Infrared Data Association

Point and Shoot Profile

Infrared Data Association℠

Version 1.0

January 12, 2000

Table of Contents

1 Document Status

Point and Shoot Working Group Convenor:

Bryce Jeppsen, Hewlett-Packard bryce_jeppsen@am.exch.hp.com

Document Editor:

Charles Knutson, Oregon State University knutson@cs.orst.edu

Contributors:

Yoshinobu Akimoto, Open Interface
Pete Bramhall, Hewlett-Packard
Eric Edwards, Sony
Melinda Grant, Hewlett-Packard
Gontaro Kitazumi, Okaya Systemware
Yusuke Kushida, NTT DoCoMo
Rob Lockhart, Motorola
Tim Looney, Kodak
Lars Novak, Ericsson
Hiroshi Ono, NEC
Gary Parham, Hewlett-Packard
Gavin Peacock, 3Com
James Scales, Nokia
Mike Sloane, Hewlett-Packard
Richard Stow, Philips
David Suvak, Extended Systems

History:

Version 1.0: First release, January 12, 2000

2 INTRODUCTION

This document presents the IrDA Point and Shoot Application Profile. Section 3 describes the Point and Shoot Usage Model upon which the Application Profile is based. Section 4 presents the Point and Shoot Application Profile.

2.1 References

[IrLAP]	"Serial Infrared Link Access Protocol, IrLAP, Version 1.1," Infrared Data Association http://www.irda.org/standards/specifications.asp
[IrLMP]	"Link Management Protocol, IrLMP, Version 1.1," Infrared Data Association http://www.irda.org/standards/specifications.asp
[IrPHY]	"Serial Infrared Physical Layer Link Specification, IrPHY, Version 1.3," Infrared Data Association http://www.irda.org/standards/specifications.asp
[TTP]	"Tiny TP: A Flow Control Mechanism for use with IrLMP, Version 1.1," Infrared Data Association http://www.irda.org/standards/pubs/Tinytp11.PDF
[LITE]	"Minimal IrDA Protocol Implementation, IrDA Lite, Version 1.0," Infrared Data Association http://www.irda.org/standards/pubs/litever10.pdf
[IrOBEX]	"IrDA Object Exchange Protocol, IrOBEX, Version 1.2," Infrared Data Association http://www.irda.org/standards/pubs/IrOBEX12.pdf
[IrMC]	"IrMC (Ir Mobile Communications) Specification, Version 1.1," February 1999, Infrared Data Association http://www.irda.org/standards/specifications.asp
[VCARD]	"vCard – The Electronic Business Card Exchange Format, Version 2.1," September 1996, The Internet Mail Consortium http://www.imc.org/pdi/vcard-21.doc
[VCAL]	"vCalendar – The Electronic Calendaring and Scheduling Exchange Format, Version 1.0," September 1996, The Internet Mail Consortium http://www.imc.org/pdi/vcal-1.0.doc
[VNOTE]	"IrMC (Ir Mobile Communications) Specification, Version 1.1," February 1999, Infrared Data Association http://www.irda.org/standards/specifications.asp
[VMSG]	"IrMC (Ir Mobile Communications) Specification, Version 1.1," February 1999, Infrared Data Association http://www.irda.org/standards/specifications.asp
[TEXT]	RFC 822: "Standard for the Format of the Arpa Internet Text Messages" RFC 2045: "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies"
[PCL3]	Available through Hewlett-Packard: http://www.hp.com/go/solutions
[PCL5]	Available through Hewlett-Packard: http://www.hp.com/go/solutions
[PS]	Postscript Language Reference, Adobe Systems, Inc., Addison-Wesley, 1999.
[ESP80]	EPSON ESC/P Reference Manual http://www.ercipd.com/isv/edr.htm
[EXIF]	"Digital Still Camera Image File Format Standard (Exif), Version 2.1," Japan Electronic Industry Development Association http://www.jeida.or.jp/document/standard/index-e.html
[FILE]	This data type refers to generic file types that can be stored by a generic file system.
[JETSEND]	JetSend Protocol on IrDA Application Note, Version 1.1, November1999, Infrared Data Association http://www.irda.org/standards/pubs/IrJetSendAppNoteV1.1.pdf

3 POINT AND SHOOT (OBJECT PUSH) USAGE MODEL

3.1 Introduction

The IrDA Point and Shoot Usage Model is based on IrDA-Data that was initially defined and adopted in 1994. It is recommended for high speed, short range, line of sight, point-to-point wireless data transfer and is targeted at the IrDA 1.1 Fast Infrared Data (FIR) 4 Mbps components. The model also applies to IrDA 1.0 Serial Infrared Data (SIR), and will be the same model for the recently announced IrDA Very Fast Infrared (VFIR). This model is useful to over 50 million electronic devices including desktop, notebook and palm computers, printers, digital cameras, public phones/kiosks, cellular phones, pagers and other mobile devices.

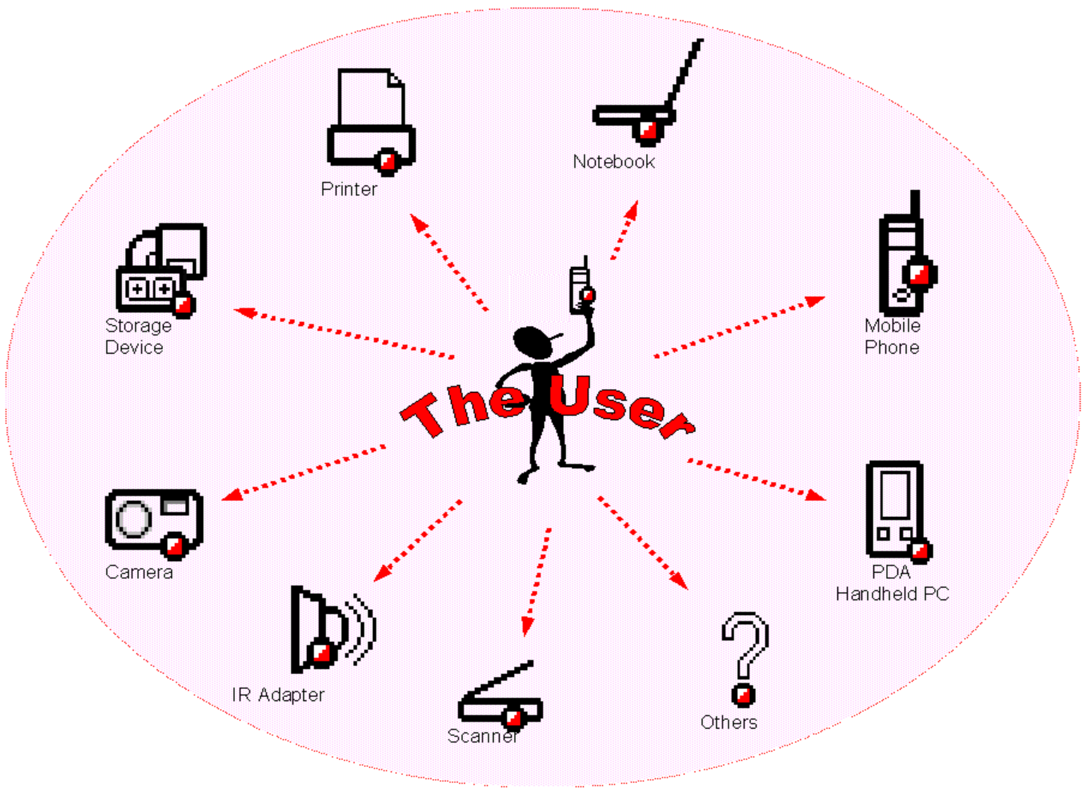
Since the Point and Shoot Usage Model is based on IrDA-Data, it presumes a maximum 30 degree half-angle cone with a maximum range of 1 meter.

3.2 Scope

The scope of the information presented in this section is based on the ability to exchange data between two IrDA-enabled devices. The focus of this Usage Model is on the user’s experience. Many data exchange operations can be reduced to simple object push events, such as printing, faxing, business card exchange, image transfer, and file transfer. The Point and Shoot Usage Model is the universal way to move data objects between IrDA-enabled devices. The key to universal object exchange is support for standard object types (for example, vCard, JPEG (Exif), and text). Almost all IrDA devices will support this capability including PCs, printers, PDAs, cameras, phones, watches, pagers, storage devices, and kiosks.

3.3 User Scenario

Many user scenarios are covered by Point and Shoot object push. The picture below captures the power and simplicity of this Usage Model.



In the following scenarios, the user can:

- push a business card from his phone to another person’s PDA or PC.
- pass the presentation stored on a PC to another PC.
- print his business card from his watch.
- fax a memo from his PDA.
- store images or sound files on a portable storage device.
- print or fax pictures directly from the storage device.
- dial a pay phone simply by pushing a business card from his PDA.
- send pictures over the Internet by pushing pictures from his camera to a kiosk or a cell phone.

The possibilities are endless and the user is control.

3.4 Interoperability

IrReady 2000 devices will have this Point and Shoot capability built-in. The use of standard object types will guarantee that objects are correctly understood on the other device. Devices will be able to alert the user when the other device will not understand an object being sent.

Below is a list of the different data types with examples of what the user may experience when pushing these objects from one device to another. This list is meant to be representative of the types of interoperable data exchange represented by the Point and Shoot Usage Model. This list is by no means exhaustive.

3.4.1 Types of Data Exchange:

- **Business Cards (vCard)**
  - Business Card Exchange
  - Phone List Exchange
  - Business Card Print

- **Appointments, To Do items, Alarms (vCalendar)**
  - Calendar item exchange
  - Calendar item print

- **Text Notes (vNote)**
  - Text item exchange
  - Text item print

- **Messages (vMessage)**
  - Email message exchange
  - Email message print

- **Digital Images**
  - Image exchange
  - Image print

- **Text Files**
  - Text file exchange
  - Text file print

- **Generic Files**
  - Exchange of files between file systems

3.5 Usability

Users will be able to transfer an object to another device by selecting the object and performing a simple operation (such as pressing a button). For example on a PC the user can send a file to another device by dragging the file and dropping it on an icon representing a remote device or an IR application. Another approach may be to select the object and perform a right mouse click operation that will bring up a menu. The user then selects the “send to IR” option and the object is sent. Sending your business card may be as simple as pushing a “send” button.

The short-range, narrow angle of IrDA-Data allows the user to aim, in a Point and Shoot style, at the intended recipient. Close proximity to the other device is natural in this type of data exchange situation, as is pointing one device at another. The limited range and angle of IrDA-Data allows others to simultaneously perform a similar activity nearby without interference. The short-range and narrow angle of IrDA-Data provides a simple form of security and a natural ease of use.

Other technologies with omni-directional capabilities are not as easy to use in this type of scenario. The user is not able to point at the intended recipient. Instead, the user must discover the other devices and choose the appropriate recipient from a list. Close proximity to the intended recipient will usually not help, and choosing the proper device from a list may require special knowledge or additional information.

Point and shoot object exchange using IrDA-Data is the simplest way to transfer objects between two devices.

3.6 Configuration

By default, no configuration should be required for pushing or receiving objects. In some systems the user can select the location of the inbox and perhaps the behavior of prompts. But the device must be equipped to Point and Shoot “out of the box.” In situations where some configuration is required, it should require minimal effort by the user, and should quickly and easily render the device ready to perform an object push or receive where appropriate.

3.7 Reliability

Objects will be sent error free. Specific reliability standards are identified in the test specifications associated with the required enabling technology. Those details are addressed in the Point and Shoot Application Profile (Section 4).

3.8 Additional Information

As objects are received, they may be put into an appropriate data store on the device or delivered to an appropriate application. For example, on a PC received business cards could be placed directly into the user’s PIM. These features may require some configuration on the part of the user.

4 Point and Shoot Profile

4.1 User Requirements

4.1.1 Scope

This Point and Shoot profile defines the minimum requirements for the protocols and procedures that shall be implemented in devices that support the Point and Shoot Usage odel. The most common devices implementing this Usage Model include PCs, notebooks, PDAs, mobile phones, printers and digital cameras.

4.1.2 User Scenarios

The basic scenario covered by this profile is an IrDA device pushing an object to another IrDA device (for example, a mobile phone pushing a business card to a PDA).

4.1.3 Data Object Types

It is necessary to define a set of standard object types for the Point and Shoot profile. The purpose of defining a standard set of data object types is to establish a baseline such that some level of interoperability can be reasonably achieved across a broad range of devices. This standard set of objects is NOT intended to span all of the possible data objects, nor is it intended to define a complete set of data objects to enable highly optimal application solutions. The intent is to define a minimal set of data objects which will establish a common denominator for specific classes of data types, so that interoperability between two devices will occur.

Having a standard set of data objects does not preclude a device manufacturer from supporting other data objects which provide improved value to the end customer, or provide a more optimal end solution. Additional data objects may be supported, but generic interoperability must still be guaranteed in these circumstances in order to comply with the Point and Shoot profile.

As an example, suppose JPEG (Exif) is chosen as the default standardized data object for image data. This format is used to exchange image data on PCs, Digital Cameras, and other devices. Point and Shoot devices that need or require the support of sending or receiving image data should then support the processing of JPEG (Exif) files. It is entirely possible (and even probable) that future digital photography solutions will utilize new file formats for improved picture quality and/or efficiency. As these new file formats are developed, device manufacturers may include support of the new formats, but JPEG (Exif) support must still be the baseline requirement for interoperability.

The standardized object types are shown in the table below. This table includes both required and optional object data types. Table 4.1.4 clarifies the types that are required for specific device classes.

Data Type	**Format**	**Examples**
Business Card	vCard [VCARD]	Business card exchange Phone list exchange Business card print
Appointments, To do items, Alarms	vCalendar [VCAL]	Exchange of calendar items Calendar print
Text Notes	VNote [VNOTE]	Exchange text notes Text note print
Messages and Emails	vMessage [VMSG]	Email exchange Email print
Text files	ASCII using CRLF [TEXT]	Exchange a text document Print a text document
Formatted Document Files	PCL3 [PCL3] PCL5 [PCL5] Postscript [PS] ESC/P [ESCP]	Print ready files Print driver output
Images	JPEG (Exif) [EXIF]	Image exchange Image print
Files	Any [FILE]	Generic file exchange

4.1.4 Device Class Matrix

The following matrix describes the requirements for the different device classes. Any product that desires to be certified as meeting the IrReady2000 interoperability requirements would support the data types shown in the *Required Data Types* section of the matrix below and defined in Section 4.1.3. This table also includes *Optional Data Types* for reference.

	Laptops & Desktops	**Printers**	**Digital Cameras**	**PDAs**	**Phones & Wireless Data Devices**
Required Data Types	[FILE]	[TEXT] [EXIF] [VCARD] [VCAL]	[EXIF]	[VCARD] [VCAL] [TEXT]	[VCARD]* [VCAL]* [VMSG]*
Optional Data Types	[VCARD] [VCAL] [VMSG] [VNOTE] [TEXT] [PCL3] [PCL5] [PS] [ESCP] [EXIF]	[PCL3] [PCL5] [PS] [VNOTE] [VMSG] [ESCP]		[VNOTE] [VMSG]	[VNOTE]

* Not all devices in this class will support the applications represented by these data types. However, if these applications are supported, the device must be capable of sending and receiving these data types.

Note: While wrist watches are not included in this table, work is underway in IrDA to determine appropriate data types for interoperability between wrist watches. When those specifications are completed, this document should be updated to reflect those changes.

4.2 Profile Overview

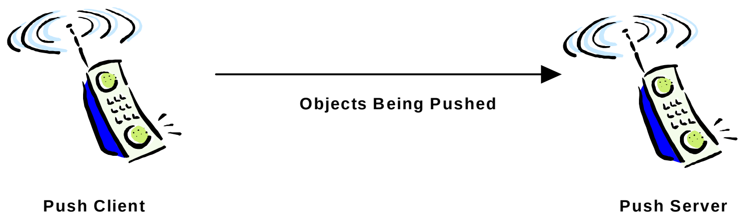
4.2.1 Configuration and Roles

The following roles are defined for this profile.

Push Server – The device that provides an object exchange server. The Push Server waits passively for the client to initiate the operation.

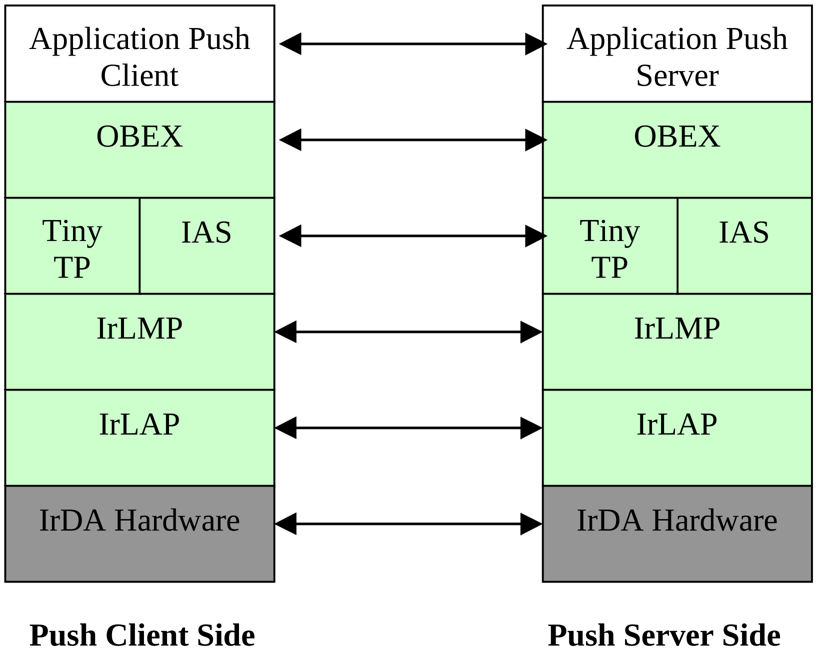
Push Client – The device that pushes the object to the Push Server. The Push Client initiates the operation.

The figure below shows an example of a Push between two mobile phones. In this example the phone on the left is a Push Client that pushes the object, and the phone on the right is a Push Server that receives the object.



4.2.2 Protocol Stack

The diagram below defines the minimum required protocol stack that shall be implemented by a device that conforms to this profile. Alternative stacks or components (such as JetSend [JETSEND]) may be implemented by a device in addition to this minimum.



IrDA Hardware is governed by the physical specification in [IrPHY].

IrLAP is the link level protocol specified in [IrLAP].

IrLMP is a multiplexing layer specified in [IrLMP].

Tiny TP provides flow control and is specified in [TTP].

IAS is the Information Access Service specified in [IrLMP].

OBEX includes both a session level protocol and an application framework. Both are specified in [IrOBEX].

Application Push Client and **Application Push Server** are the application entities, which provide the user interface and perform the operation of the Point and Shoot profile. They are discussed later in this document.

Ultra is not required for the Point and Shoot Profile. However, while not required, support is strongly encouraged in order to extend the interoperable reach of Point and Shoot devices.

4.2.3 Conformance

For a device to conform to this profile, all capabilities indicated as mandatory shall be supported in the specified manner. This also applies to all optional and conditional capabilities for which support is indicated. All supported capabilities are subject to verification as part of the IrReady 2000 certification program.

4.3 User Interface Aspects

4.3.1 Mode Selection

Push Server Mode is the state in which a Push Server is ready to receive an object from a Push Client. When entering this mode the Push Server must register the OBEX IAS entry and set the OBEX hint bit. It must be in a state where it is ready to respond to incoming Discovery frames and accept an incoming OBEX connection.

It is ideal that a Push Server be in this mode whenever the physical IR port is enabled (in other words, when the IR port is able to receive signals). In some devices the IR port is enabled whenever the device is turned on. For other devices the user must explicitly turn on the IR port. Turning on the IR port will ideally correspond to entering Push Server Mode.

However, it should be noted that for some devices it is either impractical or impossible to require that enabling the IR port will necessarily place the device into Push Server Mode. Where that ideal is not achievable, the user should be able to place the device into Push Server Mode as easily as possible, which is defined here as requiring only a single additional action on the part of the user.

To summarize, there are two scenarios in which a device will enter Push Server Mode:

Scenario One: The device automatically enters Push Server Mode when turned on. This is the recommended method.

Scenario Two: The device enters Push Server Mode when the user specifically enables it. This method may be required for devices with security or power usage constraints. The user should only be required to perform a single action via the user interface to cause the device to enter Push Server Mode.

4.3.2 Function Selection

The **Object Push Function** initiates the sending of an object to a Push Server. The user initiates this function. Typically, the function is selected for a specific object or group of objects. For example, in the Windows environment the user selects a file in the Explorer window, clicks the right mouse button and selects “IR recipient” from the “Send to” menu. When the selection is made the object is sent.

In most cases only one device will be available. The Point and Shoot Usage Model works best when the user selects the desired device by pointing at it. If multiple devices are in the IR space then the user must select from a list or be told to position the device so only one device is in range. The device may also use the hint bits of the discovered devices to identify those that support IrOBEX (since all Point and Shoot devices would support this hint bit). In this way a device could either present a more effective list to the user, or intelligently determine which of the devices is the most appropriate one to connect to.

4.3.3 Application Usage

When the user wants to push an object from a Push Client to a Push Server, the following is a typical scenario.

Push Client	Push Server
	The user sets the device into Push Server Mode if it is not already.
The user of the Push Client selects the object or objects to send.	
The user points the IR port of the Push Client device at the IR port of the Push Server device.	
The user selects the Object Push Function to send the selected object(s). (Objects being pushed should be of a type appropriate for the Push Server.) It is recommended that a status indicator show the progress of the operation.	
	It is recommended that user intervention be kept to a minimum on the Server device. It is possible that the user may be asked to accept or reject the object. Also if an object with the same name already exists the user may be asked if the existing object should be overwritten.
It is recommended that the user be notified of the result of the operation.	It is recommended that, where appropriate, the Push Server device notify its user of the result of the operation.

4.4 Application Layer

4.4.1 Feature Overview

A device following the Point and Shoot profile must be a Push Client, a Push Server or both and must support appropriate content types from those listed in the next section.

4.4.2 Content Types

To achieve application level interoperability, content formats are defined for Object Push. Sections 4.1.3 and 4.1.4 identify required and optional data types for Point and Shoot devices. If a device supports additional content formats for a given application, it must still support the required ones listed in these sections.

4.4.3 Generic File Push

Generic File transfer applications can send and receive files in any format. It is assumed that in this case both devices contain applications that understand the file format or that the file is simply stored on the device. It is also possible to send directory hierarchies containing Generic Files. Push Servers are not required to support Generic Files and if they do support them they are not required to support folder/directory hierarchies. The Push Client must verify that the Push Server supports folder/directory hierarchies by attempting to set the current folder of the receiving device to the root folder (see Section 4.5.5 for more details).

4.4.4 Application Architecture

The Push Client and Push Server are both built on top of the OBEX application framework. A Push Client uses OBEX to push objects to the inbox of a Push Server. The Push Client only knows that the objects are successfully received. It does not know the layout or construction of the Push Server's inbox.

A Push Server's inbox must take one of the following forms:

General Storage Location	*Holds objects of any type*. An example is a directory in a file system. It is possible to automatically dispatch objects from a general storage location to a database. For example, if a vCard is received it can be dispatched to the address book. The exception is when pushing a folder hierarchy. For example, vCards that are inside a folder being pushed are not automatically dispatched to the address book.
Database	*A data store or application that contains objects of a specific type.* An example is an address book in a mobile phone, which holds phone book items (vCards).
Process	*An application or program that processes the object as data*. An example is a printer, which will print the object. Processes are allowed to process the object as it is received. This means that the object does not have to be completely received before processing can begin. Since OBEX uses Tiny TP flow control the Push Server can properly pace the Push Client.

The table below shows the application procedure required by the Push Client for pushing one or more objects to a push server.

Push Client	**Details**
OBEX CONNECT	Target Header must not be used.
One or more OBEX PUTs for sending one or more objects.	
OBEX DISCONNECT	

The following table shows the application procedure required by the Push Client for pushing a folder hierarchy to a Push Server. This procedure is optional.

Push Client	**Details**
Set the current folder to the root using the SETPATH command. If this operation fails then the Push Server does not support folders.	Name header is empty.
Create a new folder (if it does not already exist) in the Push Server's current folder using SETPATH. The current folder is changed to this new folder.	Name header is set to the name of the new folder.
Push all files to the new folder using a PUT command for each file.	The Name header is set to the name of the file.
Folders are created using SETPATH.	Name header is set to folder name. This application procedure is applied recursively to each folder until the folder hierarchy is sent. Setting the current folder to the root is only performed once at the beginning.
Set the current folder back to the parent folder using SETPATH.	The Backup flag is set and no Name header is sent.

4.5 OBEX Requirements

This section details the use of IrOBEX in the Point and Shoot Profile.

4.5.1 Symbols and Conventions

The Application Profile must use the following scheme to define the support for individual features. The following symbols are used:

M	Mandatory support. Refers to capabilities that shall be used in the profile.
O	Optional support. Refers to capabilities that can be used in the profile, but are not required.
X	Excluded. Refers to capabilities that may be supported by the device but shall not be used in this profile.

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

4.5.2 OBEX Operations

The table below shows the OBEX operations that are used in the Point and Shoot profile.

OBEX Operation	Push Client	Push Server
Connect	M	M
Disconnect	O	M
Put	M	M
Get	O	O
Set Path	O	O
Abort	M	M
Reserved	X	X
User Definable	X	X

4.5.3 OBEX Headers

The table below shows the OBEX headers used in the Point and Shoot profile.

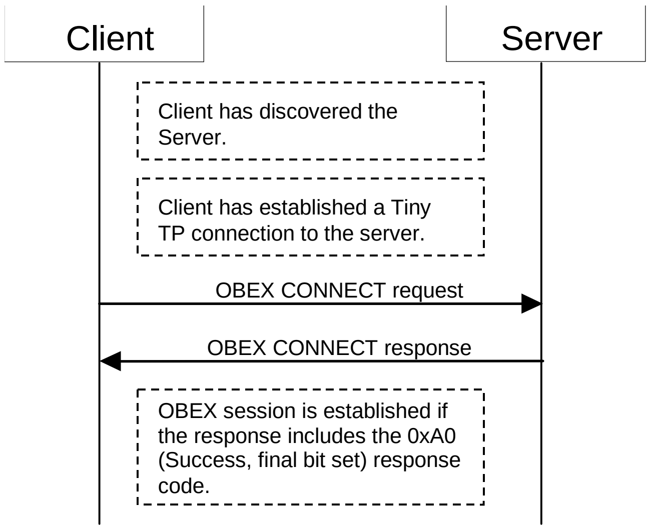
OBEX Headers	Push Client	Push Server
Count	O	O
Name	M	M
Type	O	O
Length	O	O
Time	O	O
Description	O	O
Target	X	X
HTTP	O	O
Body	M	M
End of Body	M	M
Who	X	X
Connection ID	X	X
Application Parameters	X	X
Authenticate Challenge	X	X
Authenticate Response	X	X
Object Class	X	X
Reserved	X	X
User Definable	X	X

4.5.4 Establishing an OBEX session

Setting up an OBEX session for Point and Shoot involves three steps:

1. Push Client discovers the Push Server device.
2. Push Client establishes a Tiny TP connection to the Push Server device. (See Section 4.6, Tiny TP/IrLMP Operations.)
3. Push Client performs an OBEX connect operation to the Push Server.

The figure below shows how the OBEX session is established.


Client
Server
Client has discovered the Server.
Client has established a Tiny TP connection to the server.
OBEX CONNECT request
OBEX CONNECT response
OBEX session is established if the response includes the 0xA0 (Success, final bit set) response code.

The OBEX connection is established to the inbox service so no targeting information is used. The OBEX connect request must contain the following fields.

Field/ Header	Name	Value	M/O	Explanation
Field	Opcode for CONNECT	0x80	M	
Field	Packet Length	Varies	M	
Field	OBEX Version Number	Varies	M	
Field	Flags	Varies	M	
Field	Max OBEX Packet Length	Varies	M	

The OBEX connect response must contain the following fields.

Field/ Header	Name	Value	M/O	Explanation
Field	Response code for CONNECT request	0x0A	M	0xA0 for success
Field	Packet Length	Varies	M	
Field	OBEX Version Number	Varies	M	
Field	Flags	Varies	M	
Field	Max OBEX Packet Length	Varies	M	

4.5.5 Pushing Objects

Objects are pushed to the Push Server using the OBEX PUT operation. Pushing an object can take one or more OBEX packets. Compliant OBEX Push Servers should be able to receive multiple sequential PUT requests.

The PUT packet must include the following fields and headers.

Field/ Header	Name	Value	M/O	Explanation
Field	Opcode for PUT	0x02 or 0x82	M	0x02 is used for packets previous to the last put packet. 0x82 (which is 0x02 with the high bit set) is used for the last put packet.
Field	Packet Length	Varies	M	
Header	Name	Varies	M	The header value is the name of a single object, object store, or log information.
Header	Type	Varies	O	The MIME type of the object. This header is optional but highly recommended.
Header	Length	Varies	O	Length of the object. This header is optional but highly recommended.
Header	Body/End of Body	Varies	M	End of Body identifies the last chunk of the object body.

The response to the PUT request has the following fields and headers.

Field/ Header	Name	Value	M/O	Explanation
Field	Response code for PUT	0x90, 0xAO, 0xCD or 0xCF	M	0x90 for continue 0xA0 for success 0xCD if the object is too large 0xCF if the object type is not supported
Field	Packet Length	Varies	M	

Other headers, which can be optionally used, are found in [IrOBEX].

The type of an object is distinguished in two ways, first by using an extension in the name and second by sending a **Type** header. Sending a **Type** header is optional but highly recommended. Sending a **Name** header is mandatory. The table below shows the MIME types and name extensions required for each of the content types.

Object	MIME encoding (Type)	Name extension
vCard 2.1	text/x-vcard	.vcf
vCalendar 1.0	text/x-vcalendar	.vcs
vMessage 1.1	text/x-vmessage	.vmg
vNote 1.1	text/x-vnote	.vnt
Plain ASCII text	text/plain	.txt
JPEG (Exif) Image	Image/jpeg	.jpg

4.5.6 Folder Operations

4.5.6.1 Setting the Current Folder to the Root

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

Field/ Header	Name	Value	M/O	Explanation
Field	Opcode for SETPATH	0x82	M	
Field	Packet Length	Varies	M	
Field	Flags	0x02	M	“Backup level” flag is set to 0 and “Don’t Create” flag is set to 1
Field	Constants	0x00	M	Constants are not used and must be set to 0
Header	Name	Empty	M	Name header is empty

The response to the SETPATH request for setting the current folder to the root has the following fields.

Field/ Header	Name	Value	M/O	Explanation
Field	Response code for SETPATH	0xA0, 0xC3 or 0xD1	M	0xA0 for success 0xC3 if SETPATH is not supported 0xD1 if folders in the inbox are not supported
Field	Packet Length	Varies	M	

4.5.6.2 Creating a Folder

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

Field/ Header	Name	Value	M/O	Explanation
Field	Opcode for SETPATH	0x82	M	
Field	Packet Length	Varies	M	
Field	Flags	0x00	M	"Backup level" flag is set to 0 and "Don't Create" flag is set to 0
Field	Constants	0x00	M	Constants are not used and must be set to 0
Header	Name	Varies	M	Name of the folder

The response to the SETPATH request for creating a new folder has the following fields.

Field/ Header	Name	Value	M/O	Explanation
Field	Response code for SETPATH	0xA0 or 0xCD	M	0xA0 for success 0xCD if there is not enough room for a new folder
Field	Packet Length	Varies	M	

4.5.6.3 Setting the Current Folder Back to the Parent

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

Field/ Header	Name	Value	M/O	Explanation
Field	Opcode for SETPATH	0x82	M	
Field	Packet Length	Varies	M	
Field	Flags	0x03	M	"Backup level" flag is set to 1 and "Don't Create" flag is set to 1
Field	Constants	0x00	M	Constants are not used and must be set to 0

Response to the SETPATH request for setting the current folder back to the parent has the following fields.

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

4.5.7 Disconnecting an OBEX session

An OBEX session can be disconnected in two ways. First, the OBEX connection can be disconnected using the DISCONNECT procedure. Second, the underlying Tiny TP connection can be disconnected. Normally after all objects have been pushed, the Tiny TP connection to the OBEX server is disconnected so there is really no need to perform an OBEX DISCONNECT procedure. If the Tiny TP connection is used for other purposes as well (such as synchronization or file transfer) then the Tiny TP connection must be left up and an OBEX DISCONNECT should be issued. Push Servers must be able to handle both methods of disconnect.

The OBEX DISCONNECT request must contain the following fields.

Field/ Header	Name	Value	M/O	Explanation
Field	Opcode for DISCONNECT	0x81	M	
Field	Packet Length	Varies	M	

Other headers (such as **Description)** which can be optionally used are found in [IrOBEX].

The response to an OBEX DISCONNECT request must contain the following fields.

Field/ Header	Name	Value	M/O	Explanation
Field	Response code for DISCONNECT	0xAO	M	0xA0 for success
Field	Packet Length	Varies	M	

4.6 Tiny TP/IrLMP Requirements

This section details the use of Tiny TP and IrLMP in the Point and Shoot Profile.

4.6.1 Tiny TP/IrLMP Operations

Tiny TP and IrLMP combine to form the IrDA transport layer. The steps involved in setting up a Tiny TP connection to a Push Server are as follows:

1. Push Client discovers the Push Server device and establishes an IrLAP connection.
2. Push Client queries the IAS of the Push Server for the LSAP-SEL entry of the OBEX IAS entry (see Section 4.7 IAS for more details).
3. Push Client performs a Tiny TP connect request to the LSAP-SEL retrieved in step 2.

If the Push Client already has an IrLAP connection to the Push Server then step 1 can be skipped and the Push Client should start at step 2.

Note: If the Push Client has Push Server capability and there is already a Tiny TP connection to the Push Server of the Push Client device, then there may be problems if the Push Client attempts to establish a Tiny TP connection to the Push Server device. The reason is because the Push Server device may be built using

IrDA Lite [LITE]. IrDA Lite does not require support of Tiny TP/IrLMP disconnect and some IrDA Lite devices may only allow one Tiny TP connection to OBEX at a time. This means that these devices cannot support two Tiny TP connections to OBEX at once (one from its Client to the Server of another device and one from the Client of another device to its server). If one of these devices receives an incoming Tiny TP connection when it already has an outgoing Tiny TP connection then it may perform an IrLAP disconnect which will disconnect all Tiny TP/IrLMP connections. For the purposes of this Profile, a Point and Shoot device is not expected to be capable of sending and receiving simultaneously. However, a Point and Shoot device is expected to handle such a request from another device without dropping the connection.

4.6.2 Discovering the Push Server

The Push Client device must discover the Push Server device using the IrLMP discovery service described in [IrLMP]. If the Push Client device already has an IrLAP connection to a another device, then it is assumed that this other device is the Push Server.

4.6.3 Establishing a Tiny TP Connection

The Push Client must establish a Tiny TP connection to the Push Server using the Connect request procedure described in [TTP].

4.7 IAS Requirements

The Push Server must have an IAS entry for a default OBEX server as described in [IrOBEX].

4.8 IrLAP Requirements

There are no special issues concerning IrLAP.

4.9 Physical Layer Requirements

Devices are allowed to support the short-range option as described in [IrPHY].