

[MS-PPPI]:

PPP Over IrDA Dialup Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft [Open Specification Promise](#) or the [Community Promise](#). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications do not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments you are free to take advantage of them. Certain Open Specifications are intended for use in conjunction with publicly available standard specifications and network programming art, and assumes that the reader either is familiar with the aforementioned material or has immediate access to it.

Revision Summary

Date	Revision History	Revision Class	Comments
5/11/2007	0.1		Version 0.1 release
8/10/2007	1.0	Major	Updated and revised the technical content.
9/28/2007	1.0.1	Editorial	Changed language and formatting in the technical content.
10/23/2007	1.0.2	Editorial	Changed language and formatting in the technical content.
11/30/2007	1.0.3	Editorial	Changed language and formatting in the technical content.
1/25/2008	1.0.4	Editorial	Changed language and formatting in the technical content.
3/14/2008	1.0.5	Editorial	Changed language and formatting in the technical content.
5/16/2008	1.0.6	Editorial	Changed language and formatting in the technical content.
6/20/2008	1.0.7	Editorial	Changed language and formatting in the technical content.
7/25/2008	1.0.8	Editorial	Changed language and formatting in the technical content.
8/29/2008	2.0	Major	Updated and revised the technical content.
10/24/2008	3.0	Major	Updated and revised the technical content.
12/5/2008	4.0	Major	Updated and revised the technical content.
1/16/2009	4.0.1	Editorial	Changed language and formatting in the technical content.
2/27/2009	4.0.2	Editorial	Changed language and formatting in the technical content.
4/10/2009	4.0.3	Editorial	Changed language and formatting in the technical content.
5/22/2009	4.0.4	Editorial	Changed language and formatting in the technical content.
7/2/2009	4.0.5	Editorial	Changed language and formatting in the technical content.
8/14/2009	4.0.6	Editorial	Changed language and formatting in the technical content.
9/25/2009	4.1	Minor	Clarified the meaning of the technical content.
11/6/2009	4.1.1	Editorial	Changed language and formatting in the technical content.
12/18/2009	4.1.2	Editorial	Changed language and formatting in the technical content.
1/29/2010	4.1.3	Editorial	Changed language and formatting in the technical content.
3/12/2010	4.1.4	Editorial	Changed language and formatting in the technical content.
4/23/2010	4.1.5	Editorial	Changed language and formatting in the technical content.
6/4/2010	4.1.6	Editorial	Changed language and formatting in the technical content.
7/16/2010	4.1.6	None	No changes to the meaning, language, or formatting of the technical content.
8/27/2010	4.1.6	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2010	4.1.6	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
11/19/2010	4.1.6	None	No changes to the meaning, language, or formatting of the technical content.
1/7/2011	5.0	Major	Updated and revised the technical content.
2/11/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
3/25/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
5/6/2011	5.0	None	No changes to the meaning, language, or formatting of the technical content.
6/17/2011	5.1	Minor	Clarified the meaning of the technical content.
9/23/2011	5.1	None	No changes to the meaning, language, or formatting of the technical content.
12/16/2011	6.0	Major	Updated and revised the technical content.
3/30/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
7/12/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
10/25/2012	6.0	None	No changes to the meaning, language, or formatting of the technical content.
1/31/2013	6.0	None	No changes to the meaning, language, or formatting of the technical content.
8/8/2013	7.0	Major	Updated and revised the technical content.
11/14/2013	7.0	None	No changes to the meaning, language, or formatting of the technical content.
2/13/2014	7.0	None	No changes to the meaning, language, or formatting of the technical content.
5/15/2014	7.0	None	No changes to the meaning, language, or formatting of the technical content.
6/30/2015	8.0	Major	Significantly changed the technical content.

Table of Contents

1	Introduction	6
1.1	Glossary	6
1.2	References	7
1.2.1	Normative References	7
1.2.2	Informative References	8
1.3	Overview	8
1.4	Relationship to Other Protocols	8
1.5	Prerequisites/Preconditions	10
1.6	Applicability Statement	10
1.7	Versioning and Capability Negotiation	10
1.8	Vendor-Extensible Fields	10
1.9	Standards Assignments.....	10
2	Messages.....	11
2.1	Transport.....	11
2.2	Message Syntax.....	11
2.2.1	IrDial Message Formats	11
2.2.1.1	General Formatting Rules.....	11
2.2.1.2	Dial Message	11
2.2.1.3	Dial Response Message.....	12
2.2.1.4	Hook Message	12
2.2.1.5	Hook Response Message	13
2.2.2	Data Message	13
3	Protocol Details	15
3.1	Common Details	15
3.1.1	Abstract Data Model.....	15
3.1.2	Timers	15
3.1.3	Initialization.....	15
3.1.4	Higher-Layer Triggered Events	15
3.1.5	Processing Events and Sequencing Rules	15
3.1.6	Timer Events.....	16
3.1.7	Other Local Events.....	16
3.2	Server-Specific Details.....	16
3.2.1	Abstract Data Model.....	16
3.2.2	Timers	16
3.2.3	Initialization.....	16
3.2.4	Higher-Layer Triggered Events	16
3.2.5	Processing Events and Sequencing Rules	16
3.2.5.1	Receiving a Dial Message	16
3.2.5.2	Receiving a Hook Message	17
3.2.5.3	Receiving a Data Message.....	17
3.2.6	Timer Events.....	17
3.2.7	Other Local Events.....	17
3.3	Client-Specific Details	17
3.3.1	Abstract Data Model.....	17
3.3.2	Timers	17
3.3.3	Initialization.....	17
3.3.4	Higher-Layer Triggered Events	17
3.3.5	Processing Events and Sequencing Rules	18
3.3.5.1	Sending a Dial Message	18
3.3.5.2	Receiving a Dial Response Message.....	18
3.3.5.3	Sending a Hook Message	18
3.3.5.4	Receiving a Hook Response Message	18
3.3.5.5	Receiving an Echo Message	18

3.3.5.6	Sending a Data Message	18
3.3.5.7	Receiving a Data Message	18
3.3.6	Timer Events.....	18
3.3.7	Other Local Events.....	19
4	Protocol Examples	20
4.1	Connection Setup and Data Exchange.....	20
5	Security	22
5.1	Security Considerations for Implementers	22
5.2	Index of Security Parameters	22
6	Appendix A: Product Behavior	23
7	Change Tracking.....	24
8	Index.....	26

1 Introduction

This specification defines the PPP Over IrDA Dialup Protocol, and includes contributions from Microsoft, Ericsson, and Nokia. The PPP Over IrDA Dialup Protocol is a deprecated protocol that specifies how to initialize and use a modem over an infrared link. This protocol enables the scenario in which a computer with infrared capabilities has network access by using a modem via an infrared link.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [\[RFC2119\]](#). Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

<CR>: ASCII character "Enter" (decimal symbol # 13), as specified in [\[RFC1345\]](#).

<LF>: ASCII character "Line feed" (decimal symbol # 10), as specified in [\[RFC1345\]](#).

AT command set: The Attention Code (AT) command set is used to issue one or more commands to a modem device. The AT prefix signals the modem that one or more commands are to follow, as specified in [\[V25TER\]](#).

client: A computer on which the remote procedure call (RPC) client is executing.

Data-Carrying TinyTP PDU: The **TinyTP** Protocol distinguishes between PDUs that carry data and PDUs that are used during **TinyTP** connection establishment. The "Data-Carrying" prefix denotes that the PDU being discussed is used for data exchange and not **TinyTP** connection establishment.

escape sequence: A series of three consecutive characters (+++) sent to the infrared modem that causes it to exit online data mode and enter online command mode.

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 gain information 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: The Infrared Data Association, often referred to as **IrDA**, is a nonprofit organization whose goal is to develop globally adopted specifications for infrared wireless communication.

IrDial: A component of the PPP Over IrDA Dialup Protocol that implements modem commands.

IrNet: A component of the PPP Over IrDA Dialup Protocol that implements a data path pass-through between the PPP over HDLC-like framing and **TinyTP**.

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 0x00–0x7F. With the exception of the special LSAP-SEL values 0x00 (LM-IAS), 0x70 (Connectionless Data service), 0x71–0x7E (reserved), and 0x7F (reserved for broadcast and currently not implemented), the assignment of LSAP-SEL values is arbitrary. See [\[IRLMP\]](#) section 3.1.2 for more details.

SDU: A **TinyTP** Client Service Data Unit as specified in [\[IRTPP\]](#) section 2.2.3. The **SDU** defines the maximum size of a data block that can be exchanged between two **TinyTP clients**. For example, the IrDial Protocol, which is implemented as **TinyTP client**, may negotiate an **SDU** size of 10 KB. The **TinyTP** Protocol segments and reassembles this **SDU** in multiple PDUs that are transmitted over the infrared link.

server: A computer on which the remote procedure call (RPC) server is executing.

TinyTP: Infrared Data Association Tiny Transport Protocol, as specified in [IRTTTP].

TinyTP PDU: A **TinyTP** Protocol Data Unit as specified in [IRTTTP] section 2.3.1. The maximum size of the PDU must be in the range of 64–2048 bytes. See [IRLAP] section 6.6.5 and [IRTTTP] section 2.3.1 for more details.

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

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the [Errata](#).

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.

[IRLAP] Infrared Data Association, "IrDA Link Access Protocol v.1.1", June 1996, <http://irdajp.info/specifications.html>

Note There is a charge to download the specification.

[IRLMP] Infrared Data Association, "IrDA Link Management Protocol v1.1", January 1996, <http://irdajp.info/specifications.html>

Note There is a charge to download the specification.

[IRTTTP] Infrared Data Association, "IrDA Tiny TP v1.1", October 1996, <http://irdajp.info/specifications.html>

Note There is a charge to download the specification.

[RFC1345] Simonsen, K., "Character Mnemonics and Character Sets", RFC 1345, June 1992, <http://www.ietf.org/rfc/rfc1345.txt>

[RFC1661] Simpson, W., Ed., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994, <http://www.ietf.org/rfc/rfc1661.txt>

[RFC1662] Simpson, W., Ed., "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994, <http://www.ietf.org/rfc/rfc1662.txt>

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

[V25TER] ITU-T, "Serial Asynchronous Automatic Dialling and Control", Recommendation V.25ter, July 1997, <http://www.itu.int/rec/T-REC-V.25ter-199707-S/en>

Note There is a charge to download the specification.

1.2.2 Informative References

[IRCOM] Infrared Data Association, "'IrCOMM': Serial and Parallel Port Emulation over IR (Wire Replacement) v.1.0", November 1995, <http://irdajp.info/specifications.html>

Note There is a charge to download the specification.

[IRPLS] Infrared Data Association, "IrDA Physical Layer Specification v.1.4", May 2001, <http://irdajp.info/specifications.html>

Note There is a charge to download the specification.

[MSDN-IRNET] Microsoft Corporation, "Infrared Network (IrNET)", <http://msdn.microsoft.com/en-us/library/ms817914.aspx>

1.3 Overview

The PPP Over IrDA Dialup Protocol specifies how to initialize and exchange data with a modem over an infrared link. The PPP Over IrDA Dialup Protocol has a **client** role and a **server** role:

- The server is the modem that has an infrared device and implements the PPP Over IrDA Dialup Protocol. In addition, the server must implement the family of **IrDA** protocols, specifically the IrDA TinyTP v1.1 Protocol [IRTTP], IrDA LinkManagement Protocol v1.1 [IRLMP], IrDA LinkAccess Protocol v1.1 [IRLAP], and IrDA Physical Layer Specification [IRPLS].
- The client is the computer that has an infrared device and implements the PPP Over IrDA Dialup Protocol. In addition, the client must also implement the family of IrDA protocols—again, the IrDA TinyTP v1.1 Protocol [IRTTP], IrDA LinkManagement Protocol v1.1 [IRLMP], IrDA LinkAccess Protocol v1.1 [IRLAP], and IrDA Physical Layer Specification [IRPLS].

To initialize the modem, the PPP Over IrDA Dialup Protocol uses **IrDial** messages as described in section 2.2. Briefly, the IrDial messages are AT dial commands as specified in [V25TER], which are exchanged between the computer and the modem via the infrared link.

Each IrDial message is echoed back to the client by the server. Note that the echoing of messages back to the client is applicable only while the PPP Over IrDA Dialup Protocol is initializing the modem. When the modem is used for data exchange, there are no message echoes.

After initializing the modem, the PPP Over IrDA Dialup Protocol can exchange data with it. To exchange data with the modem, this protocol uses the point-to-point protocol (PPP) in HDLC-like framing, as specified in [RFC1662], to encapsulate the higher-layer network data. The PPP in HDLC-like framing frames are segmented (if necessary) and framed in **TinyTP** Service Data Units (**SDUs**) as specified in [IRTTP].

1.4 Relationship to Other Protocols

A preferred alternative to the PPP Over IrDA Dialup Protocol is the IrDA Infrared Communications Protocol [IRCOM], which provides emulation of serial and parallel ports over the IrDA TinyTP v1.1 Protocol [IRTTP], IrDA LinkManagement Protocol v1.1 [IRLMP], and IrDA LinkAccess Protocol v1.1 [IRLAP] protocol stack. Because the IrDA Infrared Communications Protocol provides serial emulation, the computer can initialize and use the modem over an infrared link by simply reading and writing to a serial port.

The PPP Over IrDA Dialup Protocol depends on the Infrared Network (**IrNet**) Protocol [MSDN-IRNET] for the data path pass-through of PPP in HDLC-like framing frames to the TinyTP layer. The following figure depicts the PPP Over IrDA Dialup Protocol and its relationship to the other higher-layer and lower-layer protocols mentioned in this specification.

In the Figure 1, the term IRPHY (IRDA Physical Layer) refers to the physical infrared hardware device and not to a software protocol.

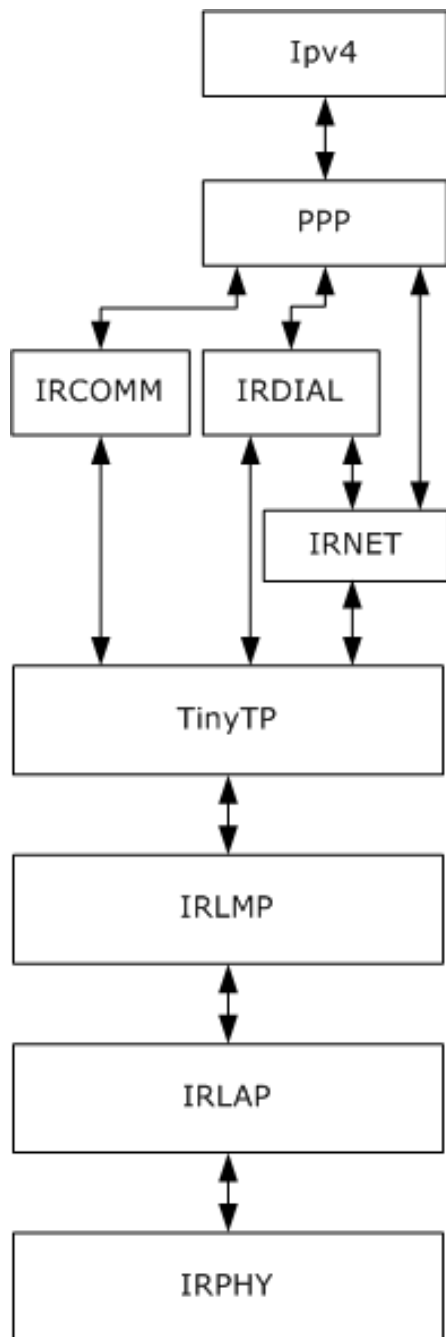


Figure 1: Protocol stack diagram showing the PPP Over IrDA Dialup Protocol

The PPP Over IrDA Dialup Protocol consists of the IrDial and IrNet components and their relationship to other protocols. Both IrDial and IrNet use TinyTP as their transport protocol.

Note The PPP component includes both [\[RFC1661\]](#) and [\[RFC1662\]](#) encapsulations.

1.5 Prerequisites/Preconditions

Prior to using the PPP Over IrDA Dialup Protocol, the client and the server must have an established TinyTP connection. See sections [3.2.3](#) and [3.3.3](#) for further details.

1.6 Applicability Statement

The PPP Over IrDA Dialup Protocol is deprecated and should not be used. Instead, the IrDA Infrared Communications Protocol (IrCOMM), as defined in [\[IRCOM\]](#), should be used for modem initialization and data transfer over an infrared link.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The PPP Over IrDA Dialup Protocol uses IrDA TinyTP, as specified in [IRTTP], as a transport protocol for initialization and data exchange with the modem. Note that for data exchange with the modem, [RFC1662] frames are transported directly via IrDA TinyTP, as specified in [IRTTP].

2.2 Message Syntax

2.2.1 IrDial Message Formats

As specified in [IRTTP] section 2.2.1, the maximum size of a single **Data-Carrying TinyTP PDUs** packet for a given connection is negotiated by the IrDA Link Access Protocol, as specified in [IRLAP].

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Delta Credit							M	IrDial Message (variable)																							
...																															

Delta Credit (7 bits): The **Delta Credit** field MUST specify the number (0-127) of additional Data-Carrying TinyTP PDUs that can be sent in the reverse direction, as specified in [IRTTP] section 2.3.1.

M (1 bit): The **M** field MUST be set to zero, as specified in [IRTTP] section 2.3.1, to denote that this is a single **TinyTP PDU**.

IrDial Message (variable): Each IrDial message is a sequence of ASCII characters and MUST conform to the definitions in section 2.2.1. Messages of this form are often referred to as **AT Command Sets**. The IrDial message MUST NOT exceed the maximum size of a TinyTP PDU, which is negotiated by the IrDA Link Access Protocol (as specified in [IRLAP]). For further details, see [IRTTP] section 2.2.1.

2.2.1.1 General Formatting Rules

- Each IrDial message MUST end with **<CR>**.
- Each IrDial message that is a response to a previous IrDial message MUST be in the format **<CR><LF>Response<CR><LF>**.

2.2.1.2 Dial Message

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
ASCII - A							ASCII - T							ASCII - D							Number (variable)										
...																															
ASCII - CR																															

ASCII - A (1 byte): The ASCII encoding for the "A" character, 0x41, as specified in [\[RFC1345\]](#).

ASCII - T (1 byte): The ASCII encoding for the "T" character, 0x54, as specified in [\[RFC1345\]](#).

ASCII - D (1 byte): The ASCII encoding for the "D" character, 0x44, as specified in [\[RFC1345\]](#).

Number (variable): A sequence of ASCII encoded numbers, as specified in [\[RFC1345\]](#). This number denotes the phone number that the modem will dial.

ASCII - CR (1 byte): The ASCII encoding for the carriage return character, 0x0d, as specified in [\[RFC1345\]](#).

2.2.1.3 Dial Response Message

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
ASCII - CR										ASCII - LF										Response (variable)													
...																																	
ASCII - CR										ASCII - LF																							

ASCII - CR (1 byte): The ASCII encoding for the carriage return character, 0x0d, as specified in [\[RFC1345\]](#).

ASCII - LF (1 byte): The ASCII encoding for the line feed character, 0x0a, as specified in [\[RFC1345\]](#).

Response (variable): The **Response** field MUST be an ASCII encoded string. The **Response** field MUST have one of the following values.

Value	Meaning
"CONNECT <speed>"	Data connection established at the rate given in <speed>.
"NO CARRIER"	Unable to establish a connection, or the connection attempt was aborted.
"ERROR"	An unexpected error occurred while trying to establish the connection.
"NO DIALTONE"	The server or modem device (for instance, a mobile phone) is being used for a voice call or has no access to (is not within coverage of) the network.
"BUSY"	The phone number called is engaged.

ASCII - CR (1 byte): The ASCII encoding for the carriage return character, 0x0d, as specified in [\[RFC1345\]](#).

ASCII - LF (1 byte): The ASCII encoding for the line feed character, 0x0a, as specified in [\[RFC1345\]](#).

2.2.1.4 Hook Message

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1								
ASCII - +										ASCII - +										ASCII - +										ASCII - A									
ASCII - T										ASCII - H										ASCII - CR																			

ASCII - + (1 byte): The first three fields specify three occurrences of the ASCII encoding for the '+' (plus sign) character, 0x2b, as specified in [\[RFC1345\]](#). This string of '+' (plus sign) characters is often referred to as an **escape sequence**.

ASCII - A (1 byte): The ASCII encoding for the 'A' character, 0x41, as specified in [\[RFC1345\]](#).

ASCII - T (1 byte): The ASCII encoding for the 'T' character, 0x54, as specified in [\[RFC1345\]](#).

ASCII - H (1 byte): The ASCII encoding for the 'H' character, 0x48, as specified in [\[RFC1345\]](#).

ASCII - CR (1 byte): The ASCII encoding for the carriage return character, 0x0d, as specified in [\[RFC1345\]](#).

2.2.1.5 Hook Response Message

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	
ASCII - CR									ASCII - LF									Response (variable)														
...																																
ASCII - CR									ASCII - LF																							

ASCII - CR (1 byte): The ASCII encoding for the carriage return character, 0x0d, as specified in [\[RFC1345\]](#).

ASCII - LF (1 byte): The ASCII encoding for the line feed character, 0x0a, as specified in [\[RFC1345\]](#).

Response (variable): The **Response** field MUST be an ASCII-encoded string. The **Response** field MUST have one of the following values.

Value	Meaning
"OK"	Modem is already in "Offline Command Mode" state.
"NO CARRIER"	Connection terminated.
"ERROR"	Unexpected error.

ASCII - CR (1 byte): The ASCII encoding for the carriage return character, 0x0d, as specified in [\[RFC1345\]](#).

ASCII - LF (1 byte): The ASCII encoding for the line feed character, 0x0a, as specified in [\[RFC1345\]](#).

2.2.2 Data Message

The PPP Over IrDA Dialup Protocol relies on the Infrared Network (IrNET) Protocol [\[MSDN-IRNET\]](#) for the pass-through of data messages between the PPP over HDLC-like framing [\[RFC1662\]](#) and TinyTP [\[IRTTP\]](#), for the client and the server.

As specified in [\[IRTTP\]](#) section 2.2.1, the maximum size of a single Data-Carrying TinyTP PDUs packet for a given connection is negotiated by the IrDA Link Access Protocol, as specified in [\[IRLAP\]](#).

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
Delta Credit						M	Data Message (variable)																										
...																																	

Delta Credit (6 bits): The **Delta Credit** field MUST specify the number (0-63) of additional Data-Carrying TinyTP PDUs that can be sent in the reverse direction, as specified in [IRTP] section 2.3.1.

M (1 bit): The **M** field MUST be set to zero, as specified in [IRTP] section 2.3.1, if this is a single TinyTP PDU. The **M** field MUST be set to 1 if it is part of, but not the last part of a multi-part TinyTP PDU. That is, if the **M** bit is set to 1, there are more parts of this **PDU** to follow.

Data Message (variable): The **Data** field MUST be ASCII-encoded. The IrDial message MUST NOT exceed the maximum size of a TinyTP PDU, which is negotiated by the IrDA Link Access Protocol (as specified in [IRLAP]). For further details, see [IRTP] section 2.2.1.

If the length of the **Data** field exceeds the maximum size of a single Data-Carrying TinyTP PDU packet, the TinyTP Protocol performs segmentation and reassembly (SAR) of the **Data** field.

3 Protocol Details

3.1 Common Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This specification does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this specification.

The following data model and states are common for the client role and the server role of the protocol:

Modem state: A data element that describes the current state of the modem device. The modem device can be in one of two states: offline command mode or online data mode.

Offline command mode state: In this state there is no call-up and the modem is accepting AT Commands via IrDial messages. This is the state in which all sessions are started.

Online data mode state: There is a call-up and the modem can receive and transmit data.

IrDial-TTP connection state: A data element that describes the current state of the TinyTP connection that is used for PPP Over IrDA Dialup Protocol communication. The server and the client can be either in an IrDial-TTP connection established state or in an IrDial-TTP connection not-established state.

3.1.2 Timers

This protocol includes the following timer:

IdleTimer: 12-second expiration timer.

Besides the aforementioned timer, the IrDA TinyTP Protocol [\[IRTTP\]](#) by which PPP Over IrDA Dialup Protocol messages are conveyed MAY have timers associated with it to achieve guaranteed and in-order delivery.

3.1.3 Initialization

The initially assumed values for the data elements defined in section [3.1.1](#) are as follows:

- Modem state: Offline command mode.
- IrDial-TTP connection state: IrDial-TTP connection not-established.

For more information, see sections [3.2.3](#) and [3.3.3](#).

3.1.4 Higher-Layer Triggered Events

For more information, see sections [3.2.4](#) and [3.3.4](#).

3.1.5 Processing Events and Sequencing Rules

For more information, see sections [3.2.5](#) and [3.3.5](#).

3.1.6 Timer Events

After the PPP Over IrDA Dialup Protocol is initialized as specified in sections [3.2.3](#) and [3.3.3](#), both the client and the server set the IdleTimer. The timer MUST be reset by the client and the server any time they send or receive a message.

When the timer fires, the PPP Over IrDA Dialup Protocol MUST be moved to the uninitialized state.

3.1.7 Other Local Events

None.

3.2 Server-Specific Details

The modem that implements the PPP Over IrDA Dialup Protocol on an infrared device MAY assume the server role.

3.2.1 Abstract Data Model

There is no additional server-specific data model. The data model in section [3.1.1](#) is assumed.

3.2.2 Timers

No timers, beyond that defined in section [3.1.2](#), are defined for the server role.

3.2.3 Initialization

The initialization of the server is a two-step sequence:

1. The server MUST register a service access point (SAP) with the **Information Access Service (IAS)**, as specified in [\[IRLMP\]](#) section 3.1.2, and wait for any client to connect to it. The service access point object contained in the IAS MUST have the class name IrModem as defined in [\[IRLMP\]](#) section 4.2.4. This allows other PPP Over IrDA Dialup Protocol clients to discover the PPP Over IrDA Dialup Protocol service on the server.
2. The modem device on the server is put in the offline command mode state.

3.2.4 Higher-Layer Triggered Events

A user can configure the device to become a PPP Over IrDA Dialup Protocol server. In the case of this event, the sequence described in section [3.2.3](#) is initiated.

3.2.5 Processing Events and Sequencing Rules

3.2.5.1 Receiving a Dial Message

When a [dial message](#) arrives, the server can be in one of two modes: offline command mode or online data mode.

If the server is in offline command mode when a dial message is received, the server responds by sending the same dial message. This response is called "echoing the message." Upon echoing the message, the server executes the AT Dial command sequence and sends a [dial response message](#) with the appropriate **Response** field string. If the dial response message **Response** field string value is "CONNECT <speed>", the server moves to the online command mode state; otherwise it remains in the offline command mode state.

If the server is in online data mode when a dial message is received, the server treats it as a regular [data message](#). That is, it does not execute the AT Dial command sequence and therefore takes no action.

3.2.5.2 Receiving a Hook Message

When a [hook message](#) arrives, the server can be in one of two modes: offline command mode or online data mode.

If a server is in offline command mode when a hook message is received, it responds by echoing the hook message followed by a [hook response message](#) with the **Response** field set to "OK". The server remains in the offline command mode state.

If a server is in online data mode when a hook message is received, it echoes the message and sends a hook response message with the **Response** field set to "NO CARRIER". As a result of this message being sent, the server moves from the online data mode state to offline command mode state.

3.2.5.3 Receiving a Data Message

When a [Data Message](#) arrives from the client, the server transmits the data toward the network as specified in [\[RFC1662\]](#) and section 2.2.2. When data arrives from the network, the server transmits a Data Message to the client as specified in [\[IRTPP\]](#) and section 2.2.2.

3.2.6 Timer Events

There are no timer-triggered events beyond those described in section [3.1.6](#).

3.2.7 Other Local Events

None.

3.3 Client-Specific Details

3.3.1 Abstract Data Model

In addition to the data model specified in section [3.1.1](#), the client has the following data element:

Phone number for modem to dial: A data element that contains the phone number that a client sends in the [dial message \(section 2.2.1.2\)](#).

3.3.2 Timers

No timers are defined for the client role, beyond the definitions provided in section [3.1.2](#).

3.3.3 Initialization

A client that wants to establish a TinyTP connection to be used by the PPP Over IrDA Dialup Protocol MUST perform an IAS GetValueByClass on the class name IrModem, 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 [\[IRTPP\]](#) section 2.2.1.

3.3.4 Higher-Layer Triggered Events

The user initializes the PPP Over IrDA Dialup Protocol as specified in section [3.3.3](#).

The user initiates a dialing sequence by passing the phone number for the modem to dial to the PPP Over IrDA Dialup Protocol.

3.3.5 Processing Events and Sequencing Rules

3.3.5.1 Sending a Dial Message

The [dial message](#) is sent to the server as specified in section 2.2.1.2.

3.3.5.2 Receiving a Dial Response Message

If the client is in online data mode when the [dial response message](#) arrives, the client treats it as a regular [data message](#) (that is, it does not parse the dial response message **Response** field and does not take any action).

If the client is in offline command mode when the dial response message arrives, the client extracts the value contained in the **Response** field and takes action based on the **Response** field.

If the **Response** field contains "CONNECT <speed>", the client changes its modem state from offline command mode to online data mode. The protocol passes the **Response** field to the higher-layer protocol, which results in the client starting to send data messages to the server, as specified in [\[RFC1661\]](#), to establish the PPP link layer.

If the **Response** field is "NO CARRIER", "ERROR", "NO DIALTONE", or "BUSY", the client modem state remains in offline command mode and it passes the **Response** field to the higher-layer protocol.

3.3.5.3 Sending a Hook Message

The client [hook message](#) is sent to the server as specified in section 2.2.1.4.

3.3.5.4 Receiving a Hook Response Message

When the [hook response message](#) arrives, the client extracts the **Response** field and passes it to a higher-layer protocol.

3.3.5.5 Receiving an Echo Message

When the client receives the echo of its own message from the server, the client discards it and waits for the response message. For example, when the client sends a [dial message](#), it receives an echo of the dial message, which it discards, and continues to wait for the arrival of a [dial response message](#).

3.3.5.6 Sending a Data Message

A client sends a data message, as specified in [\[RFC1662\]](#) section 4, encapsulated in one or more TinyTP [\[IRTP\]](#) SDUs.

3.3.5.7 Receiving a Data Message

When a client receives a data message, it is processed as specified in [\[RFC1662\]](#) section 4.

3.3.6 Timer Events

There are no timer-triggered events beyond those discussed in section [3.1.6](#).

3.3.7 Other Local Events

Upon a successful [dial/dial response message](#) exchange from the TinyTP [\[IRTP\]](#) layer and a dial response message that has a "Connect" response, the PPP Over IrDA Dialup Protocol indicates that the PPP Over IrDA Dialup Protocol link is ready for [data message](#) exchange by passing the "Connect" response to the PPP layer.

4 Protocol Examples

4.1 Connection Setup and Data Exchange

In the example illustrated in Figure 2, the user initiates the PPP Over IrDA Dialup Protocol connection by initializing the client-side protocol as specified in section [3.3.3](#). The user then passes the phone number to the client-side PPP Over IrDA Dialup Protocol, as specified in section [3.3.4](#). It is assumed that server-side initialization, as specified in section [3.2.3](#), has already taken place.

1. The client sends a [dial message](#), as specified in section 2.2.1.2, with the **Number** field containing "8001231234".
2. The server echoes the dial message back to the client as specified in section [3.2.5.1](#) (not shown in Figure 2).
3. The server sends the [dial response message](#) containing the connection speed at which it can operate, in the form <CR><LF>CONNECT 9600<CR><LF>, where 9600 indicates the negotiated speed between the two modems (one modem being the PPP Over IrDA Dialup Protocol server and the other modem being the device at number 800-123-1234 from step 1).
4. Once the connection is established and the client side of the PPP Over IrDA Dialup Protocol indicates to the PPP layer to send data, the client sends a [data message](#) to the server, as specified in section 2.2.2.
5. The server sends a data message to the client, which is delivered to the PPP Protocol as specified in [\[RFC1662\]](#).
6. The client sends a [hook message](#) to the server.
7. The server echoes the hook message back to the client as specified in section [3.2.5.2](#) (not shown in Figure 2).
8. The server sends the [hook response message](#) with the **Response** client field set to "NO CARRIER".

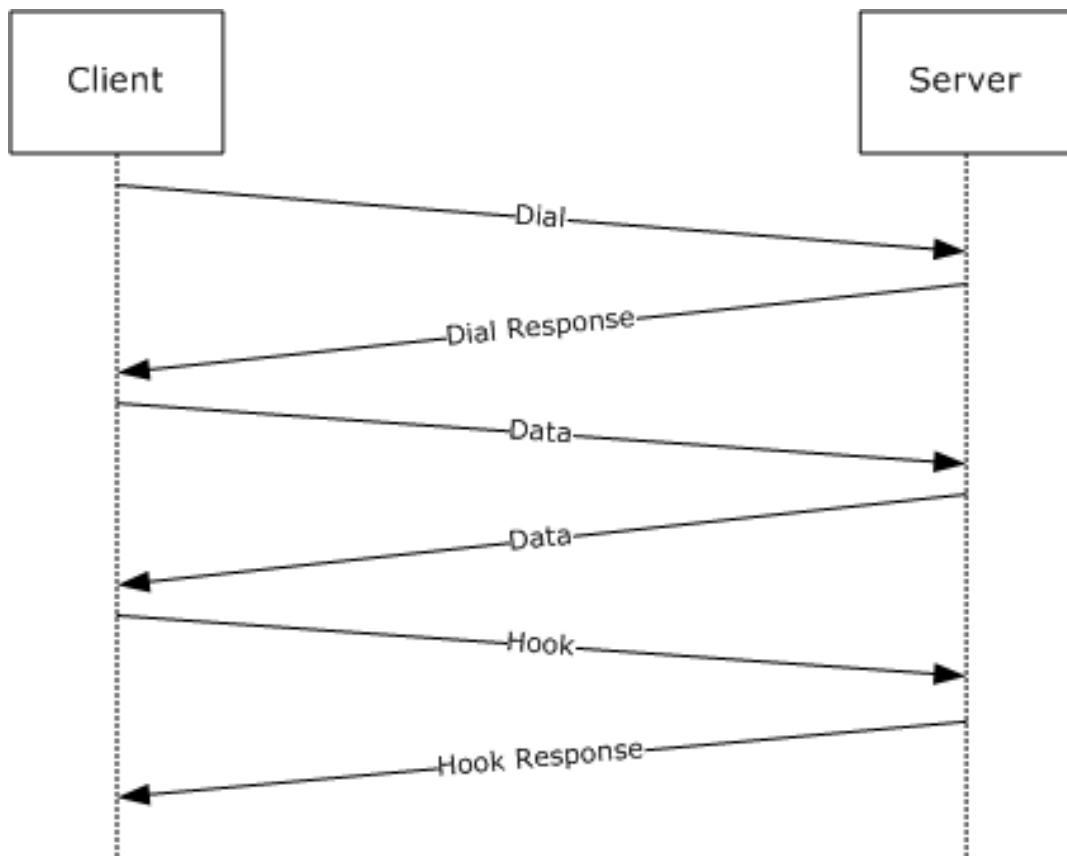


Figure 2: Example of PPP Over IrDA Dialup Protocol connection setup, data exchange, and connection teardown

5 Security

5.1 Security Considerations for Implementers

None.

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.

- Windows Server 2003 operating system
- Windows Vista operating system
- Windows 7 operating system
- Windows 8 operating system
- Windows 8.1 operating system
- Windows 10 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.

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements or functionality.
- The removal of a document from the documentation set.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **Editorial** means that the formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.
- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated**.

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
6 Appendix A: Product Behavior	Added Windows 10 to applicability list.	Y	Content update.

8 Index

A

Abstract data model
 client ([section 3.1.1](#) 15, [section 3.3.1](#) 17)
 server ([section 3.1.1](#) 15, [section 3.2.1](#) 16)
[Applicability](#) 10

C

[Capability negotiation](#) 10
[Change tracking](#) 24
Client
 abstract data model ([section 3.1.1](#) 15, [section 3.3.1](#) 17)
 higher-layer triggered events ([section 3.1.4](#) 15, [section 3.3.4](#) 17)
 initialization ([section 3.1.3](#) 15, [section 3.3.3](#) 17)
 local events ([section 3.1.7](#) 16, [section 3.3.7](#) 19)
 message processing ([section 3.1.5](#) 15, [section 3.3.5](#) 18)
 sequencing rules ([section 3.1.5](#) 15, [section 3.3.5](#) 18)
 timer events ([section 3.1.6](#) 16, [section 3.3.6](#) 18)
 timers ([section 3.1.2](#) 15, [section 3.3.2](#) 17)
[Connection setup and data exchange example](#) 20

D

Data message
 receiving ([section 3.2.5.3](#) 17, [section 3.3.5.7](#) 18)
 [sending](#) 18
[Data Message message](#) 13
Data model - abstract
 client ([section 3.1.1](#) 15, [section 3.3.1](#) 17)
 server ([section 3.1.1](#) 15, [section 3.2.1](#) 16)
[Data Message Syntax packet](#) 13
Dial message
 [receiving](#) 16
 [sending](#) 18
[Dial response message - receiving](#) 18
[Dial Message packet](#) 11
[Dial Response Message packet](#) 12

E

[Echo message - receiving](#) 18
[Example - connection setup and data exchange](#) 20

F

[Fields - vendor-extensible](#) 10
[Formatting rules](#) 11

G

[Glossary](#) 6

H

Higher-layer triggered events
 client ([section 3.1.4](#) 15, [section 3.3.4](#) 17)
 server ([section 3.1.4](#) 15, [section 3.2.4](#) 16)
Hook message
 [receiving](#) 17
 [sending](#) 18
[Hook response message - receiving](#) 18
[Hook Message packet](#) 12
[Hook Response Message packet](#) 13

I

[Implementer - security considerations](#) 22
[Index of security parameters](#) 22
[Informative references](#) 8
Initialization
 client ([section 3.1.3](#) 15, [section 3.3.3](#) 17)
 server ([section 3.1.3](#) 15, [section 3.2.3](#) 16)
[Introduction](#) 6
[IrDial message formats](#) 11
[IrDial Message Formats message](#) 11
[IrDial Message Syntax packet](#) 11

L

Local events
 client ([section 3.1.7](#) 16, [section 3.3.7](#) 19)
 server ([section 3.1.7](#) 16, [section 3.2.7](#) 17)

M

Message processing
 client ([section 3.1.5](#) 15, [section 3.3.5](#) 18)
 server ([section 3.1.5](#) 15, [section 3.2.5](#) 16)
Messages
 [Data Message](#) 13
 [IrDial Message Formats](#) 11
 [syntax](#) 11
 [transport](#) 11

N

[Normative references](#) 7

	Triggered events - higher-layer
	client (section 3.1.4 15, section 3.3.4 17)
	server (section 3.1.4 15, section 3.2.4 16)
O	
	Overview 8
	Overview (synopsis) 8
P	
	Parameters - security index 22
	Preconditions 10
	Prerequisites 10
	Product behavior 23
R	
	References 7
	informative 8
	normative 7
	Relationship to other protocols 8
S	
	Security
	implementer considerations 22
	parameter index 22
	Sequencing rules
	client (section 3.1.5 15, section 3.3.5 18)
	server (section 3.1.5 15, section 3.2.5 16)
	Server
	abstract data model (section 3.1.1 15, section 3.2.1 16)
	higher-layer triggered events (section 3.1.4 15, section 3.2.4 16)
	initialization (section 3.1.3 15, section 3.2.3 16)
	local events (section 3.1.7 16, section 3.2.7 17)
	message processing (section 3.1.5 15, section 3.2.5 16)
	overview 16
	sequencing rules (section 3.1.5 15, section 3.2.5 16)
	timer events (section 3.1.6 16, section 3.2.6 17)
	timers (section 3.1.2 15, section 3.2.2 16)
	Standards assignments 10
	Syntax 11
T	
	Timer events
	client (section 3.1.6 16, section 3.3.6 18)
	server (section 3.1.6 16, section 3.2.6 17)
	Timers
	client (section 3.1.2 15, section 3.3.2 17)
	server (section 3.1.2 15, section 3.2.2 16)
	Tracking changes 24
	Transport 11

Triggered events - higher-layer
client ([section 3.1.4](#) 15, [section 3.3.4](#) 17)
server ([section 3.1.4](#) 15, [section 3.2.4](#) 16)

V
[Vendor-extensible fields](#) 10
[Versioning](#) 10