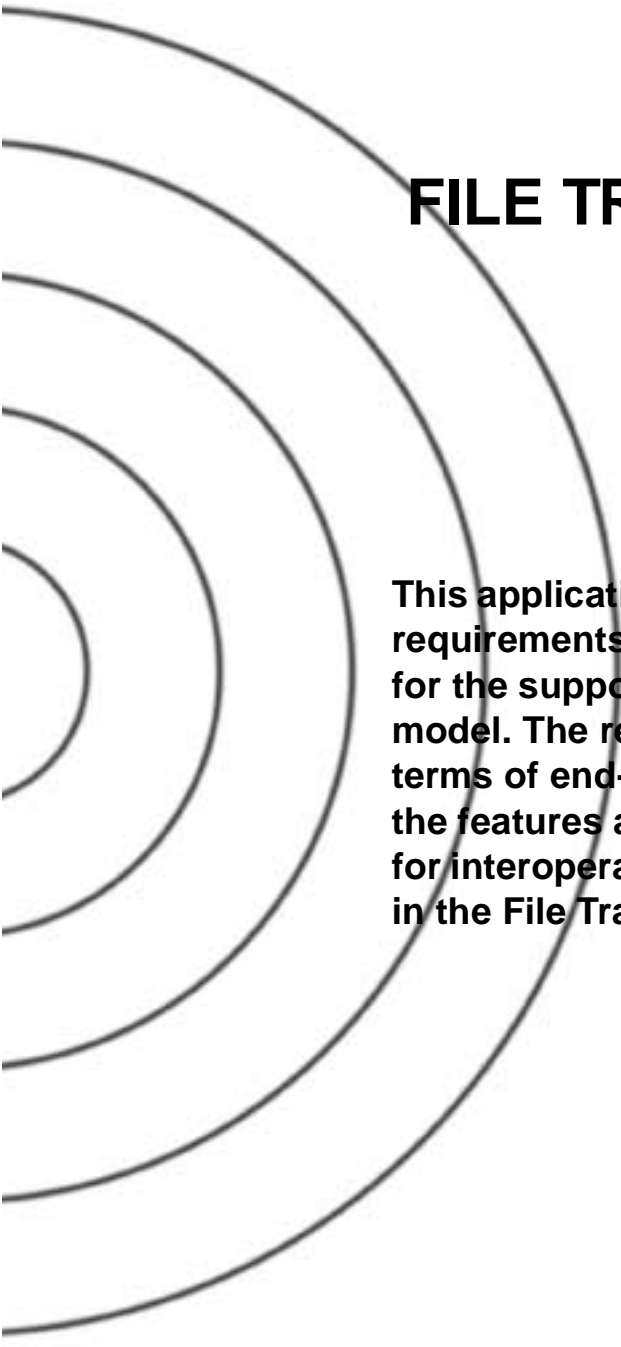


Part K:12

FILE TRANSFER PROFILE



This application profile defines the application requirements for Bluetooth devices necessary for the support of the File Transfer usage model. The requirements are expressed in terms of end-user services, and by defining the features and procedures that are required for interoperability between Bluetooth devices in the File Transfer usage model.



CONTENTS

1	Introduction	360
1.1	Scope	360
1.2	Bluetooth profile structure	360
1.3	Bluetooth OBEX-Related Specifications	361
1.4	Symbols and conventions	362
1.4.1	Requirement status symbols	362
1.4.2	Signaling diagram conventions	363
2	Profile overview	364
2.1	Profile stack	364
2.2	Configurations and roles	364
2.3	User requirements and scenarios	365
2.4	Profile fundamentals	365
3	User interface aspects	367
3.1	File Transfer Mode selection, Servers	367
3.2	Function Selection, Clients.....	367
3.3	Application usage.....	368
4	Application layer	370
4.1	Feature overview.....	370
4.2	Folder Browsing	370
4.3	Object Transfer	372
4.4	Object Manipulation	373
5	OBEX	374
5.1	OBEX Operations Used	374
5.2	OBEX Headers	375
5.3	Initialization of OBEX	375
5.4	Establishment of OBEX session	375
5.5	Browsing Folders	376
5.5.1	Pulling a Folder Listing Object.....	376
5.5.2	Setting the Current Folder (Forward)	376
5.5.3	Setting the Current Folder (Backward).....	377
5.5.4	Setting the Current Folder (Root).....	378
5.6	Pushing Objects.....	379
5.6.1	Pushing Files.....	379
5.6.2	Pushing Folders	379
5.6.2.1	Creating New Folders	380
5.7	Pulling Objects.....	381
5.7.1	Pulling Files	381
5.7.2	Pulling Folders.....	381



5.8	Manipulating Objects	381
5.8.1	Deleting Files	381
5.8.2	Deleting Folders	382
5.9	Disconnection	382
6	Service Discovery	383
6.1	SD service records	383
6.2	SDP protocol data units	384
7	References	385
7.1	Normative references	385

FOREWORD

This document, together with the Generic Object Exchange profile and the Generic Access profile form the File Transfer usage model.

Interoperability between devices from different manufacturers is provided for a specific service and usage model if the devices conform to a Bluetooth SIG-defined profile specification. A profile defines a selection of messages and procedures (generally termed *capabilities*) from the Bluetooth SIG specifications, and gives an unambiguous description of the air interface for specified service(s) and usage model(s).

All defined features are process-mandatory. This means that if a feature is used, it is used in a specified manner. Whether the provision of a feature is mandatory or optional is stated separately for both sides of the Bluetooth air interface.



1 INTRODUCTION

1.1 SCOPE

The File Transfer profile defines the requirements for the protocols and procedures that shall be used by the applications providing the File Transfer usage model. This profile uses the Generic Object Exchange profile (GOEP) as a base profile to define the interoperability requirements for the protocols needed by the applications. The most common devices using these usage models can be (but are not limited to) PCs, notebooks, and PDAs.

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

1.2 BLUETOOTH PROFILE STRUCTURE

In [Figure 1.1](#), the Bluetooth profile structure and the dependencies of the profiles are depicted. A profile is dependent upon another profile if it re-uses parts of that profile, by implicitly or explicitly referencing it. Dependency is illustrated in the figure: a profile has dependencies on the profile(s) in which it is contained – directly and indirectly.

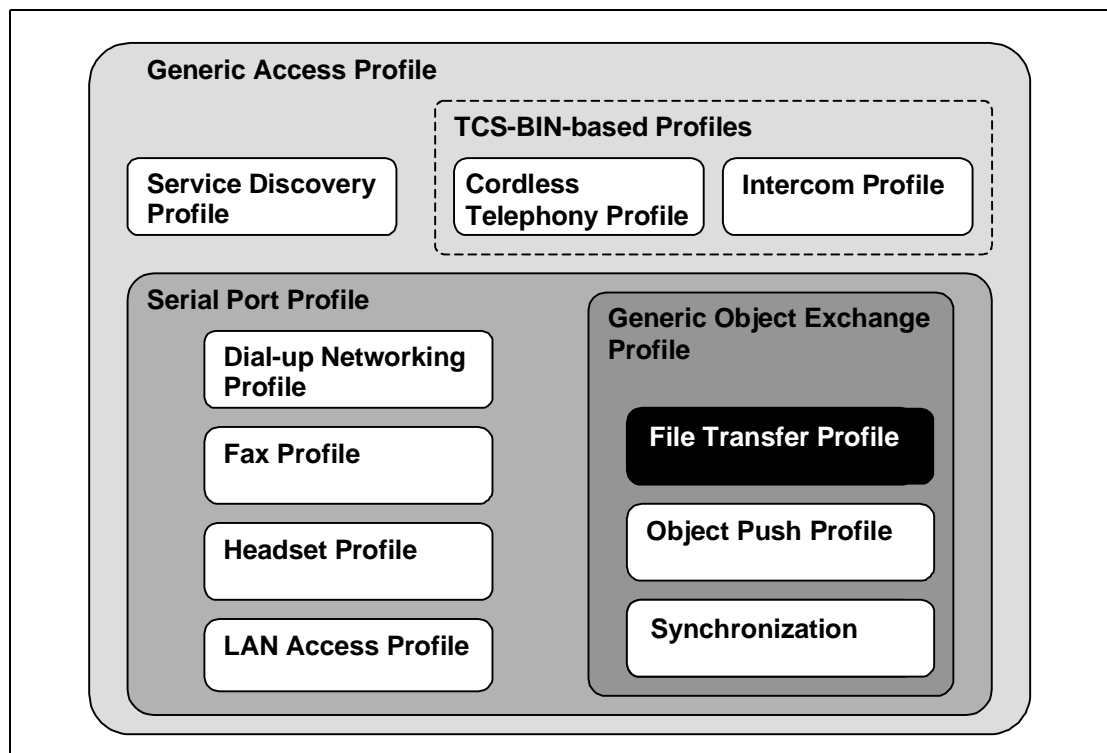


Figure 1.1: Bluetooth Profiles

1.3 BLUETOOTH OBEX-RELATED SPECIFICATIONS

Bluetooth Specification includes five separate specifications for OBEX and applications using OBEX.

1. Bluetooth IrDA Interoperability Specification [1].
 - Defines how the applications can function over both Bluetooth and IrDA.
 - Specifies how OBEX is mapped over RFCOMM and TCP.
 - Defines the application profiles using OBEX over Bluetooth.
2. Bluetooth [Generic Object Exchange Profile](#) Specification [2]
 - Generic interoperability specification for the application profiles using OBEX.
 - Defines the interoperability requirements of the lower protocol layers (e.g. Baseband and LMP) for the application profiles.
3. Bluetooth [Synchronization Profile](#) Specification [3]
 - Application Profile for Synchronization applications.
 - Defines the interoperability requirements for the applications within the Synchronization application profile.
 - Does not define the requirements for the Baseband, LMP, L2CAP, or RFCOMM.



4. Bluetooth File Transfer Profile Specification (This Specification)

- Application Profile for File Transfer applications.
- Defines the interoperability requirements for the applications within the File Transfer application profile.
- Does not define the requirements for the Baseband, LMP, L2CAP, or RFCOMM.

5. Bluetooth Object Push Profile Specification [4]

- Application Profile for Object Push applications.
- Defines the interoperability requirements for the applications within the Object Push application profile.
- Does not define the requirements for the Baseband, LMP, L2CAP, or RFCOMM.

1.4 SYMBOLS AND CONVENTIONS

1.4.1 Requirement status symbols

In this document (especially in the profile requirements tables in Annex A), the following symbols are used:

‘M’ for mandatory to support (used for capabilities that shall be used in the profile);

‘O’ for optional to support (used for capabilities that can be used in the profile);

‘C’ for conditional support (used for capabilities that shall be used in case a certain other capability is supported);

‘X’ for excluded (used for capabilities that may be supported by the unit but shall never be used in the profile);

‘N/A’ for not applicable (in the given context it is impossible to use this capability).

Some excluded capabilities are capabilities that, according to the relevant Bluetooth specification, are mandatory. These are features that may degrade operation of devices following this profile. Therefore, these features shall never be activated while a unit is operating as a unit within this profile.



1.4.2 Signaling diagram conventions

The following arrows are used in diagrams describing procedures:

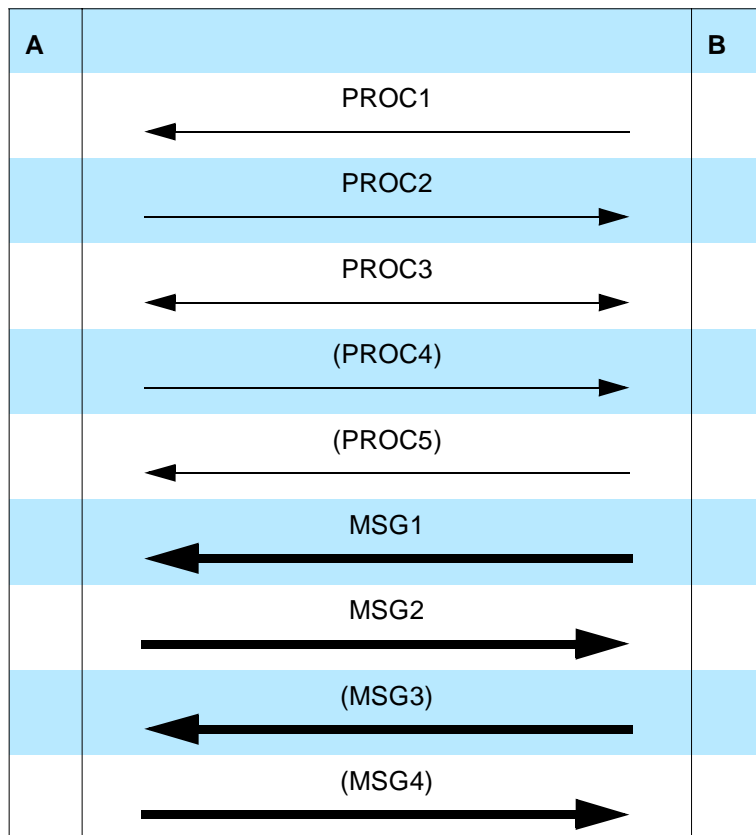


Table 1.1: Arrows used in signaling diagrams

In the table above, the following cases are shown: PROC1 is a sub-procedure initiated by B. PROC2 is a sub-procedure initiated by A. PROC3 is a sub-procedure where the initiating side is undefined (may be both A and B). PROC4 indicates an optional sub-procedure initiated by A, and PROC5 indicates an optional sub-procedure initiated by B.

MSG1 is a message sent from B to A. MSG2 is a message sent from A to B. MSG3 indicates an optional message from A to B, and MSG4 indicates an optional message from B to A.

2 PROFILE OVERVIEW

2.1 PROFILE STACK

The figure below shows the protocols and entities used in this profile.

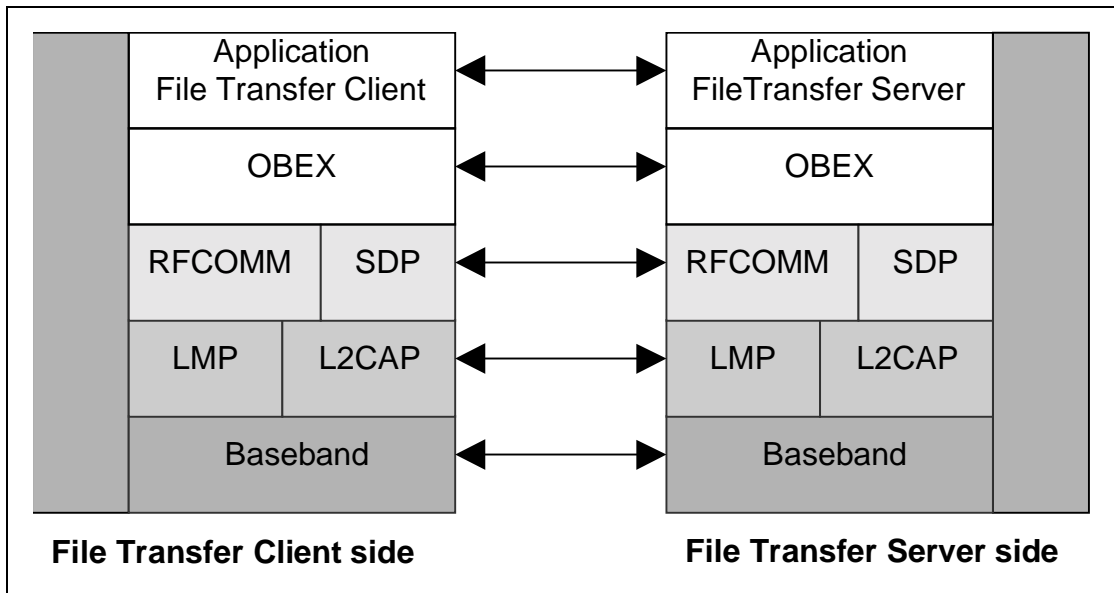


Figure 2.1: Protocol model

The Baseband [5], LMP [6] and L2CAP [7] are the OSI layer 1 and 2 Bluetooth protocols. RFCOMM [8] is the Bluetooth adaptation of GSM TS 07.10 [9]. SDP is the Bluetooth Service Discovery Protocol [10]. OBEX [1] is the Bluetooth adaptation of IrOBEX [11].

The RFCOMM, L2CAP, LMP, and Baseband interoperability requirements are defined in GOEP.

2.2 CONFIGURATIONS AND ROLES

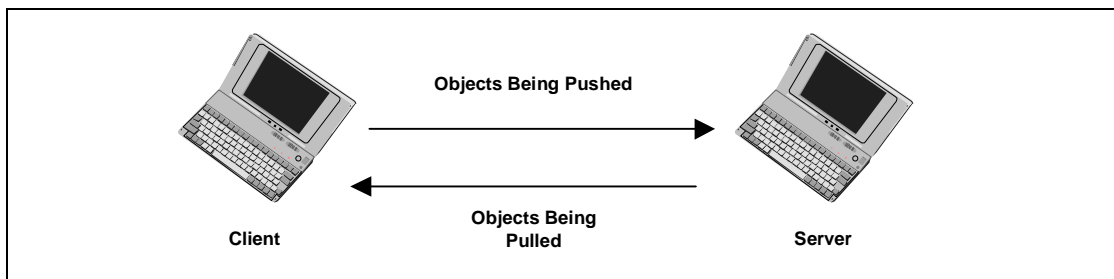


Figure 2.2: Bi-directional File Transfer Example between two Personal Computers



The following roles are defined for this profile:

Client – The Client device initiates the operation, which pushes and pulls objects to and from the *Server*. In addition to the interoperability requirements defined in this profile, the Client must also comply with the interoperability requirements for the Client of the GOEP if not defined in the contrary. The Client must be able to interpret the OBEX Folder Listing format and may display this information for the user.

Server – The Server device is the target remote Bluetooth device that provides an object exchange server and folder browsing capability using the OBEX Folder Listing format. In addition to the interoperability requirements defined in this profile, the Server must comply with the interoperability requirements for the Server of the GOEP if not defined in the contrary.

2.3 USER REQUIREMENTS AND SCENARIOS

The scenarios covered by this profile are the following:

- If file browsing is supported on the client and the server, the client is able to browse the object store of the Server. Clients are required to pull and understand Folder Listing Objects. Servers are required to respond to requests for Folder Listing Objects. Servers should have a root folder. The root folder is typically not the root directory of the file system, but a designated public folder. A server may expose different root folders based on the user or device initiating the OBEX connection. Servers are not required expose a folder hierarchy.
- Usage of the Client to transfer objects to and from the Server. The transfer of objects includes folders and files. Clients must support the ability to push or pull files from the Server. Clients are not required to push or pull folders. Servers are required to support file push, pull, or both. Servers are allowed to have read-only folders and files, which means they can restrict object pushes. Thus, Servers are not required to support folder push or pull.
- Usage of the Client to create folders and delete objects (folders and files) on the Server. Clients are not required to support folder/file deletion or folder creation. Servers are allowed to support read-only folders and files, which means they can restrict folder/file deletion and creation.

A device adhering to this profile must support Client capability, Server capability or both. The restrictions applying to this profile are the same as in the GOEP.

2.4 PROFILE FUNDAMENTALS

The profile fundamentals are the same as defined in [Section 2.4](#) in GOEP [2]. Support for link level authentication and encryption is required but their use is optional.



Support for OBEX authentication is required but its use is optional.

This profile does not mandate the server or client to enter any discoverable or connectable modes automatically, even if they are able to do so.

On the Client side, end-user intervention is typically needed to initiate file transfer (see [Chapter 3](#)).

Support of bonding is required but its use is optional.



3 USER INTERFACE ASPECTS

3.1 FILE TRANSFER MODE SELECTION, SERVERS

Servers must be placed in File Transfer mode. This mode enables a Client to initiate file transfer operations with the Server. When entering this mode, File Transfer Servers should set the device in *Limited Discoverable* mode (see [Generic Access Profile](#)), ensure that the Object Transfer Bit is set in the CoD (see [15]), and register a service record in the SDDB (see [section 6 on page 394](#)).

It is recommended that this mode be set and unset by user interaction, when possible. Public devices, devices that want to be visible at all times, or devices that can not supply a user interface to enable File Transfer mode shall use *General Discoverable* mode (see [Generic Access Profile](#)) instead of *Limited Discoverable* mode.

3.2 FUNCTION SELECTION, CLIENTS

Clients provide file transfer functions to the user via a user interface. An example of a file transfer user interface is a file-tree viewer to browse folders and files. Using such a system file-tree viewer, the user can browse and manipulate files on another PC, which appears in the network view.

File Transfer Applications may provide the following functions:

Select Server	Selecting the Server from a list of possible Servers, and setting up a connection to it.
Navigate Folders	Displaying the Server's folder hierarchy, including the files in the folders, and moving through the Server's folder hierarchy to select the current folder. The current folder is where items are pulled and/or pushed.
Pull Object	Copying a file or a folder from the Server to the Client.
Push Object	Copying a file or folder from the Client to the Server.
Delete Object	Deleting a file or folder on the Server.
Create Folder	Creating a new folder on the Server.

When the user selects the Select Server function, an inquiry procedure will be performed to produce a list of available devices in the vicinity. Requirements on inquiry procedures are discussed in [Section 6.5.1](#) of the GOEP [2].



3.3 APPLICATION USAGE

In this section, the presented scenarios work as examples. Variations in the actual implementations are possible and allowed.

When the Client wants to select a Server the following user interaction can be followed:

Client	Server
	The user sets the device into File Transfer mode . A Server typically does not need to provide any other user interaction.
The user of the Client selects the File Transfer Application on the device.	
A list of Servers that may support the File Transfer service is displayed to the user.	
The user selects a Server in which to connect. The connection may require the user to enter a password for authentication. If both link level authentication and OBEX authentication is required, then the user will need to be prompted for two passwords.	If the Client requires authentication of the Server, then the Server will need to prompt the user for a password. If both link level authentication and OBEX authentication are required, then the user will need to be prompted for two passwords.
After the connection is complete, including any authentication, the contents of the Server's root folder are displayed.	



The following user interaction shows how the user of the Client performs file transfer functions. The operations assume a Server has already been selected as described above.

Client	Server
<p>The user is presented with the folder hierarchy of the Server. The first presentation has the root folder selected as the current folder.</p>	
<p>The user chooses a folder to be the current folder. The contents of this folder are displayed.</p>	
<p>To push a file from the Client to the Server, the user selects a file on the Client and activates the Push Object function. The object is transferred to the current folder on the Server.</p>	
<p>To pull a file from the Server, the user selects a file in the current folder of the Server and activates the Pull Object function. The user is notified of the result of the operation.</p>	
<p>To delete a file on the Server, the user selects the file in the Server's current folder and activates the Delete Object function. The user is notified of the result of the operation.</p>	
<p>To create a new folder on the Server, the user activates the Create Folder function. This function requests a name from the user for the folder. When complete, a new folder is created in the Server's current folder.</p>	



4 APPLICATION LAYER

This section describes the feature requirements on units active in the File Transfer use case.

4.1 FEATURE OVERVIEW

The File Transfer application is divided into three main features, as shown in the [Table 4.1](#) below.

	Features	Support in File Transfer Client	Support in File Transfer Server
1.	Folder Browsing	M	M
2.	Object Transfer: File Transfer Folder Transfer	M O	M O*
3.	Object Manipulation	O	O*

Table 4.1: Application layer procedures

*. Optional, but the server must be able to respond with an appropriate error code, even if it doesn't support these capabilities.

4.2 FOLDER BROWSING

A folder browsing session may begin with the Client connecting to the Server and pulling the contents of the Server's root folder. When an OBEX connection is made, the Server starts out with its current folder set to the root folder. The Server may choose to expose different root folders to different users and/or devices. The Server has the right to refuse to disclose the contents of the root folder by replying to the GET folder object request with an Unauthorized or Forbidden response. If allowed, the contents of the folder must be transferred in the Folder Listing format specified in [\[11\]](#).

[Table 4.2](#) shows the application procedure required by the Client to connect to the Server and pull the contents of the root folder.



Client	Details
OBEX CONNECT.	Target Header must be set to the Folder Browsing UUID: F9EC7BC4-953C-11D2-984E-525400DC9E09. This UUID is sent in binary (16 bytes) with most significant byte sent first (0xF9 is sent first).
Pull the contents of the Server's root folder using GET.	The Type Header must be set to the MIME-type of the Folder Listing Object (x-obex/folder-listing). The Connect ID header must be set to the value returned in the Connect operation. A Name header is not used.

Table 4.2: Application layer procedure for File Transfer Connect

Browsing an object store involves displaying folder contents and setting the 'current folder'. The OBEX SETPATH command is used to set the current folder. A Server must support the SETPATH command to the root folder (default directory). To display a folder hierarchy starting with the root folder, the Client must read the root folder contents using GET. It must then retrieve the contents of all sub-folders using GET. If the sub-folders contain folders, then the Client must retrieve the contents of these folders and so on. To retrieve the contents of a folder, the Client must set the current folder to the sub-folder using SETPATH, then pull the sub-folder contents using GET. Table 4.3 shows the application procedure required for retrieving the contents of a sub-folder.

Client	Details
Set the current folder to the sub-folder using OBEX SETPATH.	Name header is set to the name of the sub-folder. Connect ID header is required.
Pull the contents of the sub-folder using GET.	No Name is sent, since the sub-folder is the current folder. The Type Header must be set to the MIME-type of the Folder Listing Object (x-obex/folder-listing). Connect ID header is required.
Set the current folder back to the root folder using OBEX SETPATH.	Name header is empty. Connect ID header is required.
If the parent of the sub-folder is not the root folder, then set the current folder to the parent folder using SETPATH.	The Backup flag is set and no Name header is sent. Connect ID header is required.

Table 4.3: Application layer procedure for Folder Browsing



4.3 OBJECT TRANSFER

Objects are transferred from the Client to the Server using OBEX PUT, and objects are transferred from the Server to the Client using OBEX GET. Transferring files requires a single PUT or GET operation per file. Successful transfer of a file does not necessarily imply that file can be immediately retrieved due to the protection policies enforced by the Server. Transferring folders requires transferring all the items stored in a folder, including other folders. The process of transferring a folder may require that new folders be created. The SETPATH command is used to create folders.

Table 4.4 shows the application procedure for transferring a folder from the Client to the Server. If the folder contains other folders, then these other folders are transferred using the same method. The folder is transferred to the current folder on the Server.

Client	Details
Create a new folder (if it does not already exist) in the Server's current folder using SETPATH. The current folder is changed to this new folder.	Name header is set to the name of the new folder. Connect ID header is required.
Push all files to the new folder using a PUT command for each file.	The Name header is set to the name of the file. Connect ID header is required.
Folders are created using SETPATH.	Name header is set to folder name. This application procedure is applied recursively to each folder. Connect ID header is required.
Set the current folder back to the parent folder using SETPATH.	The Backup flag is set and no Name header is sent. Connect ID header is required.

Table 4.4: Application layer procedure for Pushing a Folder

Table 4.5 shows the application procedure for transferring a folder from the Server to the Client.

Client	Details
Set the current folder to the folder which is to be transferred using SETPATH.	The Name header is set to the name of the folder. Connect ID header is required.
Pull the contents of the folder using GET.	A Name header is not sent, and the Type Header must be set to the MIME-type of the Folder Listing Object (x-obex/folder-listing).
Pull all files to the new folder using a GET command for each file.	The Name header is set to the name of the file. Connect ID header is required.
Pull all Folders to the new folder using this application procedure.	This application procedure is applied recursively to each folder.

Table 4.5: Application layer procedure for Pulling a Folder



Set the current folder back to the parent folder, using SETPATH.	The Backup flag is set and no Name header is sent. Connect ID header is required.
--	---

Table 4.5: Application layer procedure for Pulling a Folder

4.4 OBJECT MANIPULATION

A Client can create and delete folders and files on a Server for which it has proper access privileges. A brief summary of these functions is shown below.

- A file is deleted by using a PUT command with the name of the file in a Name header and no Body header.
- An empty folder is deleted by using a PUT command with the name of the folder in a Name header and no Body header.
- A non-empty folder can be deleted in the same way as an empty folder but Servers may not allow this operation. If a Server refuses to delete a non-empty folder it must return the “Precondition Failed” (0xCC) response code. This response code tells the Client that it must first delete all the elements of the folder individually before deleting the folder.
- A new folder is created in the Server’s current folder by using the SETPATH command with the name of the folder in a Name header. If a folder with that name already exists, then a new folder is not created. In both cases the current folder is set to the new folder.

5 OBEX

5.1 OBEX OPERATIONS USED

Table 5.1 shows the OBEX operations that are used in the File Transfer profile.

Operation no.	OBEX Operation	Client	Server
1	Connect	M	M
2	Disconnect	M	M
3	Put	M	M
4	Get	M	M
5	Abort	M	M
6	SetPath	M	M

Table 5.1: OBEX Operations



5.2 OBEX HEADERS

Table 5.2 shows the specified OBEX headers that are used in the File Transfer profile.

Header no.	OBEX Headers	Client	Server
1	Count	O	O
2	Name	M	M
3	Type	M	M
4	Length	M	M
5	Time	O	O
6	Description	O	O
7	Target	M	M
8	HTTP	O	O
9	Body	M	M
10	End of Body	M	M
11	Who	M	M
12	Connection ID	M	M
13	Authenticate Challenge	M	M
14	Authenticate Response	M	M
15	Application Parameters	X	X
16	Object Class	X	X

Table 5.2: OBEX Headers

5.3 INITIALIZATION OF OBEX

Devices implementing the File Transfer profile can optionally use OBEX authentication. The initialization procedure is defined in Section 5.3 of GOEP [2].

5.4 ESTABLISHMENT OF OBEX SESSION

The OBEX connection must use a Target header set to the File Browsing UUID, F9EC7BC4-953C-11D2-984E-525400DC9E09. This UUID is sent in binary (16 bytes) with 0xF9 sent first. OBEX authentication can optionally be used. This profile follows the procedures described in Section 5.4 of GOEP [2] with the Target, Connection ID, and Who headers being mandatory.



5.5 BROWSING FOLDERS

Browsing folders involves pulling Folder Listing objects and setting the current folder. Navigating a folder hierarchy requires moving forward and backward by changing the current folder. Upon completion of the OBEX Connect operation the Server’s current folder is the root folder. As noted previously, different root folders may be exported based on the Client device and/or user.

5.5.1 Pulling a Folder Listing Object

Pulling a Folder Listing object uses a GET operation and follows the procedure described in Section 5.6 of GOEP [2]. The Connection ID and Type headers are mandatory. A Name header containing the name of the folder is used to pull the listing of a folder. Sending the GET command without a name header is used to pull the contents of the current folder. Typically, a folder browsing application will pull the contents of the current folder, so a Name header is not used. The Type header must be set to ‘x-obex/folder-listing’.

5.5.2 Setting the Current Folder (Forward)

Setting the current folder requires the SETPATH operation. The SETPATH request must include the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for SETPATH	0x85	M	-
Field	Packet Length	Varies	M	-
Field	Flags	0x02	M	‘Backup level’ flag is set to 0 and ‘Don’t Create’ flag is set to 1.
Field	Constants	0x00	M	Constants are not used, and the field must be set to 0.
Header	Connection ID	Varies	M	Connection ID is set to the value returned by the Server during the OBEX Connect operation. This must be the first header.
Header	Name	Varies	M	Name of the folder.

Table 5.3: Fields and Headers in SETPATH Request for Setting Current Folder (Forward)



The response packet for the SETPATH request has the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for SETPATH	0xA0 or 0xC4	M	0xA0 for success or 0xC4 if the folder does not exist or 0xC1 if folder browsing is not permitted.
Field	Packet Length	Varies	M	-

Table 5.4: Fields and Headers in SETPATH Response for Setting Current Folder (Forward)

Other headers such as Description can optionally be used.

5.5.3 Setting the Current Folder (Backward)

Setting the current folder back to the parent folder requires the SETPATH operation. The SETPATH request must include the following fields and headers (note that a Name header is not used):

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for SET-PATH	0x85	M	-
Field	Packet Length	Varies	M	-
Field	Flags	0x03	M	'Backup level' flag is set to 1 and 'Don't Create' flag is set to 1.
Field	Constants	0x00	M	Constants are not used, and the field must be set to 0.
Header	Connection ID	Varies	M	Connection ID is set to the value returned by the Server during the OBEX Connect operation. This must be the first header.

Table 5.5: Fields and Headers in SETPATH Request for Setting Current Folder (Backward)



The response packet for the SETPATH request has the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for SETPATH	0xA0 or 0xC4	M	0xA0 for success, or 0xC4 if the current folder is the root.
Field	Packet Length	Varies	M	-

Table 5.6: Fields and Headers in SETPATH Response for Setting Current Folder (Backward)

Other headers, such as Description, can optionally be used.

5.5.4 Setting the Current Folder (Root)

Setting the current folder to the root requires the SETPATH operation. The SETPATH request must include the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for SET-PATH	0x85	M	-
Field	Packet Length	Varies	M	-
Field	Flags	0x02	M	'Backup level' flag is set to 0 and 'Don't Create' flag is set to 1.
Field	Constants	0x00	M	Constants are not used, and the field must be set to 0.
Header	Connection ID	Varies	M	Connection ID is set to the value returned by the Server during the OBEX Connect operation. This must be the first header.
Header	Name	Empty	M	Name header is empty.

Table 5.7: Fields and Headers in SETPATH Request for Setting Current Folder (Root)



The response packet for the SETPATH request has the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for SETPATH	0xA0	M	0xA0 for success.
Field	Packet Length	Varies	M	-

Table 5.8: Fields and Headers in SETPATH Response for Setting Current Folder (Root)

Other headers, such as Description, can optionally be used.

5.6 PUSHING OBJECTS

Pushing object involves pushing files and folders.

5.6.1 Pushing Files

Pushing files follows the procedure described in [Section 5.5](#) of GOEP [2]. The Connection ID header is mandatory.

5.6.2 Pushing Folders

Pushing folders involves creating new folders and pushing files. It may also involve navigating through the folder hierarchy. Navigation is described in [Section 5.5 on page 387](#). Pushing files is described in [Section 5.6.1 on page 390](#).



5.6.2.1 Creating New Folders

Creating a new folder requires the SETPATH operation. The SETPATH request must include the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for SET-PATH	0x85	M	-
Field	Packet Length	Varies	M	-
Field	Flags	0x00	M	'Backup level' flag is set to 0 and 'Don't Create' flag is set to 0.
Field	Constants	0x00	M	Constants are not used, and the field must be set to 0.
Header	Connection ID	Varies	M	Connection ID is set to the value returned by the Server during the OBEX Connect operation. This must be the first header.
Header	Name	Varies	M	Name of the folder.

Table 5.9: Fields and Headers in SETPATH Request for Creating a Folder.

The response packet for the SETPATH request has the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for SETPATH	0xA0 or 0xC1	M	0xA0 for success or 0xC1 if the current folder is read only and creation of a new folder is unauthorized.
Field	Packet Length	Varies	M	-

Table 5.10: Fields and Headers in SETPATH Response for Creating a Folder

Other headers such as Description can optionally be used.



5.7 PULLING OBJECTS

Pulling objects involves pulling files and folders.

5.7.1 Pulling Files

Pulling files follows the procedure described in [Section 5.6](#) of GOEP [2]. The Connect ID header is mandatory.

5.7.2 Pulling Folders

Pulling folders involves navigating the folder hierarchy, pulling folder listing objects and pulling files. Navigating the folder hierarchy and pulling folder listing-objects is described in [Section 5.5 on page 387](#). Pulling files is described in [Section 5.7.1 on page 392](#).

5.8 MANIPULATING OBJECTS

Manipulating objects includes deleting objects and creating new folders. Creating new folders is described in [Section 5.6.2.1 on page 391](#), Creating New Folders. Deleting objects involves deleting files and folders.

5.8.1 Deleting Files

Deleting a file requires the PUT operation. The PUT request must include the following fields and headers (note that no Body or End Body headers are sent):

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for PUT	0x82	M	-
Field	Packet Length	Varies	M	-
Header	ConnectionID	Varies	M	Connection ID is set to the value returned by the Server during the OBEX Connect operation. This must be the first header.
Header	Name	Varies	M	The header value is the name of the object to delete.

Table 5.11: Fields and Headers in PUT Request for Delete



The response packet for the PUT request has the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for PUT	0xA0, 0xC1 or 0xC4	M	0xA0 for success, 0xC1 for unauthorized (e.g. read only) or 0xC4 if the file does not exist.
Field	Packet Length	Varies	M	-

Table 5.12: Fields and Headers in PUT Response for Delete

Other headers such as Description can optionally be used.

5.8.2 Deleting Folders

A folder can be deleted using the same procedure used to delete a file (see [Section 5.8.1 on page 392](#)). Deleting a non-empty folder will delete all its contents, including other folders. Some Servers may not allow this operation and will return the “Precondition Failed” (0xCC) response code, indicating that the folder is not empty. In this case the Client will need to delete the contents before deleting the folder.

5.9 DISCONNECTION

See [Section 5.7](#) in GOEP [2].



6 SERVICE DISCOVERY

6.1 SD SERVICE RECORDS

The service belonging to the File Transfer profile is a server, which enables bi-directional generic file transfer. OBEX is used as a session protocol for this service. The following service records must be put into the SDDB.

Item	Definition:	Type/ Size:	Value:*	AttrID	Status	Default Value
Service Class ID List				See [15]	M	
Service Class #0		UUID	OBEX-File Transfer		M	
Protocol Descriptor list				See [15]	M	
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	CHANNEL	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	
Service name	Displayable Text name	String	Varies	See [15]	O	“OBEX File Transfer”
BluetoothProfileDescriptorList				See [15]	O	
Profile ID #0	Supported profile	UUID	OBEX File-Transfer			OBEX File Transfer [15]
Param #0	Profile version	uint16	0x100			0x100

Table 6.1: File Transfer Service Record

* UUID values are defined in the *Assigned Numbers* document.

6.2 SDP PROTOCOL DATA UNITS

Table 19 shows the specified SDP PDUs (Protocol Data Units) which are required in the File Transfer profile.

PDU no.	SDP PDU	Server	Client
1	SdpErrorResponse	M	M
2	SdpServiceSearch AttributeRequest	M	M
3	SdpServiceSearch AttributeResponse	M	M

Table 6.2: SDP PDUs Minimal Requirements

7 REFERENCES

7.1 NORMATIVE REFERENCES

- [1] Bluetooth Special Interest Group, IrDA Interoperability
- [2] Bluetooth Special Interest Group, Generic Object Exchange Profile
- [3] Bluetooth Special Interest Group, Synchronization Profile
- [4] Bluetooth Special Interest Group, Object Push Profile
- [5] Bluetooth Special Interest Group, Baseband Specification
- [6] Bluetooth Special Interest Group, LMP Specification
- [7] Bluetooth Special Interest Group, L2CAP Specification
- [8] Bluetooth Special Interest Group, RFCOMM with TS 07.10
- [9] ETSI, TS 07.10, Version 6.3.0
- [10] Bluetooth Special Interest Group, SDP Specification
- [11] Infrared Data Association, IrDA Object Exchange Protocol (IrOBEX) with Published Errata, Version 1.2, April 1999.
- [12] Infrared Data Association, IrMC (Ir Mobile Communications) Specification with Published Errata, Version 1.1, February 1999.
- [13] The Internet Mail Consortium, vCard – The Electronic Business Card Exchange Format, Version 2.1, September 1996.
- [14] The Internet Mail Consortium, vCalendar – The Electronic Calendaring and Scheduling Exchange Format, Version 1.0, September 1996.
- [15] Bluetooth Special Interest Group, Assigned Numbers specification
<http://www.bluetooth.org/assigned-numbers.htm>
- [16] Bluetooth Special Interest Group, Bluetooth Generic Access Profile Specification