[MS-IRDA]:
IrDA Object Exchange (OBEX) Protocol Profile

Revision Summary

Date	Revision History	Revision Class	Comments
07/20/2007	0.1	Major	MCPP Milestone 5 Initial Availability
09/28/2007	0.1.1	Editorial	Revised and edited the technical content.
10/23/2007	0.1.2	Editorial	Revised and edited the technical content.
11/30/2007	0.1.3	Editorial	Revised and edited the technical content.
01/25/2008	0.1.4	Editorial	Revised and edited the technical content.
03/14/2008	0.1.5	Editorial	Revised and edited the technical content.
05/16/2008	0.1.6	Editorial	Revised and edited the technical content.
06/20/2008	1.0	Major	Updated and revised the technical content.
07/25/2008	1.0.1	Editorial	Revised and edited the technical content.
08/29/2008	2.0	Major	Updated and revised the technical content.
10/24/2008	3.0	Major	Updated and revised the technical content.
12/05/2008	4.0	Major	Updated and revised the technical content.
01/16/2009	4.0.1	Editorial	Revised and edited the technical content.
02/27/2009	4.0.2	Editorial	Revised and edited the technical content.
04/10/2009	4.0.3	Editorial	Revised and edited the technical content.
05/22/2009	4.0.4	Editorial	Revised and edited the technical content.
07/02/2009	4.0.5	Editorial	Revised and edited the technical content.
08/14/2009	4.0.6	Editorial	Revised and edited the technical content.
09/25/2009	4.1	Minor	Updated the technical content.
11/06/2009	4.1.1	Editorial	Revised and edited the technical content.
12/18/2009	4.1.2	Editorial	Revised and edited the technical content.
01/29/2010	5.0	Major	Updated and revised the technical content.
03/12/2010	6.0	Major	Updated and revised the technical content.
04/23/2010	6.0.1	Editorial	Revised and edited the technical content.
06/04/2010	6.0.2	Editorial	Revised and edited the technical content.
07/16/2010	6.0.2	No change	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
08/27/2010	6.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2010	6.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
11/19/2010	6.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
01/07/2011	6.0.2	No change	No changes to the meaning, language, or formatting of the technical content.
02/11/2011	6.0.2	No change	No changes to the meaning, language, or formatting of the technical content.

Contents

1   Introduction

The Infrared Data Association (IrDA) Object Exchange (OBEX) Protocol (**IrOBEX**) is specified by the Infrared Data Association in [IROBEX]. IrOBEX describes the two major elements of the protocol: a model for representing objects (and information that describes the objects), and a session protocol that provides a structure for the "conversation" between devices. The session protocol resides on top of **TinyTP**, as specified in [IRTTP], which provides a reliable transport between the two devices.

A major use of the IrDA OBEX protocol is a "push" or "pull" application, allowing rapid and impromptu communications between portable devices. For instance, a laptop user pushes a file to another laptop or PDA, or an industrial computer pulls status and diagnostic information from a piece of factory machinery.

The Microsoft implementation of this protocol implements the ability for the inclusion of Win32 error codes. Certain optional behaviors from [IROBEX] are also implemented, whereas other behaviors (such as pull operations) are not. This information is included in the appropriate sections of this specification.

1.1   Glossary

The following terms are defined in [MS-GLOS]:

> **client**
> **server**
> **universally unique identifier (UUID)**

The following terms are specific to this document:

> **Information Access Service (IAS):** Each device that implements the set of Infrared protocols, specifically [IRLMP], maintains an information base so that one **IrDA** device can discover what services another IrDA-compliant device offers as well as finding out about the device itself. This information is held in a number of objects in the information base and is accessed by communicating with the **IAS**.

> **IrDA:** Infrared Data Association.

> **IrOBEX:** An acronym for the IrDA-defined Infrared Object Exchange protocol, as specified in [IROBEX].

> **Link Service Access Point Selector (LSAP-SEL):** A selector that distinguishes between LSAPs within a station. Legal values for an **LSAP-SEL** lie in the range 0x00 through 0x7F. With the exception of the special **LSAP-SEL** values 0x00 (LM-IAS), 0x70 (Connectionless Data service), 0x71 through 0x7E (reserved), and 0x7F (reserved for broadcast and currently not implemented), the assignment of **LSAP-SEL** values is arbitrary. More information is specified in [IRLMP] section 3.1.2.

> **TinyTP:** Infrared Data Association Tiny Transport Protocol, as specified in [IRTTP].

> **Windows UUID:** The **IrDA UUID** used by this profile to identify itself to the **IrOBEX server** in the WHO header. The **Windows UUID** (16 Byte) value is b9c7fd98-e5f8-11d1-bfce-0000f8753890.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as described in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2   References

1.2.1   Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[IRLMP] Infrared Data Association, "IrDA Link Management Protocol v1.1", January 1996, http://irda.org/displaycommon.cfm?an=1&subarticlenbr=7

Note   There is a charge to download the specification.

[IROBEX] Infrared Data Association, "IrDA Object Exchange Protocol v1.2", March 1999, http://irda.org/displaycommon.cfm?an=1&subarticlenbr=7

Note   There is a charge to download the specification.

If you have any trouble finding [IROBEX], please check here.

[IRTTP] Infrared Data Association, "IrDA Tiny TP v1.1", October 1996, http://irda.org/displaycommon.cfm?an=1&subarticlenbr=7

Note   There is a charge to download the specification.

[ISO-8601] International Organization for Standardization, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO/IEC 8601:2004, December 2004, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40874&ICS1=1&ICS2=140&ICS3=30

Note   There is a charge to download the specification.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, http://www.ietf.org/rfc/rfc2119.txt

1.2.2   Informative References

[MS-ERREF] Microsoft Corporation, "Windows Error Codes", January 2007.

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

1.3   Overview

IrOBEX is used to transport opaque data objects over TinyTP. The primary use of the protocol, as specified in [IROBEX], is to connect two devices by using an infrared link and to allow sending and receiving of opaque data objects across the infrared link.

[IROBEX] describes the message and header formats and defines how the **client** and **server** exchange messages. The IrDA Object Exchange (OBEX) Protocol Profile specification [MS-IRDA]

describes the additional, user-defined header introduced in this profile, the implementation details in light of the additional header, and the portions of [IROBEX] that are not implemented.

This profile implements a header field for Win32 error codes. Information regarding certain peculiarities in Microsoft Windows® behavior with this profile is present in Appendix A.

1.4   Relationship to Other Protocols

The IrDA OBEX Protocol profile, as specified in [MS-IRDA], does not introduce any new dependencies on lower layer or parallel protocols beyond those specified in [IROBEX] section 1.3.

1.5   Prerequisites/Preconditions

Although not explicitly specified in [IROBEX], as part of the initialization an IrOBEX server registers a service access point (SAP) with the **Information Access Service (IAS)**, as specified in [IRLMP] section 3.1.2, for clients to be able to discover the service provided by the server.

1.6   Applicability Statement

The applicability of the IrDA OBEX Protocol profile, as specified in [MS-IRDA], is limited in the following ways:

- Data objects, specifically files, can be "pushed" only to the PC or by the PC. The reasons for this limitation are specified in section 3.1.5. In brief, the IrDA OBEX Protocol profile, as specified in [MS-IRDA], does not implement the GET operations defined in [IROBEX] section 3.3.4.

- Devices that implement this IrDA OBEX Protocol profile cannot exchange data objects with devices that require IrOBEX authentication as specified in [IROBEX] section 3.5. The reasons for this limitation are specified in sections 3.1.5 and 3.2.5. In brief, the IrDA OBEX Protocol profile, as specified in [MS-IRDA], does not implement the authentication sequence as specified in [IROBEX] section 3.5.

1.7   Versioning and Capability Negotiation

The IrDA OBEX Protocol profile, as specified in [MS-IRDA], does not introduce any new versioning issues. This profile is based on version 1.0 of the IrDA OBEX Protocol.

1.8   Vendor-Extensible Fields

Portions of the IrDA OBEX Protocol profile, as specified in [MS-IRDA], use Win32 error codes. These values are taken from the error number space defined in [MS-ERREF]. Vendors SHOULD reuse those values with their indicated meaning. Choosing any other value runs the risk of a collision in the future.

Section 3.1.5.1 describes how a CONNECT message that contains a WHO header is parsed. The WHO header used in this profile contains the **Windows UUID**, which is b9c7fd98-e5f8-11d1-bfce-0000f8753890.

Vendors who want to receive Win32 error codes (in addition to the IrOBEX error codes as specified in [IROBEX] section 3.2.1), using the Win32 Error Message header as specified in section 2.2.1.1, MUST use the above specific **UUID** in a WHO header. The effect of using this UUID in a WHO header is specified in sections 3.1.5 and 3.2.5.

1.9   Standards Assignments

There are no standards assignments other than what is specified in [IROBEX] section 6.

2   Messages

2.1   Transport

All IrOBEX messages are transported over TinyTP, as specified in [IROBEX] section 1.4.1.

2.2   Message Syntax

The message syntax remains unchanged and is as specified in [IROBEX] sections 3.1 and 3.2.
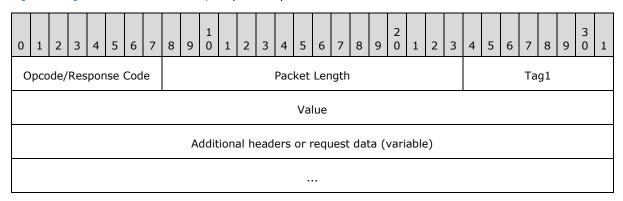
2.2.1   Header Types

Information about how a custom IrOBEX header can be constructed and used is specified in [IROBEX] sections 2.1 and 2.2.12. The custom header used in this protocol profile (section 3) is specified in section 2.2.1.1.

Beyond this, the header types and syntax remain unchanged and is as specified in [IROBEX] section 2.1.

2.2.1.1   Win32 Error Message Header

The custom IrOBEX header "Win32 Error Message", referred to in the rest of this document as the WIN32ERR header, is defined following the semantics specified in [IROBEX] section 2.2.12. The WIN32ERR header can be part of both a request message and a response message,<1> as specified in [IROBEX] sections 3.1 and 3.2, respectively.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Opcode/Response Code								Packet Length																Tag1							
Value																															
Additional headers or request data (variable)																															
...																															

Opcode/Response Code (1 byte): This value can be used as an opcode (in the request message) or as a response code (in the response message), and it defines the IrOBEX operation associated with this packet, as defined in [IROBEX] section 3.3. If this message is a response message, as defined in [IROBEX] section 3.2, the response code value MUST be taken from [IROBEX] section 3.2.1.

Packet Length (2 bytes): Describes the length (in bytes) of the entire packet including the opcode, packet length, all optional headers, and data.<2>More information is specified in [IROBEX] section 3.1.

Tag1 (1 byte): Describes the implementation-defined header identifier. The value for this field is 0xF0. This value is the bitwise OR of 0x30 and 0xC0. This signifies that the header identifier is "user-defined" (0x30) and that the length of the **Value** field is 4 bytes (0xC0). The values 0x30 and 0xC0 and the bitwise OR operation used to arrive at the final value of 0xF0 for this field are specified in [IROBEX] section 2.1.

Release: Friday, February 4, 2011

Value (4 bytes): A 4-byte value containing a Win32 error code as specified in section 1.8.

Additional headers or request data (variable): This variable length segment contains the rest of the IrOBEX message, as specified in [IROBEX] sections 3.1 and 3.2.

2.2.2 Message Types

The message types and syntax remain unchanged and are as specified in [IROBEX] section 3.

3   Protocol Details

The protocol details for both client and server are specified in [IROBEX]. The purpose of this section is to provide a context for implementation-specific notes about the client and server sides of the IrDA OBEX Protocol profile, as specified in [MS-IRDA].

3.1   Server Details

3.1.1   Abstract Data Model

No state is necessary other than that specified in [IROBEX] section 2.

3.1.2   Timers

No new timers are required beyond those in the base protocol, as specified in [IROBEX] section 3.4.

3.1.3   Initialization

No initialization is necessary other than that specified in [IROBEX].<3>.

3.1.4   Higher-Layer Triggered Events

No higher-layer triggered events are required other than those specified in [IROBEX].

3.1.5   Processing Events and Sequencing Rules

Message processing events MUST remain the same as specified in [IROBEX] section 3, except as described in this section.<4>

3.1.5.1   Receiving a CONNECT Message

CONNECT messages MUST be parsed as specified in [IROBEX] section 3.3.1. Optional headers MAY be parsed as specified in [IROBEX] section 2.1.<5> As a result of receiving a CONNECT message, the server MUST respond with a CONNECT Response message as specified in section 3.1.5.2.<6>

[IROBEX] section 2.2.7 asserts that a TARGET header MAY be used in conjunction with a WHO header. This profile does not use or rely on the TARGET header and instead relies only on the WHO header for identification of the IrOBEX client type.

3.1.5.2   Sending a CONNECT Response Message

A CONNECT Response message MUST be sent as specified in [IROBEX] section 3.3.1.8. If the CONNECT message contained a WHO header carrying a Windows UUID, the CONNECT Response message also contains a WHO header carrying the Windows UUID. The WIN32ERR header, as specified in section 2.2.1.1, is appended to the CONNECT Response message.

3.1.5.3   Receiving a PUT Message

A PUT message MUST be handled as specified in [IROBEX] section 3.3.3. Optional headers MAY be parsed as specified in [IROBEX] section 2.1.

[IROBEX] section 3.3.1.10 states that "...IrOBEX implementations MAY choose to accept PUT and GET operations without first requiring a CONNECT operation by assuming default values for the

connection parameters." This profile does not support acceptance of PUT operations without the required CONNECT operation.<7>

3.1.5.4 Sending a PUT Response Message

A PUT Response message MUST be sent as specified in [IROBEX] section 3.3.3.2. When the PUT Response returns an IrOBEX error response code, the message is processed as follows: If the PUT Response message was preceded by a CONNECT - CONNECT Response exchange containing a Windows UUID in a WHO header, then the PUT Response message will contain the WIN32ERR header (section 2.2.1.1).

3.1.5.5 Receiving a GET Message

This profile does not support processing of GET messages. Implementations of this profile discard the GET message by responding with a "Not implemented" IrOBEX response code (0xD1), as specified in section 3.2.1.

3.1.5.6 Receiving a SETPATH Message

A SETPATH message MUST be handled as specified in [IROBEX] section 3.3.6.

3.1.5.7 Sending a SETPATH Response Message

A SETPATH Response message MUST be sent as specified in [IROBEX] section 3.3.6. If the SETPATH message was preceded by a CONNECT - CONNECT Response exchange that contained a WHO header carrying the Windows UUID, this profile requires the WIN32ERR header (as specified in section 2.2.1.1) to be appended to the SETPATH Response message.

3.1.6 Timer Events

No new timer events are required beyond those in the base protocol, as specified in [IROBEX] section 3.4.

3.1.7 Other Local Events

None.

3.2 Client Details

3.2.1 Abstract Data Model

Filenames used in this protocol MUST be limited to 260 characters or less.

No other state is necessary other than that specified in [IROBEX] section 2.

3.2.2 Timers

No new timers are required beyond those in the base protocol, as specified in [IROBEX] section 3.4.

3.2.3 Initialization

No initialization is necessary other than that specified in [IROBEX].

Although not explicitly stated in [IROBEX], a client that wants to establish a TinyTP connection to be used by the IrDA OBEX protocol profile MUST perform an IAS **GetValueByClass** call on the class name "OBEX" or "OBEX:IrXfer", attribute "IrDA:TinyTP:LsapSel", as specified in [IRLMP] section 4.2.4. The client MUST initiate the TinyTP connection to the **Link Service Access Point Selector (LSAP-SEL)** value returned by the server, as specified in [IRTTP] section 2.2.1.

3.2.4 Higher-Layer Triggered Events

No higher-layer triggered events are required other than those specified in [IROBEX].

3.2.5 Processing Events and Sequencing Rules

Message processing events remain the same as specified in [IROBEX] section 3, except as described in this section.

3.2.5.1 Sending a CONNECT Message

The CONNECT message MUST be sent as specified in [IROBEX] section 3.3.1. Optional headers MAY be sent as specified in [IROBEX] section 2.1.

This profile uses the following values and optional headers in a CONNECT message:

▪ Maximum IrOBEX packet length = 32,672 bytes.

▪ WHO header carrying the Windows UUID as defined in section 1.8.

3.2.5.2 Sending a PUT Message

The PUT message MUST be sent as specified in [IROBEX] section 3.3.3. Optional headers MAY be sent as specified in [IROBEX] section 2.1.

This profile sends the following optional headers in a PUT message:

▪ NAME header

▪ LENGTH header

▪ TIME header: Microsoft Windows® uses [ISO-8601] time format as specified in [IROBEX] section 2.2.5.

3.2.5.3 Receiving a PUT Response Message

The PUT Response message MUST be handled as specified in [IROBEX] section 3.3.3.2.

If the PUT Response message was preceded by a CONNECT - CONNECT Response exchange that contained a Windows UUID in a WHO header, the PUT Response message will contain the WIN32ERR header (section 2.2.1.1) if the PUT Response also contains an IrOBEX error response code. Implementations of this profile MUST ABORT the transfer as defined in [IROBEX] if error codes that are not equal to zero are present in the WIN32ERR header.

3.2.5.4 Receiving a SETPATH Response Message

A SETPATH Response message MUST be handled as specified in [IROBEX] section 3.3.6.

3.2.6   Timer Events

No new timer events are required beyond those in the base protocol as specified in [IROBEX] section 3.4.

3.2.7   Other Local Events

None.

4 Protocol Examples

Protocol examples are specified in [IROBEX] section 7.

5 Security

5.1 Security Considerations for Implementers

This protocol profile does not implement any security function specified in [IROBEX]. As a result, devices attempting to interoperate with the protocol profile MUST NOT implement the authentication challenge as specified in [IROBEX] section 2.2.13.

Caution should be exercised when using this protocol profile. The mandatory physical proximity of 1 meter and line-of-sight positioning between the IrOBEX devices mitigates the potential security issues.

Protocol implementers SHOULD consider allowing users to turn off the functionality provided by this protocol profile.

5.2 Index of Security Parameters

None.

6   Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Windows® 98 operating system

- Microsoft Windows® Millennium Edition operating system

- Microsoft Windows NT® operating system

- Microsoft Windows® 2000 operating system

- Windows® XP operating system

- Windows Server® 2003 operating system

- Windows Vista® operating system

- Windows® 7 operating system

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 2.2.1.1: The WIN32ERR header is only sent when condition 1 and at least one of conditions 2–9 are met:

1. The NT5 dialect of the protocol is in use (and not the Win95 dialect).

2. A CONNECT, PUT, or SETPATH request is received and both of the following conditions are true:

    - The request has one or more of the following headers: NAME, LENGTH, TIME, BODY, BODY END, WHO, and WIN32ERR.

    - The value in the header is invalid or is not formatted correctly. For instance, the (nonzero) value length is less than what is expected for that header, or the file name is invalid.

      If a file name is invalid—for instance, the file name is longer than 260 characters—Microsoft implementations will always return an error.

3. A CONNECT message is received and any of the following conditions are true:

    - The HKCU\Control Panel\Infrared\File Transfer\Allow Send registry value is set to 0.

    - The system is going to sleep.

4. A CONNECT request is received and there is an error while creating a base file reception directory that does not already exist.

5. A PUT or SETPATH request is received and any of the following conditions are true:

- There is an error while creating the file or directory; for instance, access is denied.

  The user denies permission to receive the file.

  The user denies permission to create the directory through the UI.

6. A PUT request is received and there is an error while trying to write the data to the file; for instance, the disk is full.

7. A PUT request is being sent and the transfer is canceled.

8. There is an error reading the file to be sent or there is an error during the transmission (sending) of the file data.

9. Data is being received and either the transfer is canceled or a transmission error occurs.

10. If the incoming CONNECT/PUT/SETPATH request contains a WIN32ERR header with a nonzero error code, a WIN32ERR header will be sent out in the response with the same code.

<2> Section 2.2.1.1: Certain Microsoft implementations of this protocol behave differently when processing this header.

In some cases, the packet length field does not include the length of the WIN32ERR header even though the WIN32ERR header is sent. This behavior occurs in the following four cases listed in the previous Microsoft behavior note: 2 (except for receiving a CONNECT message), 5, 6, and 9.

In addition, for cases 7 and 8, the length will be wrong if the Win95 dialect of the protocol is used. The packet length will include the length of the WIN32ERR header even though the header is not sent out in the Win95 dialect. This behavior is consistent in all implementations of the Win95 dialect. The following versions of Windows are capable of using the Win95 dialect of this profile:

- Windows 98

- Windows Millennium Edition

- Windows NT

- Windows 2000

- Windows XP

- Windows Server 2003

- Windows Vista

- Windows 7

<3> Section 3.1.3: At initialization time, Windows registers two class names for the IrOBEX service in the IAS store: OBEX and OBEX:IrXfer as specified in [IROBEX] section 6.1. There is no difference in behavior by the server irrespective of which class name that the client uses to connect to the server.

<4> Section 3.1.5: [IROBEX] section 3 does not explicitly state that any headers need be supported.

Unless otherwise stated, Windows:

1. Discards all headers it receives.

2. Does not include any headers in messages that it sends over the link.

<5> Section 3.1.5.1: Windows parses the following optional headers as part of a CONNECT message as specified in [IROBEX] section 2.1.

1. NAME header

2. LENGTH header

3. TIME header

   Note Both [ISO-8601] and UNIX time formats are parsed.

4. WHO Header

<6> Section 3.1.5.1: A device that relies on authenticating the server will not interoperate with the Windows implementation of the IrDA OBEX protocol profile, as specified in [MS-IRDA], because the authentication header in a CONNECT message is discarded.

<7> Section 3.1.5.3: Windows implementations parse the following optional headers as part of a PUT message as specified in [IROBEX] section 2.1.

1. NAME header

2. LENGTH header

3. TIME header

   Note Both [ISO-8601] and UNIX time formats are parsed.

7 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

8 Index