Obex Errata to
IrOBEX specification version.1.0

Last Modified: August 20, 1997

Following are a list of corrections/clarifications and suggested amendments or changes to the IrOBEX specification Version 1.0 dated January 22 1997. Most of these Errata were approved and closed at the IrDA meeting in Portland on July 15, and open errata have been moved to the end of this document in Section 2.

The closed errata should be considered an integral part of the IrOBEX version 1.0 specification. No change to the version number is required based on these changes.

The points are classified according to the following scheme:

- CORRECTION: a change required to correct an error in the existing document corrections may change the specified behaviour of the protocol to match that which was originally intended by the authors

- CLARIFICATION: textual enhancement that provides clearer explanation of a protocol element without changing any behavior

- MODIFICATION: a modification of the currently specified behaviour that adds but does not delete any function from the protocol

- CHANGE: a modification of the currently specified protocol that may add and or delete function from the protocol

- PROBLEM: a known problem for which an alteration to the document has yet to be proposed.

1. Closed Errata

All errata in this section were approved and closed at the July 1997 IrDA meeting.

1.1. Connect - single packet limitation in each direction (4.3.1)

CLARIFICATION
The connect operation shall be confined to a single packet in each direction.

Version 1.0 was ambiguous on this point. This is sufficient for current use and for intended extensions. If for instance security is implemented, there is an excellent chance it can be fit in one packet of reasonable size. Alternatively, an additional operation could be created that was dedicated to the security operation.

1.2. Disconnect - single packet limitation in each direction (4.3.2)

CLARIFICATION
The Disconnect operation shall be confined to a single packet in each direction.

Version 1.0 was ambiguous on this point. This is sufficient for current use and for reasonable extensions, although there are no extensions currently under consideration.

1.3. Abort - single packet limitation in each direction (4.3.6)

CLARIFICATION
The Abort operation shall be confined to a single packet in each direction.

Version 1.0 was ambiguous on this point. This is sufficient for current use and for reasonable extensions, although there are no extensions currently under consideration.

1.4. Put Response fits in a single packet (4.3.3.2)

CLARIFICATION
The Put Response shall be confined to a single packet.

Version 1.0 was ambiguous on this point. The Put Response contains only a 2 byte response code in all known implementations. A Description header might be useful in a response, but 512 bytes is a generous description.

1.5. Maximum OBEX Packet Length (4.3.1.3)

MODIFICATION
Change smallest acceptable value for the negotiated "Maximum size OBEX packet" from a recommendation of 60 bytes to a requirement of 255 bytes.

This is in order to provide a greater likelihood of meeting the requirement for single packet requests and responses (documented in other sections listed above) in a broad range of cases. It has been demonstrated that this is not an unacceptable RAM burden on even very compact implementations—lengths still fit in one byte.

1.6. Get operation Final bit clarification (4.3.4)

CLARIFICATION
The diagram after the 1st paragraph should not say that the final bit is not used. This is incorrect, its use is discussed in the following paragraphs. Remove the statement that the final bit is not used.

1.7. Discussion of Request Packets (4.1)

CLARIFICATION
The second paragraph does not discuss the use of the final bit in operations that have more than one response packet, as in a length Get operation. This is clarified as follows:

Every request packet in an operation has the opcode of that operation. The high order bit of the opcode is called the Final bit. It is set to indicate the last packet for the request. For example, a Put operation sending a multi-megabyte object will typically require many Put packets to complete, but only the last packet will have the Final bit set in the Put opcode.

If the operation requires multiple response packets to complete after the Final bit is set in the request (as in the case of a Get operation returning an object that is too big to fit in one response packet), the server will signal this with the "Continue" response code, telling the client that it should ask for more. The client (requestor) should send another request packet with the same opcode, Final bit set, and no headers if it wishes to continue receiving the object. If the client does not want to continue the operation, it should send an Abort packet.

1.8. Discussion of Response Packets (4.2)

CLARIFICATION
The second paragraph is confusing in its discussion of multi-packet responses, and appears to conflict with how the Get operation is specified. This is clarified as follows:

The high order bit of the response code is called the Final bit. In OBEX 1.0 it is always set for response packets. Its meaning is slightly different from the request packet final bit—the response packet final bit tells the other side (the client) that it is once again the client's turn to send a packet. If the response packet carries a success or failure response code, the client is free to begin a new operation. However, if the response code is "Continue" as is often the case in a lengthy Get operation, then the client next issues a continuation of the Get request. See the Get operation description below for examples.

1.9. Clarification on timeouts between packets (new section in Ch 4)

CLARIFICATION
IrOBEX does not impose any timeouts between OBEX packets. Deciding whether to continue or cancel a session is normally left to user control, since user actions are often what create long periods between packets. In devices with reasonable user interface capabilities, timeouts are not recommended. However, this specification does not prohibit the use of timeouts, and some devices may find them useful or necessary, particularly when sufficient user interface to understand and/or control the reason for delay is not available.

1.10. Sending objects that may have read errors (4.3.3.5)

CLARIFICATION
When sending the last portion of an object in an End of Body header, ambiguity arises if there is any chance that there are read errors in this last portion. This is because End of Body normally triggers the receiving side to close the received object and indicate successful completion. If an Abort packet subsequently arrives, it is "too late".

The recommended approach when sending objects which may have such errors is to send the End of Body header only when the sender knows that the entire object has been safely read, even if this means sending an empty End of Body header at the end of the object. This applies to both Get and Put operations.

1.11. Mapping Obex packets to TTP/LMP packets (new section on implementation notes)

CLARIFICATION
There is no requirement on the alignment of OBEX packets with TTP or IrLMP PDUs (packets) except that all OBEX data shall be carried in Data packets, not in LMP/TTP Connect or Disconnect packets.

OBEX specifically does not use TTP segmentation and reassembly. There is no relationship between OBEX packets and TTP SDUs, and TTP connect should not contain the MaxSDUSize parameter.

1.12. Response code values (4.2.1, 4.3)

CORRECTION
Table 4.2.1 has incorrect response code values for the last group: 0x60 through 0x65 should instead be 0x50 thru 0x55.

In the 2nd paragraph of section 4.3, 0xE1 should be 0xD1.

1.13. Meaning of Length header in Connect Packet (4.3.1.4, 4.3.1.6)

CLARIFICATION
When used in the Connect operation, the Length header contains the length in bytes of the bodies of all the objects that the sender plans to send. Note that this length cannot be guaranteed correct, so while the value may be useful for status indicators and resource reservations, the receiver should not croak if the length is not correct. The receiver can depend on body headers to accurately indicate the size of an object as it is delivered, and the End of Body header to indicate when an object is complete.

1.14. Page numbering

CORRECTION
IrOBEX page numbering starts with the cover of the specification but is reset by the table of contents, unlike the numbering in the IrLAP and IrLMP specifications. The page numbering should be made consistent with those specifications by numbering continuously from front to back.

1.15. Clarifications of null terminated Unicode in headers (3.1.1)

CLARIFICATION
The length field (the 2 bytes immediately following the Header ID) for Unicode strings includes the 2 bytes of the null terminator (0x00, 0x00). Therefore the length of the string “Jumar” would be 12 bytes; 5 visible characters and the null terminator, each two bytes in length.

1.16. Use of the “Who” header with Put operation (3.2.10, 4.3.3.1)

MODIFICATION
Currently the Who header is allowed only with the Connect operation. However, it is possible for multiple applications to send data over a single OBEX connect, and it could be desirable to have source identification attached to individual objects. Therefore, the Who header shall be allowed with any operation to identify its source.

1.17. Use of byte stream terminology (Chapter 3)

CLARIFICATION
Use of the term “byte stream” to describe Headers encoded with 0x40 in the high 2 bits has caused confusion. Such headers will be referred to as a “byte sequence” or “sequence of bytes”.

1.18. Header table cleanup/clarification (Chapter 3)

CLARIFICATION
The encoding of header identifiers is confusing because the high order bits, which specify the encoding used in the header, obscure the nice linear sequence of header numbering. A note should be added after the table to clarify this.

1.19. “Command” operation removed (4.3.5)

MODIFICATION
The Command operation is removed. It complicates the creation of very small implementations of IrOBEX by introducing an irregularity into the state machines.

This operation has not been used in any implementation which the authors are aware of.

1.20. OBEX operations without Connect operation

MODIFICATION
It is highly recommended that implementation assume default values for connection parameters (currently just a minimum OBEX packet size of 255 bytes) and accept operation such as PUT and GET without first requiring a CONNECT operation.

1.21. Obex under Ultra

ADDITION
The following paragraphs replace the existing outline contained in the OBEX 1.0 specification

OBEX is constructed to take advantage of the benefits of a connection oriented transport, for instance by exchanging capabilities and version information just once in the Connect packet. However, some devices support only connectionless operations, such as those described by the IrDA Ultra proposals.

Ultra refers to communications carried out over connectionless data services of IrLAP, as described in “Guidelines for *Ultra* Protocols”. The Ultra protocols are not reliable and do not guarantee ordered delivery, so the request/response model of a full IrOBEX implementation is not appropriate. However, for delivery of small objects, the simple structure of OBEX requests and the header based object wrapper are still useful. A number of issues arise and are discussed in the following sections:

Ultra does not assure reliable ordered delivery. It does use IrLAP framing which has a CRC, so it is possible to tell if an individual packet arrived intact. It also uses a segmentation-reassembly (SAR) counter in each packet header, and a maximum time constraint between consecutive packets using SAR. The count allows up to 15 packets to be reassembled. The CRC and SAR acting together put a maximum upper bound on the size of an UltraOBEX packet of 15 times the max packet size permitted by the link speed of either 2400 bps or 9600 bps. In practice, transfers of 15 packets will take too long, and UltraOBEX is best suited for very small objects that will fit in one (or a few) packets. A typical well suited operation would be beaming a single name and phone number between two devices.

The OBEX Connect and Disconnect operations are not used in UltraOBEX, because UltraOBEX does not have any notion of a session with multiple operations. Each operation is completely distinct from every other, and each operation must fit within the SAR constraint described above. As a result, any version or capability information must be included as headers in the operation being performed.

UltraOBEX does not use response packets for Put operations. All feedback as to the success or failure of the operation is performed by some out-of-band means such as a beep or visual indication to the user that the operation was successful.

2. Open Errata

The following Errata were not approved or closed at the July 1997 IrDA meeting.

2.1. Pathnames and Telecom

ADDITION

Need to add explanation of pathnames used as part of Name Header, as is done in telecom protocol.

2.2. guaranteeing the uniqueness of “Who” - choice of Name space (3.2 10)

PROBLEM
It has been noted that without specifying the means for generating a Who header value, there is no guarantee of uniqueness. The 1.0 spec mentions a 16 byte UUID (where is reference?); should this be made mandatory? Should it be unique to device, to process that is sending, or what?

2.3. IAS Entries (Chapter 5)

MODIFICATION
It has been pointed out that the OBEX IrDA classname should now be IrDA:OBEX, since it is an IrDA specification. However, the old name may have to be used for backwards compatibility. We are soliciting feedback on this point; who would be affected by this change?

In addition, an attribute name of IrDA:OBEX:LsapSel is proposed for applications that use OBEX but have their own classname.