CSR_BT_FTS_SET_ROOT_FOLDER Primitives

Description

This signal is used for setting the current folder back to the root folder. There is no result parameter for this response, it is always success for this operation.

Parameters

type	Signal identity, CSR_BT_FTS_SET_ROOT_FOLDER_IND/RES.
connectionId	The connection Id for this session, the FTP client will use this Id in the request.

4.16 CSR_BT_FTS_SET_ADD_FOLDER

Primitives \ Parameters	type	connectionId	ucs2nameOffset	responseCode	payloadLength	*payload
CSR_BT_FTS_SET_ADD_FOLDER_IND	✓	✓	✓		✓	✓
CSR_BT_FTS_SET_ADD_FOLDER_RES	✓	✓		✓		

Table 16: CSR_BT_FTS_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 the folder is placed in the current folder. The result of the create folder operation is given in the response signal. The result can contain error codes corresponding to the reason for failure. The folder application does not permit this operation from the client. The application can also authenticate the client before accepting the operation, which is done with CSR_BT_FTS_AUTHENTICATE_REQ.

Parameters

type	Signal identity, CSR_BT_FTS_SET_ADD_FOLDER_IND/RES.
connectionId	The connection Id for this session, the FTP client will use this Id in the request.
ucs2nameOffset	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.
responseCode	The valid response codes are defined (in csr_bt_obex.h). For success in the request the code is CSR_BT_OBEX_SUCCESS_RESPONSE_CODE, any other response code indicates a failure. The responseCodes are defined in (csr_bt_obex.h) with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol.
payloadLength	Number of bytes in the payload structure.
*payload	OBEX payload data. Offsets are relative to this pointer.

4.17 CSR_BT_FTS_ABORT

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

Table 17: CSR_BT_FTS_ABORT Primitives

Description

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

Parameters

type	Signal identity, CSR_BT_FTS_ABORT_IND.
connectionId	The connection Id for this session, the FTP 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 abort description string.
payloadLength	of bytes in the payload structure.
*payload	OBEX payload data. Offsets are relative to this pointer.

4.18 CSR_BT_FTS_DISCONNECT

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

Table 18: CSR_BT_FTS_DISCONNECT Primitives

Description

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

Parameters

type	Signal identity, CSR_BT_FTS_DISCONNECT_IND.
connectionId	The connection Id for this session, the FTP client will use this Id in the request.
deviceAddr	The Bluetooth address which is connected to the device.
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

4.19 CSR_BT_FTS_SECURITY_IN

Parameters / Primitives	type	appHandle	secLevel	resultCode	resultSupplier
CSR_BT_FTS_SECURITY_IN_REQ	✓	✓	✓		
CSR_BT_FTS_SECURITY_IN_CFM	✓			✓	✓

Table 19: CSR_BT_FTS_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_FTS_SECURITY_IN_REQ/CFM.

appHandle Application handle to which the confirm message is sent.

secLevel The application must specify one of the following values:

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

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

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

resultCode The result code of the operation. Possible values depend on the value of resultSupplier. If e.g. the resultSupplier == CSR_BT_SUPPLIER_CM then the possible result codes can be found in csr_bt_cm_prim.h. If the resultSupplier == CSR_BT_SUPPLIER_OBEX then the possible result codes can be found in csr_bt_obex.h. All values which are currently not specified in the respective prim.h

	files or csr_bt_obex.h are regarded as reserved and the application should consider them as errors.
resultSupplier	This parameter specifies the supplier of the result given in resultCode. Possible values can be found in csr_bt_result.h

4.20 CSR_BT_FTS_COPY_OBJ

Primitives \ Parameters	type	connectionId	ucs2srcNameOffset	ucs2destNameOffset	*payload	payloadLength	responseCode
CSR_BT_FTS_COPY_OBJ_IND	✓	✓	✓	✓	✓	✓	
CSR_BT_FTS_COPY_OBJ_RES	✓	✓					✓

Table 20: CSR_BT_FTS_COPY_OBJ Primitives

Description

This signal is use for copying an object from one location to another on the server. The result of the Copy operation is given in the response signal. The result can contain error codes corresponding to the reason for failure if the application does not permit this operation from the client. The application can also authenticate the client before accepting this operation, which is done with CSR_BT_FTS_AUTHENTICATE_REQ, see section 4.5.

Parameters

type	Signal identity, CSR_BT_FTS_COPY_OBJ_IND/RES.
connectionId	Is the connection Id for this session, the FTP client will use this Id in the request.
ucs2srcNameOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the name of the file or folder that must be copied. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
ucs2destNameOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the name of the destination object. The function “CsrUcs2ByteString2Utf8” can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
*payload	OBEX payload data. Offsets are relative to this pointer.
* payloadLength	Number of bytes in the payload structure.
responseCode	The valid response codes are defined in csr_bt_obex.h with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. For success the response code must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure. It is recommended that the following error response Code is used when:

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	This command is not supported.
CSR_BT_OBEX_NOT_MODIFIED_RESPONSE_CODE	Cannot create folder/file, destination folder/file already exits.

4.21 CSR_BT_FTS_MOVE_OBJ

Primitives \ Parameters	type	connectionId	ucs2srcNameOffset	ucs2destNameOffset	*payload	payloadLength	responseCode
CSR_BT_FTS_MOVE_OBJ_IND	✓	✓	✓	✓	✓	✓	
CSR_BT_FTS_MOVE_OBJ_RES	✓	✓					✓

Table 21: CSR_BT_FTS_MOVE_OBJ Primitives

Description

This signal is use for moving an object from one location to another on the server. The result of the Moving operation is given in the response signal. The result can contain error codes corresponding to the reason for failure if the application does not permit this operation from the client. The application can also authenticate the client before accepting this operation, which is done with CSR_BT_FTS_AUTHENTICATE_REQ, see section 4.5.

Parameters

type	Signal identity, CSR_BT_FTS_MOVE_OBJ_IND/RES.
connectionId	Is the connection Id for this session, the FTP client will use this Id in the request.
ucs2srcNameOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the name of the file or folder that must be moved or renamed. The function "CsrUcs2ByteString2Utf8" can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
ucs2destNameOffset	Payload relative offset of a null terminated 16 bit Unicode text string (UCS2) containing the new name or destination for the object. The function "CsrUcs2ByteString2Utf8" can be used for converting a null terminated UCS2 text string into a null terminated UTF8 text string.
*payload	OBEX payload data. Offsets are relative to this pointer.
* payloadLength	Number of bytes in the payload structure.
responseCode	The valid response codes are defined in csr_bt_obex.h with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. For success the response code must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure. It is recommended that the following error response Code is used when:

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	This command is not supported.
CSR_BT_OBEX_NOT_MODIFIED_RESPONSE_CODE	Cannot create folder/file, destination folder/file already exits.

4.22 CSR_BT_FTS_SET_OBJ_PERMISSIONS

Primitives \ Parameters	type	connectionId	ucs2nameOffset	permissions	*payload	payloadLength	responseCode
CSR_BT_FTS_SET_OBJ_PERMISSIONS_IND	✓	✓	✓	✓	✓	✓	
CSR_BT_FTS_SET_OBJ_PERMISSIONS_IND	✓	✓					✓

Table 22: CSR_BT_FTS_SET_OBJ_PERMISSIONS Primitives

Description

This signal is use to set the access permissions of an object or folder on the server. The result of this operation is given in the response signal. The result can contain error codes corresponding to the reason for failure if the application does not permit this operation from the client. The application can also authenticate the client before accepting this operation, which is done with CSR_BT_FTS_AUTHENTICATE_REQ, see section 4.5.

Parameters

type	Signal identity, CSR_BT_FTS_SET_OBJ_PERMISSIONS_IND /RES.
connectionId	Is the connection Id for this session, the FTP client will use this Id in the request.
ucs2nameOffset	Payload relative offset of 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 "CsrUcs2ByteString2Utf8" can be used for converting a null terminated UCS2 text string into a null terminated UTF8 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

*payload	OBEX payload data. Offsets are relative to this pointer.
* payloadLength	Number of bytes in the payload structure.
responseCode	The valid response codes are defined in csr_bt_obex.h with the following type CsrBtObexResponseCode and can also be found in IrDA Object Exchange Protocol. For success the response code must be CSR_BT_OBEX_SUCCESS_RESPONSE_CODE. Any other response code indicates a failure.

It is recommended that the following error response Code is used when:

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	This command is 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

Life Support Policy and Use in Safety-critical Compliance

Performance and Conformance

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