BLUETOOTH® DOC	Date / Year-Month-Day 2010-08-26	Approved Adopted	Revision V20r00	Document No GOEP_SPEC
Prepared By OBEX WG	E-mail Address OBEX-feedback@bluetooth.org			N.B.

GENERIC OBJECT EXCHANGE PROFILE

Abstract:

This profile defines the requirements for *Bluetooth*® devices necessary for the support of the object exchange usage models. The requirements are expressed by defining the features and procedures that are required for interoperability between Bluetooth devices in the object exchange usage models.

Disclaimer and Copyright Notice

The copyright in this specification is owned by the Promoter Members of *Bluetooth*® Special Interest Group (SIG), Inc. ("*Bluetooth* SIG"). Use of these specifications and any related intellectual property (collectively, the "Specification"), is governed by the Promoters Membership Agreement among the Promoter Members and *Bluetooth* SIG (the "Promoters Agreement"), certain membership agreements between *Bluetooth* SIG and its Adopter and Associate Members (the "Membership Agreements") and the *Bluetooth* Specification Early Adopters Agreements (1.2 Early Adopters Agreements) among Early Adopter members of the unincorporated *Bluetooth* SIG and the Promoter Members (the "Early Adopters Agreement"). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to *Bluetooth* SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of *Bluetooth* SIG or a party to an Early Adopters Agreement (each such person or party, a "Member"), is prohibited. The legal rights and obligations of each Member are governed by their applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to *Bluetooth* SIG or any of its members for patent, copyright and/or trademark infringement.

THE SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the *Bluetooth* technology ("*Bluetooth* products") may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of *Bluetooth* products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their *Bluetooth* Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their *Bluetooth* products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses**. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.**

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST *BLUETOOTH* SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserve the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Revision History

Revision	Date	Comments
Bluetooth V1.1	22 February 2001	The GOEP Profile Specification as released with the adoption of Bluetooth Specification V1.1.
D20r00	15 Au 2005	Reformatted document and converted FrameMaker document into Microsoft Word source document.
D20r01	13 September 2005	Editorial updates
D20r02	31 October 2005	Editorial updates
D20r03	15 November 2005	Editorial updates
D20r04	30 November 2005	Editorial updates
D20r05	20 March 2006	Input reviewer's comments for 1.2
D20r06	01 Oct 2007	Input TH's comments for 2.1; Updated CR
D20r07	04 May 2010	Input comments from CRs; editorial changes
D20r08	19 May 2010	Fix ups of Draft specs after integration by Tech Ed, include KMH review
D20r09	18 June 2010	Include BARB review comments, reset version number for publication as v2.0
V20r00	26 August 2010	Adopted by the Bluetooth SIG Board of Directors

Contents

1 Introduction 7
1.1 Scope 7
1.2 Bluetooth Profile Structure 7
1.3 Bluetooth OBEX-Related Specifications 7
1.4 Symbols and Conventions 8
1.4.1 Requirement Status Symbols 8
1.4.2 Signaling Diagram Conventions 9
2 Profile Overview 10
2.1 Profile Stack 10
2.2 Configurations and Roles 10
2.3 User Requirements and Scenarios 10
2.4 Profile Fundamentals 11
3 User Interface Aspects 12
4 Application Layer 13
4.1 Feature Overview 13
4.2 Establishing an Object Exchange Connection 13
4.3 Pushing a Data Object 13
4.4 Pulling a Data Object 13
4.5 Performing an Action on a Data Object 13
4.6 Using Single Response Mode 14
4.6.1 SRMP Use Cases 15
4.6.1.1 Setup Headers Use Case 15
4.6.1.2 User Accept Use Case 16
4.6.2 SRMP Usage Requirements 17
4.7 Creating a Reliable Object Exchange Session 18
4.8 Using Reliable Object Exchange Sessions 19
5 OBEX Interoperability Requirements 21
5.1 OBEX Operations Used 21
5.2 OBEX Headers 21
5.3 Initialization of OBEX 21
5.4 Establishment of OBEX Connection 22
5.4.1 OBEX Connection without Authentication 22
5.4.2 OBEX Connection with Authentication 24
5.5 Pushing Data to Server 30
5.5.1 Pushing Data over an OBEX Connection 30
5.5.2 Deleting an Object over an OBEX Connection 31
5.6 Pulling Data from Server 31
5.6.1 Pulling Data over an OBEX Connection 32
5.7 Performing an Action on a Server Object 33
5.8 Setting up a Reliable Session with the Server 34
5.9 Suspending a Reliable Session with the Server 35
5.10 Resuming a Reliable Session with the Server 36
5.11 Closing a Reliable Session with the Server 37
5.12 Disconnection 37
6 SDP Interoperability Requirements 39
6.1 Requirements for Application Profiles which do not require backwards compatibility 39
6.2 Requirements for Application Profiles which require backwards compatibility with GOEP v1.x 39
7 Generic Access Profile Requirements 41
7.1 L2CAP Interoperability Requirements 41
7.1.1 MTU option 41
7.1.2 Flow and Error Control Option 41
7.1.3 Frame Checksum (FCS) Option 42
7.2 Link Control (LC) Interoperability Requirements 42
7.2.1 Inquiry and Inquiry Scan 42

8 Generic Access Profile Interoperability Requirements 43
8.1 Modes 43
8.2 Security Aspects 43
8.3 Idle Mode Procedures 44
8.3.1 Bonding 44
9 References 45
9.1 Normative References 45

Foreword

The purpose of this document is to work as a generic profile document for all application profiles using the OBEX protocol.

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 Generic Object Exchange profile defines the protocols and procedures that shall be used by the applications providing the usage models which need the object exchange capabilities. The usage model can be, for example, Synchronization, File Transfer, or Object Push model. The most common devices using these usage models can be notebook PCs, PDAs, smart phones, and mobile phones.

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. For example, the Object Push profile is dependent on Generic Object Exchange, and Generic Access profiles.

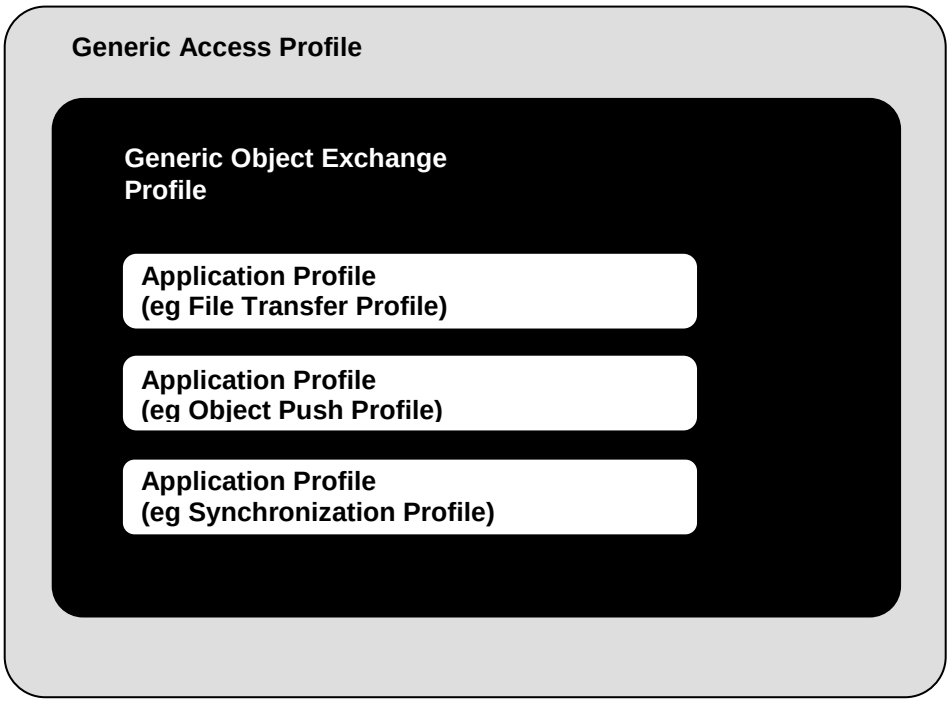

Figure 1.1: Bluetooth Profiles

1.3 Bluetooth OBEX-Related Specifications

Bluetooth Specification includes a number of separate specifications for OBEX and applications using it.

1. Bluetooth IrDA Interoperability Specification [1]

- Defines how the applications can function over both Bluetooth and IrDA.
- Specifies how OBEX is mapped over L2CAP and TCP.
- Defines the application profiles using OBEX over Bluetooth.

2. Bluetooth Generic Object Exchange Profile Specification (This specification)
   - 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. Application Profiles
   - Define the interoperability requirements for applications using OBEX.
   - Does not define the requirements for the Baseband, LMP or L2CAP.

1.4 Symbols and Conventions

1.4.1 Requirement Status Symbols

In this document, 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:

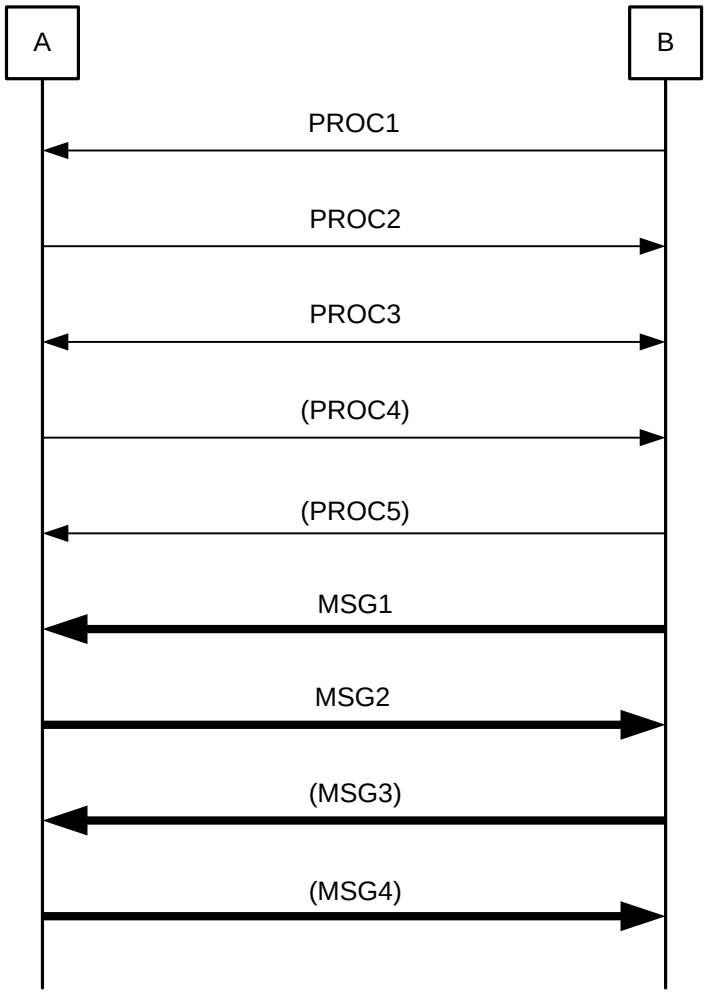

Figure 1.2: Arrows Used in Signaling Diagrams

In Figure 1.2, 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

Figure 2.1 shows the protocols and entities used in this profile.

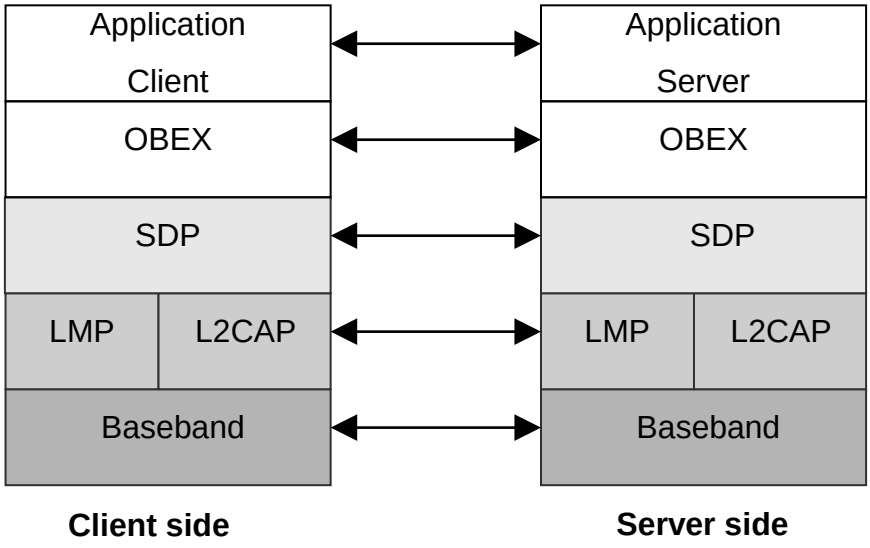

Figure 2.1: Protocol Model

The Baseband, LMP, and L2CAP are the OSI layer 1 and 2 Bluetooth protocols [2]. SDP is the Bluetooth Service Discovery Protocol [2]. OBEX [1] is the Bluetooth adaptation of IrOBEX [5].

The Application Client layer shown in Figure 2.1 is the entity sending and retrieving an object from the Server using the OBEX operations. The application Server is the data storage to and from which the data object can be sent or retrieved.

2.2 Configurations and Roles

The following roles are defined for this profile:

Server – This is the device that provides an object exchange server to and from which data objects can be pushed and pulled, respectively.

Client – This is the device that can push or/and pull data object(s) to and from the Server.

2.3 User Requirements and Scenarios

The scenarios covered by this profile are the following:

- Usage of a Server by a Client to push data object(s)
- Usage of a Server by a Client to pull data object(s)

The following restrictions apply to this profile:

4. For the device containing the Server, it is assumed that the user may have to put it into the discoverable and connectable modes when the inquiry and link establishment procedures, respectively, are processed in the Client (see Generic Access Profile).
5. The profile only supports point-to-point configurations. As a result, the Server is assumed to offer services only for one Client at a time. However, the implementation may offer a possibility for multiple Clients at a time but this is not a requirement.

2.4 Profile Fundamentals

The profile fundamentals, with which all application profiles must comply, are the following:

1. Before a Server is used with a Client for the first time, a bonding procedure including the pairing may be performed (see Section 8.3.1). This procedure must be supported, but its usage is dependent on the application profiles. The bonding typically involves manually activating bonding support and following a pairing procedure as defined in GAP see Section 8.3.1) on the keyboards of the Client and Server devices. This procedure may have to be repeated under certain circumstances; for example, if a common link key (as a bonding result) is removed on the device involved in the object exchange.
2. In addition to the link level bonding, an OBEX initialization procedure may be performed (see Section 5.3) before the Client can use the Server for the first time. The application profiles using GOEP must specify whether this procedure must be supported to provide the required security level.
3. Security can be provided by authenticating the other party upon connection establishment, and by encrypting all user data on the link level. The authentication and encryption must be supported by the devices; but whether they are used depends on the application profile using GOEP.
4. Link and channel establishments must be done according to the procedures defined in GAP (see [2]). Link and channel establishment procedures in addition to the procedures in GAP must not be defined by the application profiles using GOEP.
5. There are no fixed master/slave roles.
6. This profile does not require any lower power mode to be used.

3 User Interface Aspects

User interface aspects are not defined in this profile. They are instead defined in the application profiles, where necessary.

4 Application Layer

This section describes the service capabilities which can be utilized by the application profiles using GOEP.

4.1 Feature Overview

Table 4.1 shows the features which the Generic Object Exchange profile provides for the application profiles. The use of other features (e.g. setting the current directory) must be defined by the applications profiles needing them.

Feature no.	Feature
1	Establishing an Object Exchange connection
2	Pushing a data object
3	Pulling a data object
4	Performing an Action on a data object
5	Creating and Managing a Reliable Object Exchange Session

Table 4.1: Features Provided by GOEP

4.2 Establishing an Object Exchange Connection

This feature is used to establish the object exchange connection between the Client and Server. Before a connection is established, payload data cannot be exchanged between the Client and the Server. The use of the OBEX operations for establishing an OBEX connection is described in Section 5.4.

4.3 Pushing a Data Object

If data needs to be transferred from the Client to the Server, then this feature is used. The use of the OBEX operations for pushing the data object(s) is described in Section 5.5.

4.4 Pulling a Data Object

If data need to be transferred from the Server to the Client, then this feature is used. The use of the OBEX operations for pulling the data object(s) is described in Section 5.6.

4.5 Performing an Action on a Data Object

If an Action (copy, move/rename, or set permissions) needs to be performed on a Server object, then this feature is used. The use of the OBEX operation to perform an action on a data object is described in Section 5.12.

4.6 Using Single Response Mode

To utilize Single Response Mode (SRM), the Client shall enable this mode as described in Sections 5.5 and 5.6. SRM is deemed enabled when the Client has issued an enable request and the Server has replied with an enable response. The Server cannot issue an enable request, but can only issue a response to an enable request from the Client. SRM will remain in effect for the duration of the operation that enabled it (PUT or GET), at which time it will automatically be disabled and revert back to the normal GOEP request/response model. SRM headers shall not be sent in CONNECT request or response packets. SRM shall not be disabled during any operation. PUT or GET operations utilizing SRM shall send all non-Body headers first (unless specifically overridden by the application profile), to ensure that proper reliability can be guaranteed when used in combination with OBEX Reliable Sessions.

GET operations with SRM enabled shall not send a GET response until the GET request + Final Bit has been issued. Until this point is reached, the Client's GET request phase is ongoing and multiple unacknowledged GET requests may be issued. Once the GET request + Final bit has been issued, the GET request phase is complete and no further GET requests shall be issued (unless triggered by a Single Response Mode Parameters (SRMP) header). After the GET request phase is complete, the Server's GET responses shall begin.

The following diagram shows a normal GET request phase:

Client Requests	SRM Status	Server Responses
GET (without Final Bit) SRM header set to "enable"	**SRM Disabled**	
		Continue (with Final Bit) SRM header set to "enable"
	SRM Enabled	
GET (without Final Bit)		
GET (without Final Bit)		
GET (with Final Bit)	**GET Request phase complete** Body data may occur	
		Continue (with Final Bit) Body data may be included
		Continue (with Final Bit) Body data may be included
		…
		Success (with Final Bit) Body data may be included
Operation complete	**SRM Disabled**	

The Single Response Mode header shall be used to enable this mode, while the SRMP header may be used to alter the normal processing of SRM for a single request/response exchange during the current operation. The use of the SRMP header

is optional; however, GOEP devices that support SRM shall be able to receive and process this header. SRMP headers should be used judiciously as this will impact the overall throughput of the operation using SRM. Receipt of invalid or unexpected Single Response Mode or SRMP header values shall be ignored by the receiving device.

4.6.1 SRMP Use Cases

The SRMP header shall support the following two use cases. Both cases utilize the SRMP "wait" option (0x01). At this time no valid use cases exist for the SRMP "additional request" (0x00) and "additional request + wait" (0x02) options, therefore they shall not be used.

Each application profile should decide how SRMP headers will be used for their specific uses of GOEP. Application profiles may exclude the SRMP feature altogether if no specific advantage will be gained in using the following use cases within that profile.

4.6.1.1 Setup Headers Use Case

This use case points out specific behavior for both the Client and Server, where issuing the SRMP "wait" is useful when exchanging "setup" headers prior to sending object data during PUT or GET operations.

In this case the SRMP header can be used to facilitate additional header exchange, beyond the first request/response packet exchange used to enable SRM.

Some potential examples of "setup" headers could include:

- A header indicating the ordering of the items returned in the object.
- The maximum number of items to be returned in the object.
- The character set for the returned object,
- Other headers describing the formatting requirements of the object to be returned. The object can be an image, print document, folder listing, phonebook, message, or a generic file type, so the possible headers can span a very wide range.

For example, during a PUT operation a Server may require multiple packets of setup headers in its responses. In this case a Server would issue the SRMP "wait" as described in Section 4.6.2.

During the initiation of a GET operation, the Client and/or Server may require time to process the setup headers sent by the remote device. In the case a Client and/or Server requires extra time, it would issue the SRMP "wait" as described in section 4.6.2.

The examples below describe how the SRMP "wait" header may be used in a GET operation. If the SRMP "wait" header is used in a PUT operation, see example 1 in Section 4.6.1.2

Client Requests	SRM Status	Server Responses
GET (without Final Bit) SRM header set to "enable"	**SRM Disabled**	
		Continue (with Final Bit) SRM header set to "enable"

Client Requests	SRM Status	Server Responses
		SRMP header set to “wait”
GET (without Final Bit)	SRM Enabled but waiting	
		Continue (with Final Bit)
GET (without Final Bit)	**SRM Enabled**	
GET (without Final Bit)		
GET (with Final Bit)	**Body data may occur**	
		Continue (with Final Bit) Body data may be included
		Continue (with Final Bit) Body data may be included
		Success (with Final Bit) Body data may be included

Table 4.2: Example 1: SRMP used by a GOEP Server during the “GET request” phase

Client Requests	SRM Status	Server Responses
GET (with Final Bit) SRM header set to “enable” SRMP header set to “wait”	SRM Disabled Body data may occur	
		Continue (with Final Bit) SRM header set to “enable” Body data may be included
GET (with Final Bit) SRMP header set to “wait”	SRM Enabled but waiting	
		Continue (with Final Bit) Body data may be included
GET (with Final Bit)	SRM Enabled	
		Continue (with Final Bit) Body data may be included
		Continue (with Final Bit) Body data may be included
		Continue (with Final Bit) Body data may be included
		…
		Success (with Final Bit) Body data may be included

Table 4.3: Example 2: SRMP used by a GOEP Client

4.6.1.2 User Accept Use Case

This use case points out specific behavior for the Server, where issuing the SRMP “wait” is useful when requiring “user acceptance” of an object prior to receiving object

data during a PUT operation. In this case the SRMP “wait” can be used as described in section 4.6.2.

See the example in Table 4.4 to describe how the SRMP “wait” header may be used in a PUT operation:

Client Requests	SRM Status	Server Responses
PUT (without Final Bit) SRM header set to “enable”	**SRM Disabled**	
		Continue (with Final Bit) SRM header set to “enable” SRMP header to set “wait”
PUT (without Final Bit)	SRM Enabled but waiting	
		Continue (with Final Bit) SRMP header set to “wait”
PUT (without Final Bit)	SRM Enabled but waiting	
		Continue (with Final Bit)
PUT (without Final Bit)	**SRM Enabled**	
PUT (without Final Bit)		
PUT (without Final Bit)		
…		
PUT (with Final Bit		
		Success (with Final Bit)

Table 4.4: Example 1: SRMP used by a GOEP Server

4.6.2 SRMP Usage Requirements

This section outlines the usage of the SRMP header within PUT and GET operations to fulfill the use cases above.

GET operation using SRMP Wait:

If the SRMP header is used, either the Client or Server Case below shall be used, but both cases are allowed to occur simultaneously.

- Client Case

  Client instructs the Server to wait for the next request packet during a GET operation. This case will utilize the **0x01** option for the SRMP header, and may be issued by the Client during a GET request to instruct the Server to wait, after sending its response packet, until the next GET request is received from the Client. Once the Client issues a GET request including the Final Bit, no further GET requests shall occur during SRM mode, unless the SRMP header is used during this request packet.

  If used, the SRMP header shall be issued in the first GET request, and may be used in consecutive GET request packets to cause the Server to continue its wait;

however, once the SRMP header is not issued in a GET request, the SRMP header shall not be used in another GET request for the duration of the operation.

- Server Case

  Server instructs the Client to wait for the next response packet during a GET operation. This case will utilize the **0x01** option for the SRMP header, and may be issued by the Server during a GET response to instruct the Client to wait, after sending its request packet, until the next GET response is received from the Server. This case is useful during the "GET request" phase when a Client may be issuing multiple unacknowledged GET request packets, but may be used after the client has sent a GET request with the Final Bit set.

  If used, the SRMP header shall be issued 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 SRMP header is not issued in a GET response, the SRMP header shall not be used in another GET response for the duration of the operation.

PUT operation using SRMP Wait:

- Client Case:

  SRMP headers shall not be used in PUT requests.

- Server Case:

  Server instructs the Client to wait for the next response packet during a PUT operation. This case will utilize the 0x01 option for the SRMP header, and may be issued by the Server during a PUT response to instruct the Client to wait, after sending its request packet, until the next PUT response is received from the Server. Once the Server issues a PUT response including a SRM header confirming the enabling of SRM, no additional PUT responses shall occur during SRM mode, unless the SRMP header is used during this response packet.

  If used, the SRMP header shall be issued 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 SRMP header is not issued in a PUT response, the SRMP header shall not be used in another PUT response for the duration of the operation.

4.7 Creating a Reliable Object Exchange Session

A Reliable OBEX Session shall be used to provide reliability and the ability to resume an interrupted operation for all operations performed during the Reliable Session. When used, a Reliable OBEX Session shall be established prior to the Object Exchange Connection establishment. This is done to allow the Reliable OBEX Session to track and store the applicable OBEX connection settings. Reliable Sessions exist outside the scope of an OBEX connection, therefore the Session commands do not use the Connection Identifier header.

A Reliable OBEX Session guarantees that OBEX operations can be completed, even when interrupted by an unexpected transport disconnection. In the case of a transport disconnection, the Reliable OBEX Session will be suspended, but may be resumed

after the transport connection is reestablished. Resuming the Reliable OBEX Session shall restore all OBEX Connection settings; the interrupted operation shall also resume from where it left off when the transport disconnection occurred.

The use of the OBEX operations to create, suspend, resume, or close a Reliable OBEX Session is described in Sections 5.8 - 5.11.

4.8 Using Reliable Object Exchange Sessions

The following bulleted list describes the requirements that apply to the use of Reliable OBEX Sessions:

- No more than one Reliable OBEX Session shall exist at any time on each transport connection, which is equivalent to the L2CAP channel connection. If a CREATE SESSION command is issued while an active Reliable Session already exists, the operation shall be rejected with the Service Unavailable (0xD3) response code. If a CREATE SESSION command is issued while a suspended Reliable Session exists, the operation may be rejected with the Service Unavailable (0xD3) response code, or the suspended session may be removed and a new Reliable Session created.
- The CREATE SESSION command shall be issued prior to the application profile's OBEX Connection.
- The CLOSE SESSION command shall be used to remove an active or suspended session. This command shall be issued after the application profile's OBEX Disconnection in order to close the active session. This command shall not be rejected if the provided Session ID value pertains to a valid session, otherwise the Forbidden (0xC3) response code shall be used. Once this has been completed, another CREATE SESSION command may be issued to create a new session.
- The SUSPEND SESSION command may be issued to suspend a Reliable OBEX Session. A transport disconnection shall be issued after a SUSPEND SESSION completes.
- Reliable OBEX Sessions may be suspended or closed by the application, if it determines that this feature is no longer needed. The profile may suspend a session at any time, even during an active OBEX operation. Suspending a Reliable OBEX Session shall halt the OBEX Connection initiated during the Reliable Session, along with any OBEX operation active at the time of suspension.
- When a Reliable OBEX Session is suspended, OBEX will preserve the connection information and state of any on-going OBEX operation. However, the overall state of the system at the time of suspension may not be preserved. Therefore, if folder hierarchies have changed (via the SETPATH command) or other system information has been altered during the context of the Reliable OBEX Session, both devices shall preserve this information so that it can be restored when the session is resumed. If storage and restoration of this state information is not supported, the RESUME SESSION command shall be rejected and a new session created (after closing the suspended session) in its place.

- If SRM was active (e.g. in the middle of a PUT or GET operation) when the Reliable OBEX Session was suspended, SRM shall be restored when the session is resumed. However, in accordance with Section 4.6, SRM will be automatically disabled after the resumed operation completes.
- All Reliable Session operations (except CLOSE SESSION) may negotiate suspend timeout values. These values are recommended to be small, in order to minimize the time a device is required to store information for a suspended Reliable Session.

5 OBEX Interoperability Requirements

5.1 OBEX Operations Used

Table 5.1 shows the OBEX operations which are specified by the OBEX protocol. The application profiles using GOEP must specify which operations must be supported to provide the functionality defined in the application profiles.

Operation no.	OBEX Operation
1	Connect
2	Disconnect
3	Put
4	Get
5	Abort
6	SetPath
7	Action
8	Session

Table 5.1: OBEX Operations

The IrOBEX specification does not define how long a client should wait for a response to an OBEX request. However, implementations which do not provide a user interface for canceling an OBEX operation should wait a reasonable period between a request and response before automatically canceling the operation. A reasonable time period is 30 seconds or more.

5.2 OBEX Headers

See [1] for discussion of the available OBEX headers.

The non-Body headers such as Name or Description that may exceed the allowed OBEX packet size may be issued multiple times in consecutive packets within a single PUT/GET operation.

5.3 Initialization of OBEX

If OBEX authentication is supported and used by the Server and the Client, the initialization for this authentication (see also Section 5.4.2) shall be done before the first OBEX connection is established. The initialization can be done at any time before the first OBEX connection.

Authentication is done using an OBEX password, which may be related to the Bluetooth passkey, if present. Even if the same code is used for Bluetooth authentication and OBEX authentication, the user must enter that code twice, once for each authentication. After entering the OBEX password in both the Client and Server, the OBEX password is stored in the Client and the Server, and it may be used in the future for authenticating the Client and the Server. When an OBEX connection is established, OBEX

authentication may be used by the server to authenticate the client, and the client may also authenticate the server.

5.4 Establishment of OBEX Connection

For the object exchange, the OBEX connection can be made with or without OBEX authentication. In the next two subsections, both of these cases are explained. All application profiles using GOEP must support an OBEX session without authentication, although authentication may be used. SRM headers shall not be sent in the Connect request or response packets (note, this is to preserve backwards compatibility). SRM shall be enabled through Put and Get operations only, as shown in Sections 5.5 and 5.6.

5.4.1 OBEX Connection without Authentication

Figure 5.1 depicts how an OBEX connection is established using the CONNECT operation.

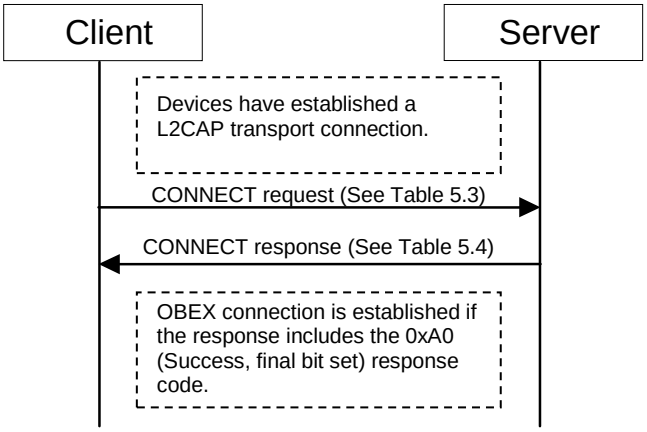

Figure 5.1: Establishment of OBEX connection without Authentication

The CONNECT request indicates a need for connection and may also indicate which service is used. The fields in the CONNECT request are listed below:

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for CONNECT	0x80	M	-
Field	Connect Packet Length	Varies	M	-
Field	OBEX Version Number	Varies	M	-
Field	Flags	Varies	M	-
Field	Max OBEX Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the current sequence number for this OBEX packet.
Header	Target	Varies	C2	Used to indicate the specific Service.
Header	Count	Varies	O	

Table 5.2: Fields and Headers in CONNECT Request

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The use of the Target header is mandatory for some application profiles. The application profiles define explicitly whether they use it or not. For the Target header, the example value could be ‘IRMC-SYNC’ to indicate the IrMC synchronization service.

The response to the CONNECT request includes the fields listed below:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for CONNECT request	Varies	M	0xA0 for success
Field	Connect Response Packet Length	Varies	M	-
Field	OBEX Version Number	Varies	M	-
Field	Flags	Varies	M	-
Field	Max OBEX Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the current sequence number for this OBEX packet.
Header	Connection Identifier	Varies	C2	The header value specifies the current connection to the specific service.
Header	Who	Varies	C2	The header value matches the Target header value.

Table 5.3: Fields and Headers in CONNECT Response

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The Who and Connection Identifier headers shall be used if the Target header is used in the Connect request.

5.4.2 OBEX Connection with Authentication

OBEX authentication is used to authenticate the Client and the Server. Figure 5.2 depicts establishment of an OBEX connection with authentication.

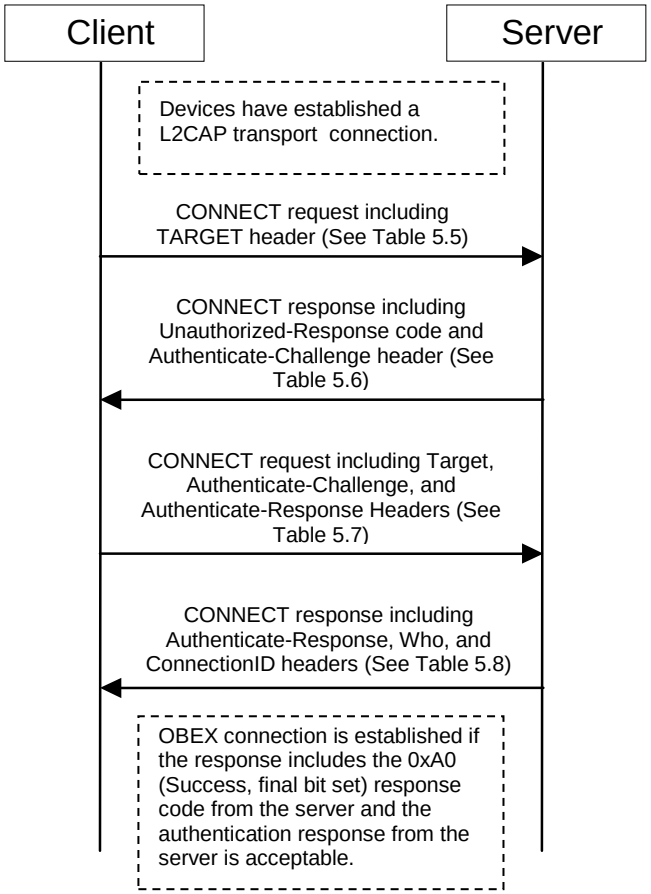

Figure 5.2: Establishment of OBEX Connection with Authentication

The first CONNECT request indicates a need for connection and which service is used. The fields and the header in the CONNECT request are listed in Table 5.4:

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for CONNECT	0x80	M	-
Field	Connect Packet Length	Varies	M	-
Field	OBEX Version Number	Varies	M	-
Field	Flags	Varies	M	-
Field	Max OBEX Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the current sequence number for this OBEX packet.
Header	Target	Varies	C2	Used to indicate the specific Service

Table 5.4: Fields and Headers in CONNECT Request When Authentication Used

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The use of the Target header is mandatory for some application profiles. The application profiles define explicitly whether they use it or not. For the Target header, the example value could be ‘IRMC-SYNC’ to indicate the IrMC synchronization service.

The response to the first CONNECT request, which authenticates the Client, includes the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for CONNECT request	Varies	M	0xC1 for Unauthorized, because OBEX authentication is used.
Field	Connect Response Packet Length	Varies	M	-
Field	OBEX Version Number	Varies	M	-
Field	Flags	Varies	M	-
Field	Max OBEX Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the current sequence number for this OBEX packet.
Header	Authenticate Challenge	Varies	M	Carries the digest-challenge string.

Table 5.5: Fields and Headers in First CONNECT Response When Authenticating

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

The second CONNECT request has the following fields and headers in this order:

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for CONNECT	0x80	M	-
Field	Connect Packet Length	Varies	M	-
Field	OBEX Version Number	Varies	M	-
Field	Flags	Varies	M	-
Field	Max OBEX Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the current sequence number for this OBEX packet.
Header	Target	Varies	C2	-
Header	Authenticate Challenge	Varies	M	Carries the digest-challenge string.
Header	Authenticate Response	Varies	M	Carries the digest-response string. This is the response to the challenge from the Server.

Table 5.6: Fields and Headers in Second CONNECT Request When Authentication Used

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: see Table 5.3

The response to the second CONNECT request has the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for CONNECT request	Varies	M	0xA0 for success
Field	Connect Response Packet Length	Varies	M	-
Field	OBEX Version Number	Varies	M	-
Field	Flags	Varies	M	-
Field	Max OBEX Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the current sequence number for this OBEX packet.
Header	Connection Identifier	Varies	C2	The header value specifies the current connection to the specific service.
Header	Who	Varies	C2	The header value matches the Target header value.
Header	Authenticate Response	Varies	M	Carries the digest-response string. This is the response to the challenge from the Client.

Table 5.7: Fields and Headers in Second CONNECT Response When Authenticating

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The Who and Connection Identifier headers shall be used if the Target header is used in the Connect request.

If the response code from the Server is successful, and the Client accepts the authentication response from the Server, the connection is established and authenticated.

5.5 Pushing Data to Server

The data object(s) is pushed to the Server using the PUT operation of the OBEX protocol. The data can be sent in one or more OBEX packets.

5.5.1 Pushing Data over an OBEX Connection

The Client may initiate SRM by sending a Single Response Mode header. SRM allows the Client to issue unacknowledged request packets until the completion of the PUT operation. Additional response packet(s) can be issued during the PUT operation if requested by the Server through the optional SRMP header (see Section 4.6 for SRMP usage requirements).

If a PUT operation is ongoing and the server cannot receive more Body data after the initial Body header has been received, and SRM is enabled, the server may send a PUT response including an unsuccessful response code (excluding CONTINUE and SUCCESS) prior to the entire object being received at the server. Alternatively, the server may choose to disconnect the transport in this case without waiting for the PUT operation to complete. If the server sent a PUT response without waiting for the last chunk of the object, it shall not send another response upon arrival of the entire object in a PUT Request including the Final Bit.

If the Client receives a PUT response (other than SUCCESS or CONTINUE) in the middle of the SRM operation it may do one of three things: 1) Cease processing the PUT operation, 2) Complete its queued PUT packets and then cease processing the PUT operation, or 3) Complete the PUT operation and then process the early response packet. In all cases the Server shall drop any additional Body data until the operation completes. The Server knows the operation is complete when it receives another non-Body header indicating the beginning of a new operation. This non-Body header excludes the Session Sequence Number (SSN) header which arrives in all OBEX packets during a Reliable Session.

In the case the Client receives an early PUT response with unexpected SUCCESS or CONTINUE response code, the transport shall be disconnected.

The first PUT request includes the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for PUT	0x02 or 0x82	M	-
Field	Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the current sequence number for this OBEX packet.
Header	Connection Identifier	Varies	C2	The header value specifies the current connection to the specific service.
Header	Single Response Mode	0x01	C3	Used to indicate a Single Response Mode enable request.

Header	Name	Varies	M	The header value is the name of a single object, object store, or log information.
Header	Body/End of Body	Varies	M	End of Body identifies the last chunk of the object body.

Table 5.8: Fields and Headers in PUT Request

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The Connection Identifier header is mandatory if the Target header is used when establishing the OBEX connection. This header should be used in the first request of the PUT operation only.

C3: The Single Response Mode header is mandatory if Single Response Mode is supported and is requested by the application profile. If this mode is not supported, this header shall not be sent.

Other headers, which can be optionally used, are specified in [5].

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

Field/ Header	Name	Value	Status	Explanation
Field	Response code for PUT	Varies	M	0x90 for continue or 0xA0 for success
Field	Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the sequence number expected in the next OBEX packet.
Header	Single Response Mode	0x01	C2	Specifies acceptance of the Single Response Mode enable request.
Header	Single Response Mode Parameters	0x01	O	This header allows the Server to instruct the Client to wait, after sending its request, until the next response has been received before issuing another PUT request.

Table 5.9: Fields and Headers in PUT Response

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The Single Response Mode header is mandatory if Single Response Mode is supported and is requested by the application profile. If this mode is supported, requested by the application profile, and a SRM header has been received in the previous PUT request, this header shall be sent back, thereby allowing this mode to be enabled. This header shall not be sent if the Response code indicates a failure.

Other headers, which can be optionally used, are specified in [5].

5.5.2 Deleting an Object over an OBEX Connection

Deleting an object shall be performed by using the PUT operation, as shown above, but without sending any Body or End Of Body headers.

5.6 Pulling Data from Server

The data object(s) is pulled from the Server using the GET operation of the OBEX protocol. The data can be sent in one or more OBEX packets.

5.6.1 Pulling Data over an OBEX Connection

The Client may initiate SRM by sending a Single Response Mode header. SRM allows the Server to issue unacknowledged response packets until the completion of the GET operation. Additional request packet(s) can be issued during the GET operation if requested by the Client through the optional SRMP header (see Section 4.6.2 for SRMP usage requirements). Additionally, see Section 4.6 for discussion on the GET Request phase of operation when SRM is enabled.

In the case where SRM is enabled, but the server receives an unexpected GET request, this packet shall be ignored.

The first GET request includes the following fields and headers.

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for GET	0x03 or 0x83	M	-
Field	Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the current sequence number for this OBEX packet.
Header	Connection Identifier	Varies	C2	The header value specifies the current connection to the specific service.
Header	Single Response Mode	0x01	C3	Used to indicate a Single Response Mode enable request
Header	Single Response Mode Parameters	0x01	O	This header allows the Client to instruct the Server to wait, after sending its response, until the next request has been received before issuing another GET response.
Header	Type	Varies	C4	Indicates the type of the object to be pulled.
Header	Name	Varies	C4	The header value is the name of a single object, object store, or log information.

Table 5.10: Fields and Headers in GET Request

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The Connection Identifier header is mandatory if the Target header is used when establishing the OBEX connection. This header should be used in the first request of the GET operation only.

C3: The Single Response Mode header is mandatory if Single Response Mode is supported and is requested by the application profile. If this mode is not supported, this header shall not be sent.

C4: Either the Type header or the Name header shall be included in the GET request when it is sent to the server.

Other headers, which can be optionally used, are specified in [5].

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

Field/ Header	Name	Value	Status	Explanation

Field/ Header	Name	Value	Status	Explanation
Field	Response code for Get	Varies	M	0x90 or 0xA0 if the packet is the last the object
Field	Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the sequence number expected in the next OBEX packet.
Header	Single Response Mode	0x01	C2	Specifies acceptance of the Single Response Mode enable request.
Header	Name	Varies	O	The header value is the name of a single object, object store, or log information.
Header	Body/End of Body	Varies	M	End of Body identifies the last chunk of the object body.

Table 5.11: Fields and Headers in GET Response

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The Single Response Mode header is mandatory if Single Response Mode is supported and is requested by the application profile. If this mode is supported, requested by the application profile, and a SRM header has been received in the previous GET request, this header shall be sent back, thereby allowing this mode to be enabled. This header shall not be sent if the Response code indicates a failure.

Other headers, which can be optionally used, are specified in [5].

5.7 Performing an Action on a Server Object

The ACTION request includes the following fields and headers.

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for ACTION	0x86	M	-
Field	Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the sequence number expected in the next OBEX packet.
Header	Connection Identifier	Varies	C2	The header value specifies the current connection to the specific service.
Header	Action Identifier	Varies	M	This header specifies the type of Action to perform – Copy (0x00), Move/Rename (0x01), or Set Permissions (0x02).
Header	Name	Varies	M	Indicates the name of the object to perform the action upon.

Field/ Header	Name	Value	Status	Explanation
Header	DestName	Varies	C3	Indicates the destination name for a Copy or Move/Rename action.
Header	Permissions	Varies	C4	Indicates the permissions for a Set Permissions action.

Table 5.12: Fields and Headers in ACTION Request

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The Connection Identifier header is mandatory if the Target header is used when establishing the OBEX connection.

C3: This header is only used for Copy and Move/Rename actions, and is mandatory for these actions. This header shall express relative path names to the current folder.

C4: This header is only used for the Set Permissions action, and is mandatory for this action.

Other headers, which can be optionally used, are specified in [5].

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

Field/ Header	Name	Value	Status	Explanation
Field	Response code for ACTION request	0xA0	M	-
Field	Packet Length	0x03	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the sequence number expected in the next OBEX packet.

Table 5.13: Fields and Headers in ACTION Response

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

Other headers, which can be optionally used, are specified in [5].

5.8 Setting up a Reliable Session with the Server

The CREATE SESSION request includes the following fields and headers.

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for SESSION	0x87	M	-
Field	Packet Length	Varies	M	-

Field/ Header	Name	Value	Status	Explanation
Header	Session-Parameters	Varies	M	This header specifies the Session Opcode (0x00 for Create Session), Nonce, and Device Address tag/length/value triplets. Optionally, the Timeout triplet can be included.

Table 5.14: Fields and Headers in CREATE SESSION Request

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

Field/ Header	Name	Value	Status	Explanation
Field	Response code for SESSION request	0xA0	M	-
Field	Packet Length	Varies	M	-
Header	Session-Parameters	Varies	M	This header specifies the Device Address, Nonce, and Session ID tag/length/value triplets. Optionally, the Timeout triplet can be included.

Table 5.15: Fields and Headers in CREATE SESSION Response

5.9 Suspending a Reliable Session with the Server

A Reliable OBEX Session can be suspended in two ways:

1. A transport disconnection occurs, which intrinsically enacts a suspension of the active session.
2. The SUSPEND SESSION is performed. This command can be issued during an active operation (an immediate suspend) or outside the scope of an active operation (a graceful suspend). In the former case, it is possible for Suspend to act like Abort, where the SUSPEND SESSION can occur anywhere during the processing of the active operation.

The SUSPEND SESSION request includes the following fields and headers.

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for SESSION	0x87	M	-
Field	Packet Length	Varies	M	-
Header	Session-Parameters	Varies	M	This header specifies the Session Opcode (0x02 for Suspend Session) tag/length/value triplet. Optionally, the Timeout triplet can be included.

Table 5.16: Fields and Headers in SUSPEND SESSION Request

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

Field/ Header	Name	Value	Status	Explanation
Field	Response code for SESSION request	0xA0	M	-
Field	Packet Length	Varies	M	-
Header	Session-Parameters	Varies	M	This header specifies the Session Timeout tag/length/value triplet. Optionally, the Timeout triplet can be included.

Table 5.17: Fields and Headers in SUSPEND SESSION Response

5.10 Resuming a Reliable Session with the Server

The RESUME SESSION request includes the following fields and headers.

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for SESSION	0x87	M	-
Field	Packet Length	Varies	M	-
Header	Session-Parameters	Varies	M	This header specifies the Session Opcode (0x03 for Resume Session), Device Address, Nonce, Session ID, and File Offset tag/length/value triplets. Optionally, the Timeout triplet can be included.

Table 5.18: Fields and Headers in RESUME SESSION Request

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

Field/ Header	Name	Value	Status	Explanation
Field	Response code for SESSION request	0xA0	M	-
Field	Packet Length	Varies	M	-
Header	Session-Parameters	Varies	M	This header specifies the Device Address, Nonce, Session ID, Next Sequence Number, and File Offset tag/length/value triplets. Optionally, the Timeout triplet can be included.

Table 5.19: Fields and Headers in RESUME SESSION Response

5.11 Closing a Reliable Session with the Server

The CLOSE SESSION request includes the following fields and headers.

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for SESSION	0x87	M	-
Field	Packet Length	Varies	M	-
Header	Session-Parameters	Varies	M	This header specifies the Session Opcode (0x01 for Close Session) and Session ID tag/length/value triplets.

Table 5.20: Fields and Headers in CLOSE SESSION Request

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

Field/ Header	Name	Value	Status	Explanation
Field	Response code for SESSION request	0xA0	M	-
Field	Packet Length	0x03	M	-

Table 5.21: Fields and Headers in CLOSE SESSION Response

5.12 Disconnection

The DISCONNECT request indicates a disconnection for a particular service. This command has no impact on a Reliable OBEX Session, as only a transport disconnection or SUSPEND SESSION command will cause a suspension of the Reliable OBEX Session. However, as a DISCONNECT command signals the end of an application profile connection, it is recommended that a CLOSE SESSION command is issued (if a Reliable Session exists) after the disconnection completes.

The fields and the header in the DISCONNECT request are listed in Table 5.22:

Field/ Header	Name	Value	Status	Explanation
Field	Opcode for DISCONNECT	0x81	M	-
Field	Packet Length	Varies	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the sequence number expected in the next OBEX packet.

Field/ Header	Name	Value	Status	Explanation
Header	Connection Identifier	Varies	C2	The header value specifies the current connection to the specific service.

Table 5.22: Fields and Headers in DISCONNECT Request

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

C2: The Connection Identifier header is mandatory if the Target header is used when establishing the OBEX connection.

The response to the DISCONNECT request from the Server includes the following fields and headers:

Field/ Header	Name	Value	Status	Explanation
Field	Response code for DISCONNECT request	Varies	M	0xA0 for success
Field	Packet Length	0x03	M	-
Header	Session Sequence Number	Varies	C1	This header specifies the sequence number expected in the next OBEX packet.

Table 5.23: Fields and Headers in DISCONNECT Response

C1: The Session Sequence Number header is mandatory if an active Reliable OBEX Session exists when this operation is issued.

6 SDP Interoperability Requirements

This section covers the SDP requirements of GOEP v2.0. Section 6.1 covers requirements for new profiles that require no backwards compatibility with GOEP v1.1 implementations. Section 6.2 covers requirements for legacy profiles which are enhanced to utilize GOEP v2.0 and require backwards compatibility with earlier implementations of the same profile.

6.1 Requirements for Application Profiles which do not require backwards compatibility

For application profiles based on GOEPv2.0 or later that do not require backwards compatibility, the GoepL2capPSM shall not be present in the SDP record of the relevant application profile on the server device. The ProtocolDescriptorList shall not contain the UUID for RFCOMM. The Parameter for the ProtocolDescriptorList containing the L2CAP UUID shall contain the PSM on which the profile is running OBEX. The client conforming to the application profile based on GOEP v2.0 shall read the ProtocolDescriptorList in the SDP record of the profile of the server device to determine the PSM to which to create an OBEX connection.

6.2 Requirements for Application Profiles which require backwards compatibility with GOEP v1.x

This section applies only to profiles referencing GOEP v2.0 which are enhancements of older profiles that utilized OBEX over RFCOMM as defined in GOEP v1.1. To maintain backward compatibility with previous versions of the profile using GOEP v1.1, an existing profile enhanced to reference GOEP v2.0 should expose an SDP record which is identical to the SDP record defined for the last version of the profile which referenced GOEP v1.1, but with the exceptions that:

- The profile version number in the BluetoothProfileDescriptorList attribute shall be incremented to match the version number of the enhanced profile specification. If earlier versions of the profile do not include this SDP attribute it shall be included in revised versions.
- The SDP records shall include the GoepL2capPsm attribute [1].
- New profile-specific attributes may be added if necessary.

A client conforming to the enhanced profile supporting GOEP v2.0 shall check for the existence of the GoepL2capPsm attribute [1] in the SDP record of the remote GOEP v2.0 enhanced service it is intending to use.

If the GoepL2capPsm attribute exists in the server's application profile record, the value of the attribute is the L2CAP PSM to which the client shall connect if it initiates an OBEX connection to the server device. When connecting via GOEP v2.0, OBEX over L2CAP shall be used, and the new features of IrOBEX v1.5 may be used within the application profile. The profile client implementation may proceed to create a GOEP v2.0

connection to the server if the GoepL2capPsm attribute is present in the server's service record and only on the PSM indicated by the value of the attribute.

If the GoepL2capPsm attribute does not exist in the server's application profile record the profile client shall only connect to the remote service using GOEP v1.1. In this case, the client shall retrieve the ProtocolDescriptorList attribute from the profile server to confirm that the profile server device supports GOEP v1.1 using OBEX over RFCOMM and to obtain the RFCOMM channel number to use. If a connection is made to the profile on the server device using GOEP v1.1, OBEX over RFCOMM shall be used and the new features of IrOBEX v1.5 shall not be used within the application profile.

Note that clients that support GOEP v1.1 but not GOEP v2.0 are expected to ignore the GoepL2capPsm attribute and use OBEX over RFCOMM to access the service.

7 Generic Access Profile Requirements

This profile requires compliance to the protocol requirements of the Generic Access Profile.

7.1 L2CAP Interoperability Requirements

7.1.1 MTU option

According to [1], OBEX packets must fit completely within a single L2CAP SDU, so an implementation shall either configure the L2CAP MTU to be large enough to hold the largest OBEX packet or configure the OBEX packet size to be small enough to fit within the L2CAP MTU. In any case the OBEX packet size should be set as large as possible, especially in the case that the Single Response Mode feature of IrOBEX is not supported. By definition:

- The L2CAP MTU refers to the SDU size
- The SDU is the application packet (OBEX packet in GOEP)

Therefore the Maximum OBEX Packet Length (MOPL) is the smaller of the largest OBEX packet that the implementation allows and the actual L2CAP MTU for the channel in the inbound direction. The MOPL is configured separately for each direction and may be different for each direction.

The MOPL that an OBEX client can receive is stated in the OBEX Connect request, this shall be less than or equal to the MTU configured for packets received by L2CAP for the OBEX connection. A separate, and possibly different, MOPL is stated by the OBEX server in the OBEX Connect response, this shall be less than or equal to the L2CAP MTU configured by L2CAP for the OBEX connection.

GOEP implementations shall support a minimum MOPL packet size of 255 bytes. For OBEX over L2CAP this means that implementations shall also support a minimum MTU of 255 bytes.

7.1.2 Flow and Error Control Option

The L2CAP channel for OBEX connections shall be configured to use Enhanced Retransmission Mode.

The Maximum PDU Payload Size (see [2]) should be set in accordance with the recommendations in the Core Specification. In particular many OBEX application profiles benefit from running over AMP controllers. Therefore the MPS should be set to be equal to the largest of the ACL buffers across all of the AMP and BR/EDR Controllers in a device.

Additionally, to maximize throughput (especially if the device intends to move the L2CAP channel carrying OBEX from one controller to another) the TxWindow size value should be configured as large as possible. In particular if OBEX is configured to use Single Response Mode this will tend to increase the throughput available to the application profiles.

7.1.3 Frame Checksum (FCS) Option

Application profiles can transmit large objects (including executable data). When running GOEP over the BR/EDR Controller, the value of the FCS option should be configured to the default value of “16-bit FCS”. When running GOEP over AMP Controllers the FCS option should be configured to the value of “No FCS”.

The FCS option can be reconfigured between these two values. An implementation may reconfigure the FCS option for the L2CAP channel for GOEP after the channel is moved between BR/EDR and AMP controllers.

Other L2CAP Configuration options are excluded from GOEP.

7.2 Link Control (LC) Interoperability Requirements

7.2.1 Inquiry and Inquiry Scan

For the inquiry, the returned CoD in the FHS packet shall indicate that Object Transfer service is supported (see [8]). The major and minor device classes depend on the device supporting this profile. Therefore, usage of them is not defined in this profile.

For information on Limited discoverable mode, Limited Inquiry mode, see GAP [2]

8 Generic Access Profile Interoperability Requirements

This profile requires compliance to the Generic Access Profile. This section defines the support requirements with regards to procedures and capabilities defined in GAP.

8.1 Modes

Table 8.1 shows the support status for Modes within this profile.

	Procedure	Support in Client	Support in Server
1	**Discoverability modes**		
	Non-discoverable mode	N/A	M
	Limited discoverable mode	N/A	C1
	General discoverable mode	N/A	C1
2	**Connectability modes**		
	Non-connectable mode	N/A	O
	Connectable mode	N/A	M
3	**Pairing modes**		
	Non-pairable mode	N/A	O
	Pairable mode	N/A	M

Table 8.1: Modes

C1: The Limited discoverable mode should be used, but if the Server device for some reason (e.g. lack of a sufficient user interface) wants to be visible at all times, the General discoverable mode can be used instead.

8.2 Security Aspects

Table 8.2 shows the support status for Security aspects within this profile.

	Procedure	Support in Client	Support in Server
1	Authentication	M	M
2	Security modes		
	Security modes 1	X	X
	Security modes 2	O1	O1
	Security modes 3	X	X
	Security modes 4	M	M

Table 8.2: Security Aspects

O1: Security Mode 2 may only be used for backwards compatibility when the remote device does not support Secure Simple Pairing

8.3 Idle Mode Procedures

Table 8.3 shows the support status for Idle mode procedures within this profile.

	Procedure	Support in Client	Support in Server
1	General inquiry	M	N/A
2	Limited inquiry	O	N/A
3	Name discovery	M	N/A
4	Device discovery	M	N/A
5	Bonding	M (Note 1)	M (Note 1)

Table 8.3: Idle Mode Procedures

Note 1: See section 8.3.1

8.3.1 Bonding

It is mandatory for the Client and Server to support bonding. Bonding may be required before permitting communication between a Client and a Server.

The usage of bonding is optional for both Client and Server. The bonding procedures are defined in GAP [2].

9 References

9.1 Normative References

[1] IrDA Interoperability, version 2.0 or Later

[2] Core Specification, version 3.0 or later

[3] RFCOMM with TS 07.10, Specification of the Bluetooth System, version 1.2 or Later

[4] ETSI, TS 07.10, Version 6.3.0

[5] Infrared Data Association, IrDA Object Exchange Protocol (IrOBEX) with Published Errata, Version 1.5

[6] Serial Port Profile, version 1.2 or Later

[7] Internet Assigned Numbers Authority, IANA Protocol/Number Assignments Directory (http://www.iana.org).

[8] Bluetooth SIG Assigned Numbers (http://www.bluetooth.org)