BLUETOOTH DOC	Date / Year-Month-Day	Approved	Revision	Document No
	2001-10-05	Draft	0.95a	
Prepared	e-mail address			N.B.
Printing Working Group	bt-printing-feedback@bluetooth.org			Confidential

BASIC PRINTING PROFILE

Interoperability Specification

Abstract

This application profile defines the application requirements for Bluetooth™ devices necessary for the support of the Basic Printing usage model. The requirements are expressed in terms of end-user services, and by defining the features and procedures that are required for interoperability between Bluetooth devices in the Basic Printing usage model.

Revision History

Revision	Date	Comments
0.1 draft	2000-04-14	First draft for PWG review.
0.19	2000-07-31	Addressed authentication, encryption, master-slave switch, and referenced object retrieval.
0.20-0.35	2000-08	Addressed GAP requirements. General editorial changes.
0.40	2000-09-19	Added the IPP concepts, media size list, character set list, and Asian-language line-break handling
0.41-.42	2000-09/10	General editorial changes.
0.45	2000-11-20	Added XML-encoding approach to job control, mandatory image format. Prepare for 0.5 draft release.
0.5	2000-11-22	Updated based on 0.5 vote.
0.55	2001-01-05	Updated based on 4Q/2000 F2F and Conference calls.
0.6-0.66	2001-02	Updated based on 1Q/2001 F2F and Conference calls.
0.70	2001-02-25	Updated based on 0.7 vote.
0.74 –0.80	2001-03/04	Updated based on associate input, 1Q/2001 Conference calls, and F2F meetings.
0.81-0.88	2001-05	Restructuring for readability; prepare for 0.9 draft release.
0.9	2001-05	Promotion of v0.88 to 0.9.
0.91	2001-06	Applied BARB changes, IEEE conformance language, and general editorial clean up.
0.92-0.945	2001-07/08	Applied changes from technical writer, Associate/Adopter comments, and 2Q/2001 Conference call changes.
0.95	2001-08	Promotion of v0.945/6 to 0.95 with approved minor corrections.
0.95a	2001-10	1. Added text in Section 11.9 to clarify the usage requirements of HTTP Headers in SOAP messages. 2. Re-ordered entries in the Service Record tables to match requirements made by SDP. 3. Corrected reference to the Assigned Numbers Document in the Normative References section. 4. Added clarifying example algorithm for locating BPP devices via the CoD bits. 5. Corrected typo-error in Table 16 example.

Contributors

Olof Larsson	Axis Communications
Patrik Olsson	Ericsson Mobile Communications AB
Alan Berkema	Hewlett-Packard Company
Todd Fischer	Hewlett-Packard Company
Melinda Grant	Hewlett-Packard Company
Jeff Morgan	Hewlett-Packard Company
John Waters	Hewlett-Packard Company
Harry Lewis	IBM Corporation
Henrik Holst	i-data International
Jim Combs	Lexmark International, Inc.
Jerry Thrasher	Lexmark International, Inc.
Don Wright	Lexmark International, Inc.
Thomas Nielson	Microsoft Corporation
Patrick Vine	Microsoft Corporation
Steve Kranish	Motorola, Inc.
Don Levinstone	Motorola, Inc.
Martin Roter	Nokia Mobile Phones
Goro Ishida	Seiko Epson Corporation

Disclaimer and Copyright Notice:

List of Tables

List of Figures

Contents

Foreword

This document, together with the Generic Object Exchange Profile and the Generic Access Profile, forms the Basic Printing usage model.

Interoperability between devices from different manufacturers is provided for a specific service and usage model if the devices conform to a Bluetooth SIG-defined profile specification. A profile defines a selection of messages and procedures (generally termed *capabilities*) from the Bluetooth SIG specifications and gives an unambiguous description of the air interface for specified service(s) and usage model(s).

The Basic Printing Profile (BPP) is designed to operate over Bluetooth interfaces implementing Version 1.0B (plus critical errata) or later of the Bluetooth Core Specification and is intended to be Bluetooth radio interface-independent.

All defined features are process-mandatory. This means that if a feature is used, it is used in a specified manner. Whether the provision of a feature is mandatory or optional is stated separately for both sides of the Bluetooth air interface.

1 Introduction

1.1 Scope

The Basic Printing Profile defines the requirements for the protocols and procedures that shall be used by applications providing the Basic Printing usage model. This Profile makes use of the Generic Object Exchange Profile (GOEP) [10] to define the interoperability requirements for the protocols needed by applications. The most common devices using these usage models are mobile devices such as mobile phones, pagers, and PDAs, although more complex devices are not excluded. Usage models include printing of text emails, short messages, and formatted documents. Optional support for the printing of structured data objects such as vCard and vCalendar is also defined, as well as methods for negotiating the use of other formats supported by the printer.

Printing from sending devices such as laptops and desktop PCs, where printer-specific drivers may be loaded, is defined in the Hardcopy Cable Replacement Profile [21].

1.2 Bluetooth Profile Structure

In Figure 1: Bluetooth Foundation Specification Profiles, the Bluetooth profile structure and the dependencies of the profiles are depicted. A profile is dependent upon another profile if it re-uses parts of that profile, by implicitly or explicitly referencing it. Dependency is illustrated in the figure: a profile has dependencies on the profile(s) in which it is contained — directly and indirectly. For example, the Basic Printing Profile is dependent on the Generic Object Exchange, Serial Port, and Generic Access Profiles.

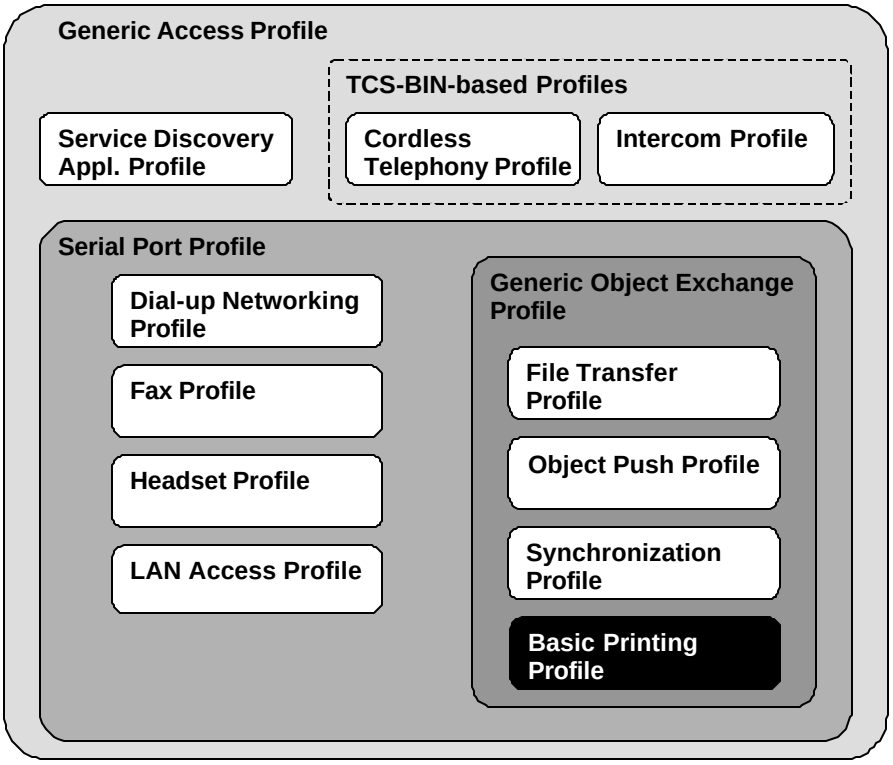

Figure 1: Bluetooth Foundation Specification Profiles plus the Basic Printing Profile

1.3 Related Specifications

The Bluetooth Foundation Specification includes two separate specifications related to OBEX and applications using OBEX:

1. Bluetooth IrDA Interoperability Specification [7]

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

2. Bluetooth Generic Object Exchange Profile Specification [10]

   - Generic interoperability specification for the application profiles using OBEX.
   - Defines the interoperability requirements of the lower protocol layers (e.g., Baseband, LMP, L2CAP, and RFCOMM) for the application profiles.

3. Bluetooth Basic Printing Profile Specification (this specification)

- Application profile for basic printing applications.
- Defines the interoperability requirements for the applications within the Basic Printing application profile.
- Does not define the requirements for Baseband, LMP, L2CAP, or RFCOMM.

1.4 Symbols and Conventions

1.4.1 Requirement Status Symbols

In this document the following symbols are used:

"M" for mandatory to support (used for capabilities that shall be used in the profile). For each operation, the definition of mandatory is clarified for both the Sender and the Printer.

"O" for optional to support (used for capabilities that may be used in the profile).

"C" for conditional support (used for capabilities that shall be used as a result of the support for other capabilities).

"X" for excluded (used for capabilities that may be supported by the unit but shall never be used in the profile).

"N/A" for not applicable (in the given context it is impossible to use this capability).

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

1.4.2 Document Naming Conventions

In general, operations and operation attributes are mixed case with the first letter of each word being capitalized. Most of these names are derived directly from the IPP 1.1 Specification [19] by removing the hyphens and capitalizing the first letter of each word.

Attribute values shall be lower-case with hyphens separating words.

Numeric values in SOAP-encoded messages shall be represented by US-ASCII strings interpreted as the value. For hexadecimal numbers, the “x” shall be lower case.

Negative values for signed integers (s4) shall be represented in their signed 2’s complement representation.

1.4.3 Conformance Language

The keywords “shall”, “should”, “may”, and “can” in this document are to be interpreted as described in the IEEE Standards Style Manual [37].

2 Profile Overview

2.1 Protocol Stack

The figure below shows the protocols and entities used in this Profile.

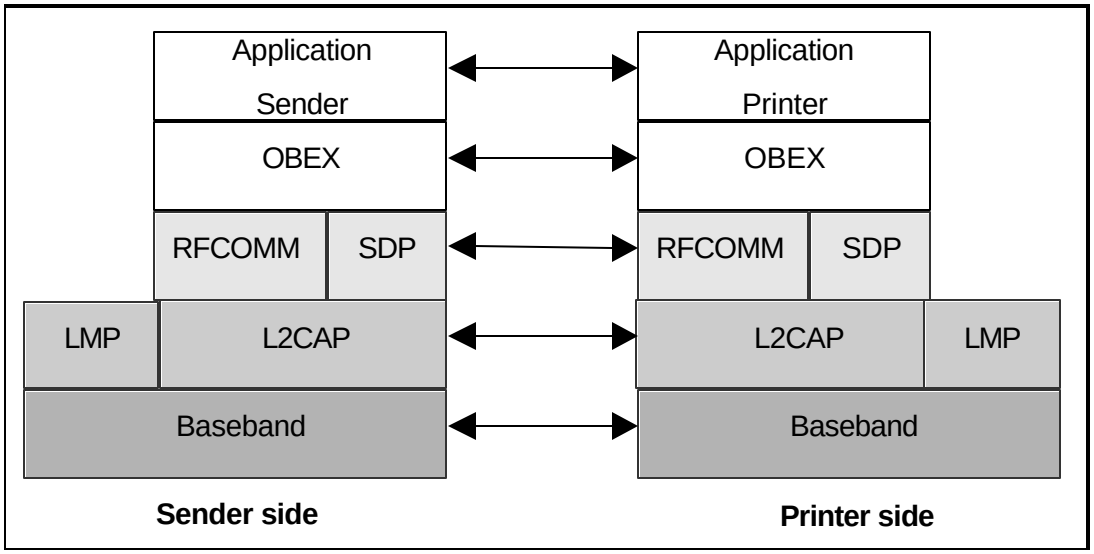

Figure 2: Protocol Model

Baseband [1], LMP [2], and L2CAP [3] are the OSI layer 1 and 2 Bluetooth protocols. RFCOMM [4] is the Bluetooth adaptation of GSM TS 07.10 [5]. SDP is the Bluetooth Service Discovery Protocol [6]. OBEX [8] is the Bluetooth adaptation of IrOBEX [7].

The RFCOMM, L2CAP, LMP, and Baseband interoperability requirements are defined in Chapter 6 in GOEP [10].

2.2 Configurations and Roles

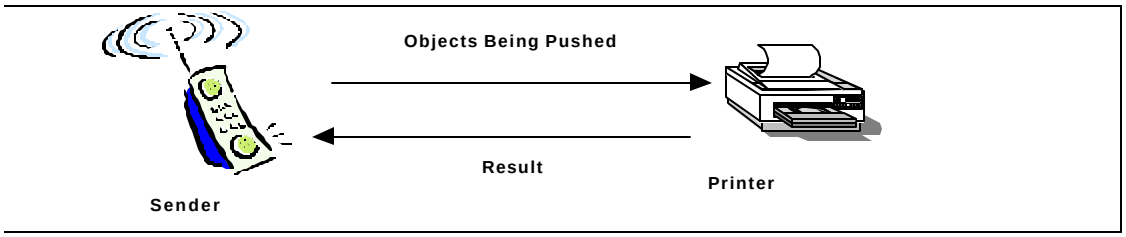

Figure 3: Example of a Mobile Phone Sending an Object to a Printer

The following roles are defined for this Profile:

Printer – This is the server device that provides an object exchange server. In addition to the interoperability requirements defined in this Profile, the Printer shall comply with the interoperability requirements for the server of the GOEP if not defined in the contrary.

Sender – This is the client device that pushes an object to the Printer. In addition to the interoperability requirements defined in this Profile, the Sender shall comply with the interoperability requirements for the client of the GOEP if not defined in the contrary.

2.3 User Requirements and Scenarios

The scenarios covered by this Profile are:

- Printing emails, text messages, and plain or formatted text from a Sender (e.g., a mobile phone or PDA). Plain text and text formatted according to the mandatory PDL and character encoding may be printed. Worldwide language support is optional. Depending on the choice of PDL, either the Printer or the Sender may format the page.

- Printing information obtained via a Sender (e.g., WML pages, HTML, etc.). Mandatory requirements are defined to support text and raster information. Depending on the choice of PDL, either the Printer or the Sender may format the page.

- Printing structured information stored on a Sender (e.g., vCard, vCalendar, etc.). The object is sent unformatted to the Printer and the Printer is responsible for formatting the object.

- Printing information that is stored or created on the Internet or an intranet by passing a reference or pointer to that information from the Sender to the Printer.

- Printing of information via a Sender with precise control over location and presentation on the printed page.

- Printing of information and controlling a Printer's settings via a user interface on the Sender that is created and supplied by the Printer.

2.4 Profile Fundamentals

The profile fundamentals are the same as defined in the GOEP unless otherwise stated in this specification.

Link-level authentication and encryption are mandatory to support and optional to use; see Section 13.1.

Bonding [13] is mandatory to support and optional to use; see Section 13.1.

There are no fixed master-slave roles for the execution of this Profile. The master-slave switch is optional to support and optional to use.

OBEX authentication is mandatory to support for Senders. OBEX authentication is optional to support and optional to use for Printers. For details, see Section 11.4.

2.5 Conformance

If conformance to this Profile is claimed, all capabilities indicated as mandatory for this Profile shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory capabilities, and optional and conditional capabilities for which support is indicated, are subject to verification as part of the Bluetooth qualification program.

3 Printer Modes

The Printer can be in one of four different modes. The table below describes these modes and the requirements for the Printer in each mode.

Printer Mode	Requirement	Description
Bluetooth Offline mode *	Optional	Printer is not able to receive print jobs over the Bluetooth link. It is not possible to connect to the Printer. The Printer cannot be discovered using either of the two inquiry procedures described in [13].
Bonding mode	Mandatory	Printer is ready to be bonded with Sender devices.
Private Online mode	Optional	Printer can receive print jobs over the Bluetooth link only from devices that know the Printer's Bluetooth device address. The Printer cannot be discovered using any of the two inquiry procedures described in [13].
Public Online mode	Mandatory	Printer can receive print jobs over the Bluetooth link from devices that know the Printer's Bluetooth device address. The Printer can be discovered using one or both of the two inquiry procedures described in [13].

Table 1: Printer Modes and Requirements

* Bluetooth Offline mode refers to the Printer’s Bluetooth link.

The table below describes the GAP modes (see [13]) that a device shall be in for the different printer modes described above.

Printer Mode	**GAP Mode when Printer is Idle**	**GAP Mode when Printer is Busy**
Bluetooth Offline mode *	Mandatory: Non-connectable mode Mandatory: Non-discoverable mode	Mandatory: Non-connectable mode Mandatory: Non-discoverable mode
Bonding mode	Mandatory: Connectable mode Mandatory: Either limited discoverable mode, general discoverable mode, or both Mandatory: Pairable mode	N/A
Private Online mode	Mandatory: Connectable mode Mandatory: Non-discoverable mode	Optional: Connectable mode Mandatory: Non-discoverable mode
Public Online mode	Mandatory: Connectable mode Mandatory: Either limited discoverable mode, general discoverable mode, or both	Optional: Connectable mode Optional: Either limited discoverable mode, general discoverable mode, or both

Table 2: GAP Modes Associated with Printer Modes

* Bluetooth Offline mode refers to the Printer's Bluetooth link.

When entering Bonding mode, Private Online mode or Public Online mode, Printers shall ensure that the appropriate bits are set in the Class of Device (CoD) fields (see Section 16.1). When entering any of these modes the Printer shall register a service record in the SDDB (see Section 12).

4 User Interface Aspects

In the following sections the presented scenarios work as examples and variations of implementations that are possible and allowed.

4.1 Print Object: Bluetooth Device Address of Printer Known

When a Sender wants to print an object on a Printer, the following scenario can be followed. In this case, the Sender already knows the Bluetooth device address of the Printer.

If link-level authentication is used, the user might have to enter a Bluetooth PIN at some point. If OBEX authentication is used, the user shall enter a password and possibly a user ID at some point.

Sender	**Printer**
	The Printer is (or the user sets the device) in **Private Online mode** or **Public Online mode**.
The user of the Sender selects the print function on the device and selects the object, or a reference to the object, to print.	
The Sender connects to the Printer with the known Bluetooth device address. If the Sender cannot connect to the Printer, the Printer may be out of reach, busy, or not in either of the two Online modes. It is recommended that this be communicated to the user.	
The Sender then optionally configures the Printer (via direct control or printer-supplied user interface) and sends the object or reference to the Printer.	
	The Printer receives the object, or reference, and prints it. If the Printer is busy or if an error occurs, the Printer returns a status code to the Sender.
It is recommended that the user be notified of the result of the operation.	

Table 3: Print Object for Known Printer Address

4.2 Print Object: Bluetooth Device Address of Printer Not Known

When a Sender wants to print an object on a Printer, the following scenario can be followed. In this case the Sender does not know the Bluetooth device address of the Printer.

If link-level authentication is used, the user might have to enter a Bluetooth PIN at some point. If OBEX authentication is used, the user shall enter a password and possibly a user ID at some point.

Sender	**Printer**
	The Printer is (or the user sets the device) in **Public Online mode**.
The user of the Sender selects the print function on the device and selects the object, or a reference to the object, to print.	
A list of devices that may support the Basic Printing service is displayed to the user. The user selects a device to send an object. If the desired Printer is not in the list of devices or if the Sender cannot connect to the Printer, the Printer may be out of reach or busy. If the selected device does not support the Basic Printing service, the user is prompted to select another device.	
The Sender then optionally configures the printer (via direct control or printer-supplied user interface) and sends the object or reference to the printer.	
	The Printer receives the object, or reference, and prints it. If the Printer is busy or if an error occurs, the Printer returns a status code to the Sender.
It is recommended that the user be notified of the result of the operation.	

Table 4: Print Object for Discovered Printer Address

5 Application Layer Overview

The Basic Printing Profile specifies the following printing services:

- Direct Printing: The content to be printed by the Printer is stored on the Sender
- Reference Printing: The content to be printed by the Printer is stored on the network with a reference to that content stored on the Sender.

The Printer's main task is to accept information from a Sender, queue it (if the Printer is capable), and either print the content or print the referenced content.

5.1 Print Model Descriptions

This section describes the print models which provide the services that are required or allowed by devices supporting the Basic Printing Profile.

5.1.1 Simple Push Transfer Model

For Simple Push Transfer, any details about the level of functionality and support from the Printer are discovered via the Basic Printing Service Record. The Sender shall use OBEX PUT requests (referred to as a FilePush operation in this Profile) to push print content to the Printer. No mechanisms for any type of status or error recovery for the Printer are required, other than those provided by the OBEX transport; nor are any mechanisms for Printer control supported. The Printer shall print the information from the Sender using its own parameters and settings (i.e., default paper, orientation, quality, etc.). If the print content contains a referenced object such as an image there is a mechanism allowing this image to be retrieved by the Printer from the Sender over a separate OBEX connection that is initiated by the Printer.

Printers that intend to print using the Basic Printing Profile shall support the Simple Push Transfer model. Senders shall provide support for either the Simple Push model or the Job-Based Transfer model and may provide support for both.

The Simple Push Transfer model also provides mechanisms for transferring a reference (referred to as SimpleReferencePush) to print content; if this capability is supported, the Printer retrieves this content and prints it.

Figure 4 contains a diagram of the basic structure of the Simple Push Transfer model.

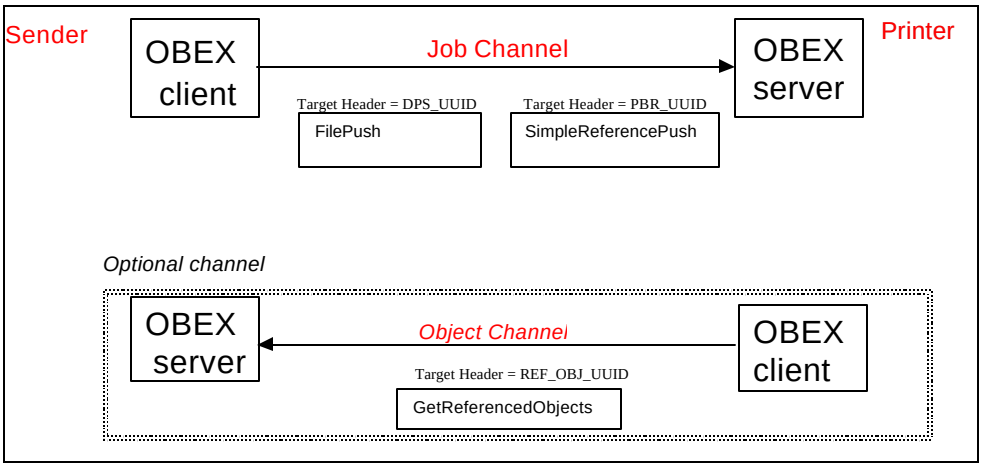

Figure 4: Simple Push Transfer Model Connections

The FilePush operation and its use of OBEX is described in Section 6. A mapping of OBEX headers in the FilePush operation is described in Section 11.8.

The SimpleReferencePush operation is described in Section 8.

The GetReferencedObjects operation is described in Section 7.1.6.

5.1.2 Job-Based Transfer Model

For job-based data transfer, information from a Sender shall be transferred to the Printer as part of a print session, initiated by the Sender. This session allows the Sender more control over the printed output as well as more detailed information about both the Printer and the print job. Each print job shall be identified by a four-byte integer, the JobId, allocated by the Printer and used to track the print job by both the Sender and the Printer.

The operations and attributes for job-based data transfer are based on the Internet Printing Protocol (IPP) model [19]. Because IPP/1.1 is very comprehensive, this Profile leverages a subset of its operations and attributes. This subset is then encoded using XML [28] and SOAP [29].

The operation of the simplest job-based print session requires one OBEX connection with the Sender as OBEX Client and the Printer as OBEX Server. OBEX GET operations shall be used over this connection to transfer SOAP-encoded messages for job creation, configuration, and management.

Optional OBEX connections shall be opened, if necessary, to expand the print session to include:

1. Real-time job and Printer status, while printing

2. Retrieval of image data, local to the Sender, referenced as part of the print content

3. Retrieval of referenced print content (which may also include referenced image data, local to the Sender or otherwise)

All Printers compliant with the Basic Printing Profile shall provide support for job-based data transfer; however, this support is optional for Senders. Specific details of those parts of the Job-Based Transfer model that are mandatory and optional are discussed in later sections.

Job-based data transfer also contains certain optional features that further enhance the capabilities of the print session for both the Printer and Sender. These features include:

1. Enhanced layout capabilities (Section 7.2)

2. The capability to print externally referenced or rendered content (Section 8)

A detailed binding of application-layer operations to OBEX is discussed in Section 11, as well as the initialization of RFCOMM channels and OBEX connections. The following figure describes the basic structure of the job-based print session:

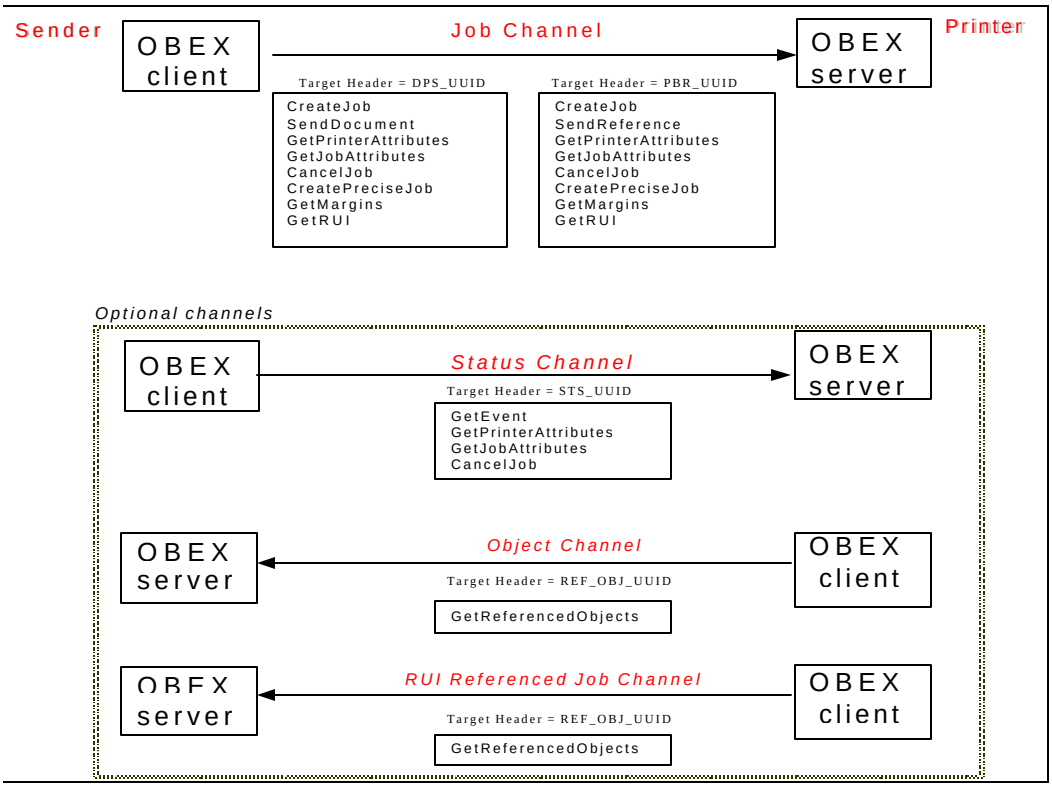

Figure 5: Job-Based Transfer Model Connections

5.1.3 Reflected User Interface-Based Transfer Model

5.1.3.1 RUI for Direct Printing and Reference Printing

Section 9 defines an optional browser-based method of accessing and controlling a Printer or print session that may be used in conjunction with either Direct Printing or the Reference Printing. The use of RUI is optional and can only be supported by suitably configured Senders and Printers.

The Reflected User Interface provides a means to extend and customize all aspects of Printer control or status and allows independent evolution and differentiation of features and services by printer manufacturers without affecting interoperability with Senders.

RUI defines the GetRUI operation that may be targeted to either the Direct Printing or the Reference Printing services independently for return of a control page for those services. A user is then provided with a browser-based interface to the printer for job submission and tracking.

5.1.3.2 RUI for Printer Administration

In addition to browser-based control of the Direct Printing and Reference Printing services, a mechanism is defined for access to a Printer's administration and general configuration control pages. Section 9 details this optional, additional, print job independent service of the Basic Print Profile.

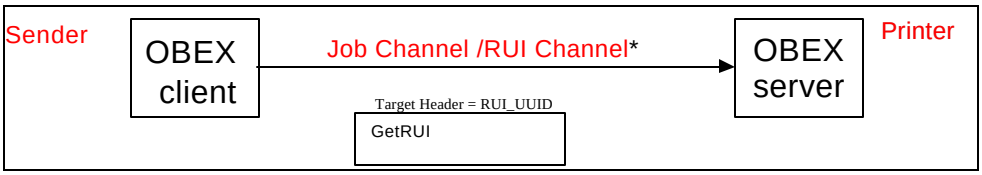

Figure 6: Printer Administration RUI Connections

*Note: The RUI for Printer Administration may also be implemented on an independent channel if the RFCOMM channel designated for the Job Channel in the Basic Printing Service Record is not the same as the RFCOMM channel designated for the ReflectedUI Channel in the Printer Administrative User Interface Service Record.

6 Simple Push Model

To enable very constrained Senders to generate minimal print output, the Printer provides a simple FilePush service. An OBEX PUT request received by the Printer without a preceding CreateJob (indicating job-based data transfer) shall elicit a “best faith” effort on the part of the Printer. (This circumstance can be identified by the Printer in the following manner: an OBEX PUT request has been received, and no CreateJob has preceded it, and the Application Parameters header does not contain a JobId.) The PUT request shall contain an OBEX Type header specifying a document type that is supported by the printer. If the printer does not support the indicated document type, an appropriate OBEX response code (e.g., 0x4F(0xCF) – unsupported media type) shall be returned.

Default printer configuration and job attributes are assumed when print data has not been preceded by a CreateJob operation. **No provisions are made for configuration, job status monitoring, or job control such as cancellation for these minimalistic jobs!** Applications requiring this level of functionality shall use the job-based data transfer application layer.

A Printer that has received a CreateJob operation over a given job channel, but has not received an associated SendDocument operation, may refuse any OBEX PUT request without an Application Parameters header or with an Application Parameters header that does not contain a valid JobId. In this case an appropriate OBEX response code (e.g., 0x43(0xC3) – forbidden) shall be returned in the OBEX response. Section 16.3 lists all OBEX response codes and their associated meaning.

Table 5 contains an example of the OBEX sequences for a FilePush operation. Note that the FilePush may be broken into multiple OBEX PUT packets by the OBEX transport layer if the file being pushed is larger that the maximum packet size for an OBEX packet.

Client Request:	**Bytes**	**Meaning**
Opcode	0x82	**PUT**, Final bit set
	0xnnnn	Length of packet
	0xCB	HI for **Connection Id** header
	0xnnnnnnnn	
	0x42	HI for **Type** header
	0x001C	Length of **Type** header
	image/jpeg	MIME Media-Type of object, null terminated US-ASCII text
	0x01	HI for **Name** header (optional)
	0x0015	Length of **Name** header
	0069 006D 0067 0035 002E 006A 0070 0067 0000	**Name** header content "img5.jpg" (UTF-16 encoded), null terminated
	0x48	Hi for Object **Body** header
	0x0xxx	Length of **Body**
	0x……	Body object bytes of img5.jpg
Server response:		
Response code	0xA0	SUCCESS, Final bit set
	0x0003	Length of response packet

Table 5: FilePush Operation Example

7 Job-Based Transfer

The operations covered by the Job-Based Transfer model of the Basic Printing Profile are summarized in the following table. For reference, a mapping is shown between Basic Printing Profile operations and comparable IPP operations where applicable.

Printer Operation	IPP Operation	Support in Sender	Support in Printer
CreateJob	Create-Job	C1	M
SendDocument	Send-Document	C1	M
GetJobAttributes	Get-Job-Attributes	O	M
GetPrinterAttributes	Get-Printer-Attributes	O	M
CancelJob	Cancel-Job	O	M
GetReferencedObjects	N/A	C2	M
GetEvent	N/A	O	M
CreatePreciseJob	N/A	O	O
GetMargins	N/A	O	O
SendReference	N/A	O	O
GetRUI	N/A	O	O

Table 6: Job -Based Transfer Application Layer Features

C1: A Sender shall support this operation if the Simple Push model is not supported.

C2: A Sender shall support this operation if it sends an object to the Printer that requires the Printer to fetch objects from the Sender.

Job-Based Transfer Application Layer Operation Descriptions

1. **GetPrinterAttributes**: This SOAP action may be used to query printer status and capabilities.

2. **CreateJob**: This SOAP action may be used to submit the job attributes for the print job. The allocated JobId shall be returned.

3. **SendDocument**: This action is used to transfer the actual print content of a print job to the printer. It shall be used in conjunction with a CreateJob action.

4. **GetJobAttributes**: This SOAP action may be used to query specific attributes and status of a specific job.

5. **CancelJob**: This SOAP action may be used to cancel a job using the JobId.

6. **GetReferencedObjects**: This operation may be used by the Printer, on the Object Channel or the RUI Referenced Job Channel, to retrieve referenced objects (content and/or images) from the Sender.

7. **GetEvent**: This SOAP action may be used by the Sender to query status from the printer over the Status Channel while a print job is being sent on the Job Channel.

8. **CreatePreciseJob**: This SOAP action may be used by a Sender as part of the enhanced layout features for the job-based usage model. This operation is described in Section 7.2.

9. **GetMargins**: This SOAP action may be used by a Sender as part of the enhanced layout features for the job-based usage model. This operation is described in Section 7.2.

10. **SendReference**: This operation may be used by the Sender to transfer an Internet/intranet reference to the Printer for printing. This operation is described in Section 8.

11. **GetRUI**: This operation may be used by the Sender to request a “web” page from the Printer for Printer/print job configuration and control. This operation is described in Section 9.

7.1 Job-Based Transfer Application Layer Operations

The following sections describe the operations (and attributes) of the job-based model that shall be supported by a Printer. Operations that are added by optional features (e.g., CreatePreciseJob and GetMargins) are described in the sections detailing the optional feature.

When an attribute is Mandatory on the Printer, the Printer shall be able to at least parse and interpret the attribute; however, the Printer may restrict the values associated with those attributes. For example, the Copies attribute in the CreateJob operation shall be understood by the Printer; however, the Printer is not required to support a value greater than 1 for the MaxCopiesSupported attribute in response to a GetPrinterAttributes operation.

For attributes that are collections or arrays of information, the entire collection or array shall be specified by the Sender or returned by the Printer. Specific array element selection is not supported.

When an attribute is mandatory on the Sender, the Sender shall be able to send the particular attribute in a request, then parse and interpret a mandatory attribute in the response. An attribute can also be mandatory in the Sender as a result of using or requiring an optional application-layer operation. For example, if the Sender chooses to send a print job that contains a reference to a file on the Sender that contains additional print content, the Sender shall be able to support the GetReferencedObjects operation on the Object Channel as well as the associated attributes of the GetReferencedObjects operation.

Since the protocol assumes a well-formed XML encoding, both the Printer and Sender shall be able to parse all attributes in an operation or response, even if it does not understand the meaning of an individual attribute. For example, the GetPrinterAttributes operation may return attributes that are foreign to the Sender; however, the Sender shall still parse the response and act on the attributes that it does understand.

In general, if a Printer receives an attribute that it does not understand or cannot comply with, the Printer shall ignore the attribute and make a “best effort” to print the document data. (The CreatePreciseJob operation is an exception to this general rule.) The Printer shall also indicate this action via the OperationResponse attribute for any operation.

7.1.1 GetPrinterAttributes

7.1.1.1 GetPrinterAttributes Request

This operation is used to request details about the printer's capabilities and status. If the RequestedPrinterAttributes attribute is included in the request, the printer shall respond only with those attributes specifically requested as long as all of those attributes are valid, else the printer shall respond with all attributes that it supports. An OBEX GET request is used to send the GetPrinterAttributes request to the printer. All attributes, with the exception of OperationStatus, defined in the GetPrinterAttributes response may be individually requested. OperationStatus shall be returned as part of every GetPrinterAttributes response.

The GetPrinterAttributes request may contain the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
Array of String	RequestedPrinterAttributes	O	M
String	PrinterAttribute		
	Description: Specifies the specific printer attributes to be reported in the GetPrinterAttributes response.		

Table 7: GetPrinterAttributes Request Attributes

The following example contains a SOAP-encoded GetPrinterAttributes request:

```
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-bluetooth-org:service:Printer:1#GetPrinterAttributes"

<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
  <u:GetPrinterAttributes xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
        <RequestedPrinterAttributes>
              <PrinterAttribute> PrinterName </PrinterAttribute>
              <PrinterAttribute> PrinterState </PrinterAttribute>
              <PrinterAttribute> PrinterStateReasons </PrinterAttribute>
        </RequestedPrinterAttributes>
  </u:GetPrinterAttributes>
 </s:Body>
</s:Envelope>
```

7.1.1.2 GetPrinterAttributes Response

The Printer's response to the GetPrinterAttributes operation is detailed below. Since a Sender may only be interested in specific attributes, all attributes are considered optional for the Sender. However, if the Sender requests an attribute(s) from the Printer using the GetPrinterAttributes operation, it should honor any value restrictions indicated by the Printer in a subsequent CreateJob operation. For the Printer, all attributes are mandatory to the extent that the Printer shall respond with correctly formed and valid information.

The GetPrinterAttributes response contains the following attributes:

Type	Attribute Name	Support in Printer
String	PrinterName	M
	Description: The administratively assigned user-friendly name of the Printer. Printer may return an empty string or a default string if no configuration mechanisms are available (e.g., <*PrinterName*><*/PrinterName*>).	
String	PrinterLocation	M
	Description: Indicates the location of the device. For example, "My Office". Printer may return an empty string or a default string if no configuration mechanisms are available (e.g., <*PrinterLocation*><*/PrinterLocation*>).	
String	PrinterState	M
	Description: Identifies the current state of the printer. Values: • ***idle*** *- new jobs can start processing without waiting.* • ***processing*** *- jobs are processing; new jobs may wait before processing.* • ***stopped*** *- no jobs can be processed and intervention is required.* Printer shall be able to accurately report PrinterState values. (i.e., a Printer is not allowed to always report the *idle* PrinterState value).	
String	PrinterStateReasons	M

Type	Attribute Name	Support in Printer
	Description: Indicates additional information about why the Printer is in its current state. The values shall be formed as follows: <reason>[<severity>] • <reason> is a value from the list below. • <severity> is "report", "warning" or "error". If no severity is given then clients shall assume "error". • ***none*** *- Indicates that there are no current state reasons.* • ***attention-required*** *– Indicates reason other than those listed.* • ***media-jam*** *- The device has a media jam.* • ***paused*** *- Someone has paused the printer and the PrinterState is "stopped". In this state, a Printer may not produce printed output.* • ***door-open*** *- One or more covers on the device are open.* • ***media-low*** *- At least one input tray is low on media.* • ***media-empty*** *- At least one input tray is empty.* • ***output-area-almost-full*** *- One or more output area is almost full: e.g., tray, stacker, collator.* • ***output-area-full*** *- One or more output area is full: e.g., tray, stacker, collator.* • ***marker-supply-low*** *- The device is low on at least one marker supply: e.g., toner, ink, ribbon.* • ***marker-supply-empty*** *- The device is out of at least one marker supply: e.g., toner, ink, ribbon.* • ***marker-failure*** *– The device has at least one marking device which has failed and requires service or replacement: e.g., pen.*	
Array of String	DocumentFormatsSupported	M
String	DocumentFormat	
	Description: Specifies the document formats supported as MIME media-types. The document format may also include a version as described in Section 12.2.2. **Compliance Note for Printer**: As a condition of compliance, the Printer shall return support for XHTML-Print Version 1.0.	
Boolean	ColorSupported	M
	Description: Identifies whether or not the device is capable of full color printing. **Note:** Does not include highlight color (i.e., black and one color), or grayscale rendering of color data.	
I4	MaxCopiesSupported	M

Type	Attribute Name	Support in Printer
	Description: Identifies the largest value of the Copies parameter that the Printer supports in CreateJob actions. (MaxCopiesSupported shall be > 0) **Note for Printers:** It is allowable for the printer to not honor the indicated range of MaxCopiesSupported for very memory-intensive or complex pages. (For example, a printer that can support multiple copies of a simple text job may not be able buffer an entire image page.)	
Array of String String	SidesSupported Sides	M
	Description: Identifies the set of values of the Sides parameter that the Printer supports in the CreateJob action. See Section 7.1.2. CreateJob for a list of possible values.	
I4	NumberUpSupported	M
	Description: Identifies the maximum value of the NumberUp parameter that the Printer supports in the CreateJob action. See Section 7.1.2. CreateJob for a list of possible values. (NumberUpSupported shall be > 0) **Note for Printers:** It is allowable for the printer to not honor the indicated value for NumberUpSupported for complex pages. (For example, a printer that can format a number of simple text impressions on a single page might not be able to support multiple impressions of a complex image.)	
Array of String	OrientationsSupported	M
String	Orientation	
	Description: Identifies the set of values of the OrientationRequested parameter that the Printer supports in CreateJob actions. See Section 7.1.2. CreateJob for a list of possible values.	
Array of String	MediaSizesSupported	M
String	MediaSize	
	Description: Identifies the set of all values of the MediaSize parameter that the Printer can support in CreateJob operations. The Printer does not necessarily have to have this size media installed. Refer to the Media Standardized Names standard [27] for a complete listing of all Media Size Self-Describing Names.	
Array of String String	MediaTypesSupported MediaType	M
	Description: Identifies the types of "paper" that the printer can support. The Printer does not necessarily have to have this type of media loaded. Refer to the Media Standardized Names standard [27] for a complete listing of all Media Type Names.	

Type	Attribute Name		Support in Printer
Array of Arrays	MediaLoaded		M
	Array of String	LoadedMediumDetails	
	String	LoadedMediumSize	
	String	LoadedMediumType	
	Description: Identifies a collection (array of arrays) of medium type/size pairs that are currently loaded in the Printer. This paired list should, in theory, be a subset of the information reported by the printer via the MediaSizesSupported and MediaTypesSupported attributes. **Note for Printers:** If the printer has no way of sensing or configuring media details, it may always return the following: …… <MediaLoaded> <LoadedMediumDetails> <LoadedMediumSize>unspecified</LoadedMediumSize> <LoadedMediumType>unspecified</LoadedMediumType> </LoadedMediumDetails> </MediaLoaded> ……..		
Array of String	PrintQualitySupported		M
String	PrintQuality		
	Description: Identifies the set of values of the PrintQuality parameter that the Printer supports in CreateJob actions. See Section 7.1.2. CreateJob for a list of possible values.		
i4	*QueuedJobCount*		M
	Description: Indicates the number of jobs in the Printer's queue. (QueuedJobCount shall be >= 0) **Note for Printers:** Printers that support other input sources that don't support job tracking may always return zero if idle, and one if printing.		
Array of String	ImageFormatsSupported		M
String	ImageFormat		
	Description: Identifies the types of images, as MIME types, that are supported when embedded in an XHTML-Print document. The image format may also include a version as described in Section 12.2.2. Image formats supported outside of XHTML-Print, as stand-alone document types should also be returned in the DocumentFormatsSupported response. **Compliance Note for Printer**: As a condition of compliance, the Printer shall return support for JPEG as described in Section 10.7.		

Type	Attribute Name	Support in Printer
i4	BasicTextPageWidth	M
	Description: Indicates the number of characters of the default mono-spaced font that fit the width of the page for the “default” paper size. Applicable for XHTML-Print and Basic Text document formats. (BasicTextPageWidth shall be >= 0)	
i4	BasicTextPageHeight	M
	Description: Indicates the number of rows of characters of the default mono-spaced font that can fit on the height of the “default” paper size. Applicable for XHTML-Print and Basic Text document formats. (BasicTextPageHeight shall be >= 0)	
String	PrinterGeneralCurrentOperator *Description:* Provides general information about who to contact (and how) to report Printer support requirements. Printer may return an empty string or a default string if no configuration mechanisms are available (e.g., <*PrinterGeneralCurrentOperator*></*PrinterGeneralCurrentOperator*>).	M
String	OperationStatus	M
	Description: The Status indicates the success or failure of the GetPrinterAttributes operation. Error codes are mapped to their meanings according to the table in Section 16.2.	

Table 8: GetPrinterAttributes Response Attributes

The following example contains a SOAP -encoded response for a GetPrinterAttributes request that did not contain the RequestedAttributes attribute.

```
CONTENT-LENGTH:bytes in body
CONTENT-TYPE:text/xml; charset="utf-8"

<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<s:Body>
  <u:GetPrinterAttributesResponse xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
        <PrinterName>MyPrinter</PrinterName>
        <PrinterLocation>MyLocation</PrinterLocation>
        <PrinterState>idle</PrinterState>
        <PrinterStateReasons>none</PrinterStateReasons>
        <PrinterGeneralCurrentOperator></PrinterGeneralCurrentOperator>
        <DocumentFormatsSupported>
              <DocumentFormat> application/vnd.pwg-xhtml-print+xml:1.0 </DocumentFormat>
              <DocumentFormat> application/vnd.hp-PCL:5E</DocumentFormat>
              <DocumentFormat> text/plain</DocumentFormat>
              <DocumentFormat> application/PostScript:3</DocumentFormat>
        </DocumentFormatsSupported>
        <ImageFormatsSupported>
              <ImageFormat>image/jpeg</ImageFormat>
              <ImageFormat>image/gif</ImageFormat>
        </ImageFormatsSupported>
        <ColorSupported>true</ColorSupported>
        <MaxCopiesSupported>1</MaxCopiesSupported>
        <SidesSupported>
              <Sides> one-sided</Sides>
              <Sides> two-sided-long-edge</Sides>
              <Sides> two-sided-short-edge</Sides>
        </SidesSupported>
        <NumberUpSupported>4</NumberUpSupported>
        <OrientationsSupported>
              <Orientation>portrait</Orientation>
              <Orientation>landscape</Orientation>
        </OrientationsSupported>
        <MediaSizesSupported>
              <MediaSize>iso_a4_210x297mm</MediaSize>
              <MediaSize>iso_a3_297x420mm</MediaSize>
        </MediaSizesSupported>
        <MediaTypesSupported>
              <MediaType>stationery</MediaType>
              <MediaType>photographic</MediaType>
              <MediaType>cardstock</MediaType>
        </MediaTypesSupported>
        <MediaLoaded>
              <LoadedMediumDetails>
                    <LoadedMediumSize> iso_a4_210x297mm</LoadedMediumSize>
                    <LoadedMediumType>stationery</LoadedMediumType>
              </LoadedMediumDetails>
              <LoadedMediumDetails>
                    <LoadedMediumSize> iso_a4_210x297mm</LoadedMediumSize>
                    <LoadedMediumType>photographic</LoadedMediumType>
              </LoadedMediumDetails>
```

```
                    </MediaLoaded>
                    <PrintQualitySupported>
                            <PrintQuality>draft</PrintQuality>
                            <PrintQuality>normal</PrintQuality>
          </PrintQualitySupported>
          <QueuedJobCount>1</QueuedJobCount>
                    <OperationStatus>0x0000</OperationStatus>
           </u:GetPrinterAttributesResponse>
</s:Body>
 </s:Envelope>
```

7.1.2 CreateJob

This operation is used to configure a print job.

7.1.2.1 CreateJob Request

The OBEX GET request shall be used to send the SOAP action, in the OBEX Body header, to the Printer. The CreateJob request uses the OBEX Type header (x-obex/bt-SOAP) to indicate a SOAP transaction using XML.

For the Printer, all attributes of the CreateJob request shall be parsed and interpreted. The Printer should also adhere to the attributes and their values as defined by the Sender.

The CreateJob request contains the following XML-encoded attributes:

Type	Attribute Name	Support in Sender	Support in Printer
String	JobName	O	M
	Description: The user-friendly name of the specified job.		
String	JobOriginatingUserName	O	M
	Description: The name of the user that submitted the specified job; supplied either by the client or by the security infrastructure, if any.		
String	DocumentFormat	O	M
	Description: Specifies the document format of the current job as a MIME media-type and any applicable version.		
I4	Copies	O	M
	Description: Specifies the number of copies of the current job to be printed.		
String	Sides	O	M

Type	Attribute Name	Support in Sender	Support in Printer
	Description: Specifies how pages are to be imposed upon the sides of a selected medium for the current job. Values: • *one-sided* • *two-sided-long-edge* • *two-sided-short-edge*		
I4	NumberUp	O	M
	Description: Indicates the number of print-stream pages to impose upon a single side of an instance of a selected medium for the current job. (NumberUp shall be > 0) Examples: ***1*** **=** *One page per side.* ***2*** **=** *Two pages per side.* ***4*** **=** *Four pages per side.*		
String	OrientationRequested	O	M
	Description: Indicates the desired orientation for printed pages of the current job. Values: • *portrait* • *landscape* • *reverse-landscape* • *reverse-portrait*		
String	MediaSize	O	M
	Description: Identifies the size of medium to use for the current job. Refer to the Media Standardized Names standard [27] for a complete listing of all Media Size Self-Describing Names.		
String	PrintQuality	O	M
	Description: Specifies the print quality requested for the current job. Values: • *draft* • *normal* • *high* **Note:** Because of the vendor dependencies in print quality metrics (dots per inch, etc.), print quality descriptions are kept intentionally generic.		
String	MediaType	O	M

Type	Attribute Name	Support in Sender	Support in Printer
	Description: Identifies the type of medium to be used for the current job. Refer to the Media Standardized Names standard [27] for a complete listing of all Media Type Names.		
Boolean	CancelOnLostLink	O	M
	Description: Indicates that this job, initiated by the Sender, shall be cancelled by the Printer if the Bluetooth Radio link is lost or closed. **Note for Printers:** If not set by the Sender, the action taken by the Printer in the event of a lost or closed radio link is unspecified.		

Table 9: CreateJob Request Attributes

The following example contains a SOAP-encoded CreateJob request:

```
CONTENT-LENGTH: bytes in body
CONTENT-LANGUAGE: en-US
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-bluetooth-org:service:Printer:1#CreateJob"

<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
    <u:CreateJob xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
            <JobName> MyJob</JobName>
            <JobOriginatingUserName>MyName</JobOriginatingUserName>
            <DocumentFormat> application/PostScript:3</DocumentFormat>
            <Copies >1</Copies >
            <Sides> one-sided</Sides >
            <NumberUp> 1</NumberUp>
            <OrientationRequested> portrait</OrientationRequested>
            <MediaSize> iso_a4_210x297mm</MediaSize>
            <MediaType>cardstock </MediaType >
            <PrintQuality> normal</PrintQuality>
            <CancelOnLostLink> true</CancelOnLostLink >
    </u:CreateJob>
 </s:Body>
 </s:Envelope>
```

Note: The Content-Language entity header field may be used as an indication to the Printer of the preferred language of the Sender. The syntax and content of this header follows the HTTP 1.1 Specification [30].

7.1.2.2 CreateJob Response

The CreateJob response contains the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
I4	JobId	M	M
	Description: The job identifier of the job for which the Printer can accept print data in a subsequent SendDocument. (JobId shall be > 0) Note for Senders: If the CreateJob operation is used, the Sender shall use the JobId in the subsequent SendDocument operation.		
String	OperationStatus	O	M
	Description: The Status indicates the success or failure of the CreateJob operation. Error codes are mapped to their meanings according to the table in Section 16.2.		

Table 10: CreateJob Response Attributes

The JobId shall be used by one and only one SendDocument operation (in the Application Parameters header) to send a print job to the printer. This print job shall be printed using the configuration established by the CreateJob operation. The JobId shall also be used by the GetJobAttributes and CancelJob operations to monitor job progress or cancel the job, respectively.

The following example contains a SOAP-encoded CreateJob response:

```
CONTENT-LENGTH:  bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"

<s:Envelope
   xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
   <u:CreateJobResponse xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
            <JobId> 12345 </JobId>
            <OperationStatus>0x0001</OperationStatus>
 </u:CreateJobResponse>
 </s:Body>
</s:Envelope>
```

The JobId shall also be returned in the Application Parameters header in the OBEX GET response. This enables simple devices to extract the JobId without requiring an XML parser. The syntax of Application Parameters header entries is defined in Section 11.12.

7.1.3 SendDocument

The SendDocument action is used to send the print data. For each CreateJob, one and only one SendDocument action shall be issued. The OBEX Type header shall specify the document format as a MIME media-type, and the print data shall be

included in the OBEX Body header. Refer to Section 10 for more details. The JobId shall be included in the Application Parameters header as defined in Section 11.12.

7.1.4 GetJobAttributes

This operation is used to get information about a specific job.

7.1.4.1 GetJobAttributes Request

The OBEX GET request shall be used to send the GetJobAttributes request to the Printer. If the RequestedJobAttributes attribute is included in the request, the Printer shall respond only with those attributes specifically requested as long as all of those attributes are valid, else the Printer shall respond with all attributes that it supports. An OBEX GET request is used to send the GetJobAttributes request to the Printer. All attributes may be individually requested, with the exception of OperationStatus described in Section 7.1.4.2. OperationStatus shall be returned as part of every GetJobAttributes response.

The GetJobAttributes request contains the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
I4	JobId *Description:* The job identifier indicates the print job whose attributes the Printer is to return.	M	M
Array of String	RequestedJobAttributes	O	M
String	JobAttribute		
	Description: Specifies the specific attributes to be reported in the GetJobAttributes response.		

Table 11: GetJobAttributes Request Attributes

The following example contains a SOAP-encoded GetJobAttributes request:

```
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-bluetooth-org:service:Printer:1#GetJobAttributes"

<s:Envelope
   xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
   <u:GetJobAttributes xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
          <JobId>12345</JobId>
          <RequestedJobAttributes>
```

```
                    <JobAttribute> JobState</JobAttribute >
                </RequestedJobAttributes >
            </u:GetJobAttributes>
        </s:Body>
    </s:Envelope>
```

7.1.4.2 GetJobAttributes Response

The Printer's response to the GetJobAttributes operation is detailed below. All attributes are optional for the Sender with the exception of the JobId attribute. The Sender shall use the JobId as an identifier to the submitted print job. For the Printer, all attributes are mandatory to the extent that the Printer shall respond with correctly formed and valid information.

The GetJobAttributes response contains the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
I4	JobId	M	M
String	JobState *Description: One of the following:* • ***printing –*** *The job currently printing.* • ***waiting*** *– The Job is waiting to print.* • ***stopped*** *– The job is stopped.* • ***completed*** *– The job has printed.* • ***aborted*** *– The job was aborted.* • ***cancelled*** *– The job was cancelled.* • ***unknown*** *– The job state is unknown (i.e., unrecognized or no longer valid JobId).* The Printer, or printing device, shall accurately report the state of the Job; however, all job states are not explicitly required to be supported. (For example, an external print server (smart parallel port dongle that supports BPP) might be unable to report Job Completed with accuracy when attached to a printer that only supports IEEE1284 Compatibility mode on its parallel port.)	O	M
String	JobName	O	M
	Description: The user-friendly name of the specified job. Printer may return an empty string if this attribute was not set in the CreateJob operation by the Sender (e.g., <JobName></JobName>).		
String	JobOriginatingUserName	O	M
	Description: The name of the user that submitted the specified job; supplied either by the client or by the security infrastructure, if any. Printer may return an empty string if this attribute was not set in the CreateJob operation by the Sender (e.g., <JobOriginatingUserName></JobOriginatingUserName>).		

Type	Attribute Name	Support in Sender	Support in Printer
I4	JobMediaSheetsCompleted	O	M
	Description: The media sheets completed for the specified job so far. (JobMediaSheetsCompleted shall be >= 0) Printer may always return zero.		
I4	NumberOfInterveningJobs	O	M
	Description: The number of unique print jobs that may be printed ahead of the print job specified by the JobId. (NumberOfInterveningJobs shall be >= 0)		
String	OperationStatus	O	M
	Description: The Status indicates the success or failure of the GetJobAttributes operation. Error codes are mapped to their meanings according to the table in Section 16.2.		

Table 12: GetJobAttributes Response Attributes

The following example contains a SOAP-encoded GetJobAttributes response:

```
CONTENT-LENGTH:  bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"

<s:Envelope
   xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
   <u:GetJobAttributesResponse xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
            <JobId> 12345 </JobId>
            <JobState> printing </JobState>
             < JobName> MyExpenseReport </JobName>
            <JobOriginatingUserName> MyName </JobOriginatingUserName>
            <JobMediaSheetsCompleted> 2 </JobMediaSheetsCompleted>
            <NumberOfInterveningJobs> 0 </NumberOfInterveningJobs>
            <OperationStatus> 0x0000 </OperationStatus>
  </u:GetJobAttributesResponse>
  </s:Body>
</s:Envelope>
```

7.1.5 CancelJob

This operation is used to cancel a particular print job. The OBEX GET request shall be used to send the SOAP-encoded request to the Printer. For the Printer, this operation is mandatory to the extent that the job should be cancelled if possible.

Note: A Printer should honor the CancelJob operation only for jobs (JobIds) originated by the canceling Sender. Furthermore, a Sender should initiate the CancelJob operation only for jobs it submitted.

7.1.5.1 CancelJob Request

The CancelJob request contains the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
I4	JobId	M	M
	Description: The job identifier of the job which the Printer should cancel.		

Table 13: CancelJob Request Attributes

The following example contains a SOAP-encoded CancelJob request:

```
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-bluetooth-org:service:Printer:1#CancelJob"

<s:Envelope
   xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
   <u:CancelJob xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
            <JobId>123465</JobId>
    </u:CancelJob>
 </s:Body>
</s:Envelope>
```

7.1.5.2 CancelJob Response

The CancelJob response contains the following attributes:

Type	Attribute Name	Support in Printer
I4	JobId	M
	Description: The job identifier of the print job, for which the Printer is returning the CancelJob response.	
String	OperationStatus *Description:* The Status indicates the success or failure of the CancelJob request. Error codes are mapped to their meanings according to the table in Section 16.2.	M

Table 14: CancelJob Response Attributes

The following example contains a SOAP-encoded CancelJob response:

```
CONTENT-LENGTH:  bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"

<s:Envelope
   xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
   <u:CancelJobResponse xmlns:u="um:schemas-bluetooth-org:service:Printer:1">
            <JobId> 12345 </JobId>
            <OperationStatus>0x0406</OperationStatus>
   </u:CancelJobResponse>
 </s:Body>
</s:Envelope>
```

7.1.6 GetReferencedObjects

Some print data formats (e.g., XHTML-Print) can require the Printer be able to retrieve objects or parts of objects from the Sender. Typical examples of objects that the Printer retrieves from the Sender are image(s) that need to be interleaved with text, or externally referenced style sheets. In the Basic Printing Profile this is supported by establishing a separate OBEX connection (Object Channel), with the Printer as OBEX Client and the Sender as OBEX Server. The only operation that shall be allowed on the Object Channel is the GetReferencedObjects operation.

Printing using the Reflected User Interface option requires the Printer to retrieve all print content from the Sender, not just referenced image content. In this scenario, two additional OBEX Channels shall be opened: one for the print stream (RUI Referenced Job Channel) and one for any referenced objects (Object Channel) that are part of that print stream. The only operation that shall be allowed on either the Object Channel or RUI Referenced Job Channel is the GetReferencedObjects operation.

7.1.6.1 GetReferencedObjects Request

For the Printer, all mandatory attributes shall be included in the GetReferencedObjects operation (FileSize is optional). For the Sender, this operation and all its attributes are mandatory if the print content being sent by the Sender contains objects to be retrieved or if a Remote User Interface (RUI) is used to configure and control the job.

This operation has the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
String	ObjectIdentifier	M	M
	Description: The identifier of the object to be retrieved. This identifier shall be referenced to the Sender. Any routing to object sources outside the Sender shall be handled by the Sender.		
i4	Offset	M	M
	Description: The byte offset into the image or print object. The first byte of the image is byte zero. (Offset shall be >= 0)		
s4	Count	M	M
	Description: The number of bytes to send. If the Count is –1, the remainder of the image or print object (from the Offset) is returned. (Count shall be >= -1)		
s4	FileSize	M	O
	Description: The size of the referenced file in bytes. (FileSize shall be >= -1)		

Table 15: GetReferencedObjects Attributes.

Section 11.4.1.5 describes the use of OBEX for pulling referenced content.

The ObjectIdentifier attribute is encoded in the Name header. It must therefore be translated from the UTF-8 encoding in which it was received by the Printer, into the UTF-16 encoding required by the OBEX Name header.

Section 11.12 defines the encoding of the Offset, Count and FileSize attributes in the OBEX Application Parameters header. The value associated with the FileSize tag is ignored in the request. The presence of the FileSize tag in the request indicates to the Sender that it should return the size of the requested object (in bytes) in the Application Parameters header of the response. If the size of the object is unknown, the Sender shall return the FileSize tag with a value of –1 in the Application Parameters header of the response.

The Printer may issue a GetReferencedObjects request with a Count of zero. In this case, an empty OBEX Body header is returned by the Sender.

If the requested Count is larger than the number of bytes remaining in the object or the number of bytes available, the available bytes shall be returned in the Body header of the OBEX response. The “Final Bit” of the OBEX response may not be set in cases where more data will be available (e.g., a case where the Sender is streaming data to the Printer from another source).

If more than one referenced object is contained in the print stream, the objects may be retrieved either sequentially (i.e., all of one image then all of the next) or concurrently (i.e., parts of either image in arbitrary order). The printer may also request an object, or part of an object, more than once.

In the case that the Sender is not able to honor the Printer’s request for referenced content (i.e., that part of the image or file does not exist or is currently unavailable), then the Sender shall respond with an appropriate OBEX response code, listed in Section 16.3.

Table 16 contains an example of the OBEX sequences for two consecutive GetReferencedObjects operations.

First GetReferencedObjects Operation

Client Request:	Bytes	Meaning
Opcode	0x83	**GET**, Final bit set.
	0xnnnn	Length of packet.
	0xCB	HI for **Connection Id** header.
	0xnnnnnnnn	
	0x42	HI for **Type** header.
	0x001B	Length of **Type** header.
	x-obex/ referencedobject	Type of object, null terminated US-ASCII text.
	0x01	HI for **Name** header.
	0x0015	Length of **Name** header.
	0069 006D 0067	**Name** header content “img5.jpg”
	0035 002E 006A	(UTF-16 encoded), null terminated.
	0070 0067 0000	
	0x4C	HI for **Application Parameters** header.
	0x0015	Length of **Application Parameters** header.
	01 04 00000000	Tag 1 (Offset = 0)
	02 04 00000400	Tag 2 (Count = 1024 Bytes)
	04 04 00000000	Tag 4 (FileSize request)
Server Response:		
Response code	0xA0	SUCCESS, Final bit set.
	0xnnnn	Length of response packet.
	0x4C	HI for **Application Parameters** header.
	0x0009	Length of **Application Parameters** header.
	04 04 FFFFFFFF	Tag 4 (FileSize = unknown)
	0x48	Hi for Object **Body** header.
	0x0400	Length of **Body** header.
	0x……	Body object first 1024 bytes of img5.jpg.

Second GetReferencedObjects Operation

Client Request:		
Opcode	0x83 0xnnnn 0xCB 0xnnnnnnnn 0x42 0x001B *x-obex/ referencedobject* 0x01 0x0015 0069 006D 0067 0035 002E 006A 0070 0067 0000 0x4C 0x000F 01 04 00000400 02 04 000003FF	**GET**, Final bit set. Length of packet. HI for **Connection Id** header. HI for **Type** header. Length of **Type** header. Type of object, null terminated US-ASCII text. HI for **Name** header. Length of **Name** header. **Name** header content “img5.jpg” (UTF-16 encoded), null terminated. HI for **Application Parameters** header. Length of **Application Parameters** header. Tag 1 (Offset = 400) Tag 2 (Count = 1023 Bytes)
Server Response:		
Response Code	0xA0 0xnnnn 0x48 0x0400 0x……	SUCCESS, Final bit set. Length of response packet. Hi for Object **Body** header. Length of **Body** header. Body object second 1024 bytes of img5.jpg.
…..		

Table 16: GetReferencedObjects Operation OBEX Mapping

7.1.7 GetEvent (Event Notification)

This operation is used by the Sender to allow the Printer to provide notification of changes in its status and changes in the status of the job that is identified by the JobId. Arbitrary job monitoring is not supported.

In the Basic Printing Profile this is supported using a separate OBEX channel (Status Channel). When the Sender issues the GetEvent request, the Printer shall respond with the current state of all attributes listed in Table 18. This operation remains active (i.e., the OBEX response code is CONTINUE with the Final bit set) until aborted by the Sender. The Printer subsequently shall provide a complete response each time one or more of the attributes changes.

Note: OBEX transport layer implementations may require that timeouts be disabled to support this operation.

7.1.7.1 GetEvent Request

The GetEvent request contains the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
I4	JobId	M	M
	Description: The job identifier of the job for which the Printer is to return the job variables.		

Table 17: GetEvent Request Attributes

The following example contains a SOAP-encoded GetEvent request:

```
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-bluetooth-org:service:Printer:1#GetEvent"

<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
  <u:GetEvent xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
        <JobId>123465</JobId>
  </u:GetEvent>
 </s:Body>
</s:Envelope>
```

7.1.7.2 GetEvent Response

The GetEvent response shall contain the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
I4	JobId	M	M
String	JobState	O	M
	Description: See Section 7.1.4 GetJobAttributes		
String	PrinterState	O	M
	Description: See Section 7.1.1 GetPrinterAttributes		
String	PrinterStateReasons	O	M
	Description: See Section 7.1.1 GetPrinterAttributes		
String	OperationStatus	O	M
	Description: The Status indicates the success or failure of the GetEvent operation. Error codes are mapped to their meanings according to the table in Section 16.2.		

Table 18: GetEvent Response Attributes

The following example contains a SOAP-encoded GetEvent response:

```
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"

<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
  <u:GetEventResponse xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
          <JobId>12345</JobId>
          <JobState>stopped</JobState>
          <PrinterState>stopped</PrinterState>
          <PrinterStateReasons>media-empty</PrinterStateReasons>
          <OperationStatus>0x0000</OperationStatus>
   </u:GetEventResponse>
 </s:Body>
</s:Envelope>
```

7.2 Enhanced Layout (Optional)

If support for Enhanced Layout is indicated in the Basic Printing Service Record then all of the functions described in this section are mandatory for the Printer. The features and operations defined by the Enhanced Layout option are optional for the Sender to implement. However, if implemented in the Sender, the operations shall be used as defined.

7.2.1 Abstract

Enhanced Layout is a set of features that provides the Sender with additional control over the presentation of content to be printed. This feature is intended to address usage cases such as the printing of multiple images on a page, with precise sizing, positioning, rotation, etc.

7.2.2 Requirements

In addition to all other requirements of this Profile, printers that support this feature shall meet the following requirements:

- Provide support for the "Enhanced Layout Extension", as defined within the XHTML-Print specification [22].
- Provide support for two additional operations, “CreatePreciseJob” and “GetMargins” as defined below.
- Implement at least the “portrait” and “landscape” values of the OrientationRequested parameter to CreateJob or CreatePreciseJob actions.

7.2.3 Additional Operations (CreatePreciseJob and GetMargins)

The following operations are added to support this option.

7.2.3.1 CreatePreciseJob

This operation is used to configure a print job. Its definition and usage are exactly as that defined for CreateJob in Section 7.1.2, except that all Job attributes shall be honored.

If the Printer is able to determine that it cannot produce the job as requested when the CreatePreciseJob operation is processed, the OperationStatus returned is 0x40b (client-error-attributes-or-values-not-supported). If the printer discovers during the rendering or printing of the job that it cannot produce the document as requested, it shall abort printing the job at the earliest opportunity. It should return an appropriate OBEX response code (e.g., 0x51 – Not implemented) to an associated SendDocument operation.

The following example contains a SOAP-encoded CreatePreciseJob operation:

```
CONTENT-LENGTH: bytes in body
CONTENT-LANGUAGE: language used
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-bluetooth-org:service:Printer:1#CreatePreciseJob"

<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
    <u:CreatePreciseJob xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
            <JobName> MyPhotoPage </JobName>
            <JobOriginatingUserName>SuesCamera</JobOriginatingUserName>
            <DocumentFormat> application/vnd.pwg-xhtml-print+xml:1.0</DocumentFormat>
            <Copies>1</Copies>
            <Sides> one-sided </Sides>
            <NumberUp> 1</NumberUp>
            <OrientationRequested> landscape </OrientationRequested>
            <MediaSize>na_letter_8.5x11in</MediaSize>
            <PrintQuality> high</PrintQuality>
    </u:CreatePreciseJob>
 </s:Body>
 </s:Envelope>
```

7.2.3.2 GetMargins

This operation is used to obtain margin information for a particular medium size and type. This enables the Sender to learn the exact dimensions and location of the printable area for a particular Printer on a particular medium size and type, to

facilitate exact page layout. The OBEX GET request is used to send the GetMargins request to the Printer. Support for this operation is optional for the Sender to use but mandatory for a Printer that supports Enhanced Layout.

The GetMargins request contains the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
String	MediaSize	M	M
	Description: The size of the medium for which margin information is requested. Refer to the Media Standardized Names standard [27] for a complete listing of all Media Size Self-Describing Names.		
String	MediaType	M	M
	Description: The type of the medium for which margin information is requested. Refer to the Media Standardized Names standard [27] for a complete listing of all Media Type Names.		

Table 19: GetMargins Request Attributes

The following example contains a SOAP-encoded GetMargins request:

```
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"
SOAPACTION: "urn:schemas-bluetooth-org:service:Printer:1#GetMargins"

<s:Envelope
   xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
   s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
   <u:GetMargins xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
            <MediaSize>na_letter_8.5x11in</MediaSize>
            <MediaType>photographic</MediaType>
    </u:GetMargins>
 </s:Body>
</s:Envelope>
```

7.2.3.3 GetMargins Response

The GetMargins response contains the following attributes:

Type	Attribute Name	Support in Sender	Support in Printer
String	Margins	M	M
	Description: The string may take one of two forms: "Unsupported" indicates that the printer does not support the size of medium specified for the type of medium specified. (For example, the printer may not support a size of "jpn_hagaki_100x148mm" with a type of "transparency".) Otherwise the string identifies the margin sizes defining the printable area for the MediaSize and MediaType specified in the request. The margins shall be provided as a comma-separated list of four values with no spaces anywhere. Each value shall not have leading zeroes. Each value may include a non-zero decimal fraction set off by a period (.). Each of the four values is separated by a comma (,) and the order of the values indicates Top margin, Right margin, Bottom margin, and Left margin. All media are assumed to be portrait for purposes of defining Top, Right, Bottom and Left. Each value shall include the Inch or Millimeter dimension indicator: 'in' or mm', respectively, immediately after each dimension. The value indicates the displacement from the associated edge in the indicated units.		
String	OperationStatus *Description:* The Status indicates the success or failure of the GetMargins operation. Error codes are mapped to their meanings according to the table in Section 16.2.	O	M

Table 20: GetMargins Response Attributes

The following example contains a SOAP-encoded GetMargins response:

```
CONTENT-LENGTH: bytes in body
CONTENT-TYPE: text/xml; charset="utf-8"

<s:Envelope
  xmlns:s="http://schemas.xmlsoap.org/soap/envelope/"
  s:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
 <s:Body>
  <u:GetMarginsResponse xmlns:u="urn:schemas-bluetooth-org:service:Printer:1">
        <Margins>.25in,.25in,0in,.25in</Margins>
        <OperationStatus>0x0000</OperationStatus>
 </u:GetMarginsResponse>
 </s:Body>
</s:Envelope>
```

8 Print-By-Reference (Optional)

If support for Reference Printing (PBR) is indicated in the Basic Printing Service Record then all of the functions and operations described in this section are mandatory for the Printer. Print-By-Reference functions and operations are optional for the Sender to implement. However, if implemented in the Sender the operations shall be implemented as defined.

8.1 Abstract

Print-By-Reference is the ability for a Bluetooth-enabled Sender to originate and control a print job where the information to be printed resides in the network. The printer uses a reference passed to it by the Sender to resolve and access the information to be printed.

8.2 Scope

The general architecture of the Print-By-Reference model is shown in Figure 7. The primary scope of this section is to specify interface A, which supports the passing of reference and control information between the Sender and the Printer.

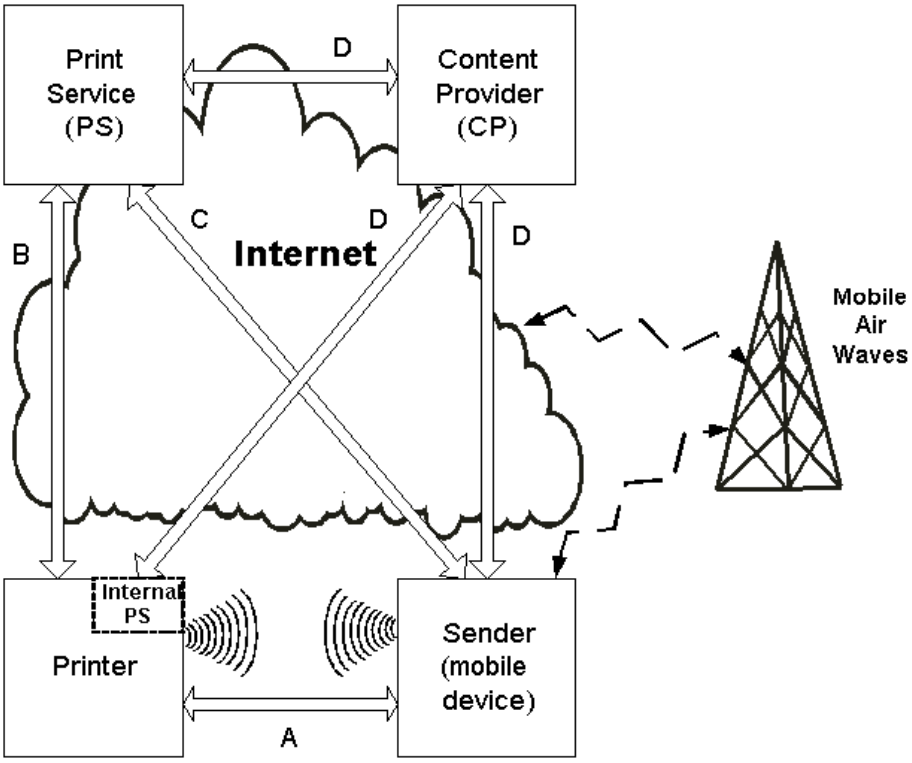

Figure 7: Print-By-Reference System Architecture

Much of the functionality involved in actually printing a reference occurs via other network services. In order to provide a complete system-level description of a Print-By-Reference feature the interfaces B, C, and D also need to be considered. These interfaces have little to do with Bluetooth connectivity except that interface C could be used to stream rendered (print-ready) content to the Sender and then forward to the Printer using either the Simple Push or Job-Based Data Transfer models of BPP. The functional details of the B, C, and D interfaces are outside of the scope of this Profile.

8.3 General PBR Model Description

Print-By-Reference supports the passing of a network reference to a Printer, which then processes this reference in a manner specific to the implementation of the Printer. This interaction model introduces the notion of a Print Portal, where a Printer provides access to a local or Internet-based Print Service that supports printing for Senders without native printing capability. The basic architecture of such an interaction is shown in Figure 8.

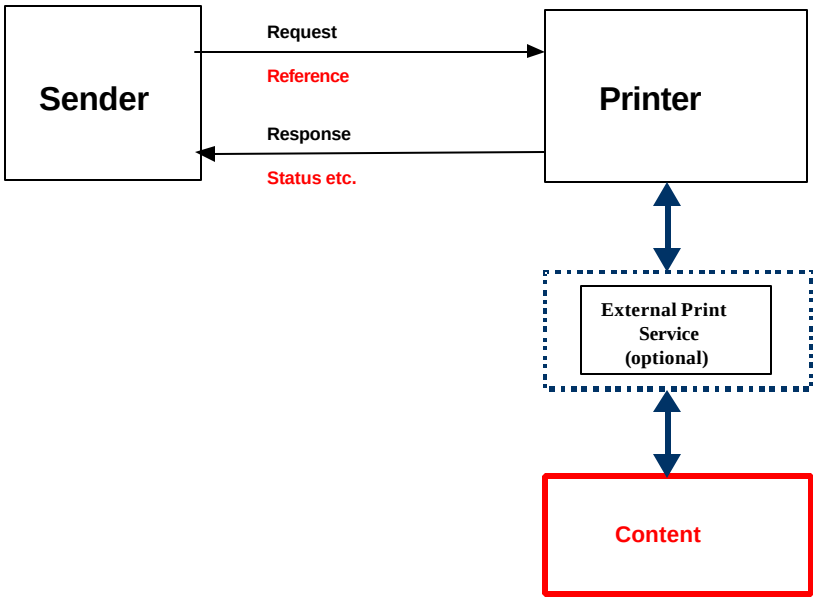

Figure 8: Simple Reference Exchange

The protocol shown above is a basic request response protocol. The reference is sent as part of the request. The Printer processes this reference and returns a response to the Sender. The Printer may either process the reference locally or access a network/Internet-based Print Service to process the request. The Print Service might in turn invoke numerous other services to process the request.

The interactions in PBR are similar to those described in the previous sections; the reference request may be a simple reference push, a job preceded by a CreateJob, or utilize the Reflected User Interface (RUI) service described in Section 9.

The reference is accessed on the network according to the protocol scheme specified in the URI, which is included as part of the reference. The HTTP protocol shall be supported for PBR. Support for other protocols (HTTPS, TLS, FTP, etc.) is optional. If the Printer does not support a protocol specified in a URI, an error is returned.

Since the reference can point to completely arbitrary information, it is not practical to enumerate or mandate all possible data formats that could be encountered in a PBR reference. It is recommended, however, that the Printer (or Print Service) support the following common data formats:

- HTML 4.0
- XHTML 1.0 plus CSS 2
- PDF 1.3
- US ASCII

The Printer shall provide a best-effort attempt to print the information referred to in a reference. If there is information that cannot be printed (e.g., an HTML page that includes an unsupported image type), the Printer should provide at least a partial printout of the information.

8.4 Reference Formats

For Print-By-Reference, three different reference formats are defined.

8.4.1 Simple Reference

The Printer shall be able to deal with a request that only contains a URL. For example, the following URL is considered a simple reference:

http://www.bluetooth.org/

Simple references shall be fully qualified and contain the protocol scheme (HTTP, HTTPS, FTP, etc.) to access the resource referred to by the reference.

8.4.2 XML Reference

The XML reference specification may be used to support the inclusion of meta-information that is passed along with the URL. This meta-information may be used to improve the descriptive qualities of the reference (e.g., provide security hints, alternative access paths etc.).

The XML reference is essentially an XML tag with a specific syntax. The tags allow the reference to be directly embedded in XML documents and may be composed to form a list of references. The general syntax of the tag is:

```
<reference url="scheme://host/resourcepath"
      attr1="attr1val" attr2="attr2val"...
      attrn="attrnval" />
```

All attributes are optional except for the URL that shall be included and fully qualified. Table 21 defines the attributes of the XML reference.

Attribute Name	Comment
label	Descriptive name of the resource referred to by the reference (e.g., label="Sales Document").
url	Fully qualified Uniform Resource Locator. See RFC 2396, [34].
use_proxy	Fully qualified URL that represents an HTTP proxy service that shall be used to access the content referred to by the reference. The Port for the proxy shall be specified in the URL if not port 80 (e.g., use_proxy="http://corp_proxy:8088").
use_service	Fully qualified URL that represents the location of a remote Print Service that shall be used to process the reference (e.g., use_service=" http://myprintsvc:456").
ip_addr	Internet Address of the Sender. Provides a hint to the Printer or Print Service that the Sender is an Internet device. Examples: • ip_addr="126.34.07.6" • ip_addr="::126.34.07.6" • ip_addr="FC01:0:0:0:0:0:0:23"
type	Encoding of the resource referred to by the reference (e.g., type="text/html").
cookie	Cookie that can be used by the receiver when processing the reference. This attribute uses the general form of the HTTP Cookie MIME header NAME1=OPAQUEVALUE; NAME2=OPAQUEVALUE (e.g., cookie="CUSTOMER=WILE_E_COYOTE; PART_NUMBER=ROCKET_LAUNCHER_0001").
on401	Provides the receiver with information that can be used when an HTTP 401 security challenge is encountered while processing the reference. The format of this attribute is that of the HTTP Authorization header defined in RFC 2617, [33]. For example, a credential that uses the "Basic" security scheme with the user ID Aladdin and password "open sesame" would be encoded as follows: (e.g., on401="Basic QWxhZGRpbjpvcGVuIHN1c2FtZQ==").
on407	Same as on401 except used in response to an HTTP 407 proxy challenge. The format is the same as on401.

Attribute Name	Comment
time	Used to signify the time the reference was sent to the Printer. Time is a byte sequence that gives the object's UTC date/time of last modification in ISO 8601 format. Local times should be represented in the format YYYYMMDDTHHMMSS and UTC time in the format YYYYMMDDTHHMMSSZ. The letter "T" delimits the date from the time. UTC time is identified by concatenating a "Z" to the end of the sequence. When possible, UTC times should be used. See preferred time format in [8].
key	The key attribute may be used to associate security information with the reference. This may be used when challenged for security information. In general such keys should be encrypted and time-sensitive to prevent arbitrary delegation from compromising security. The format of this attribute conforms to the Authorization field in RFC 2617, [33] and contains both the security scheme and the encrypted key. This format is the same as on401.
on[status code]	This status code allows the sender to provide the receiver with an alternative URL to process in the event of a particular HTTP status code (e.g., on400="http://w ww.bluetooth.org"). The URL specified would be used in the event of an HTTP 400 status code.
new_sheet	This Boolean attribute indicates to the Printer/Print Service that printing of the referenced document shall start on a new sheet of media. For example, new_sheet="false" indicates that the document should not start a new sheet. If not explicitly stated, it is assumed new_sheet="true". **Note:** The new_sheet attribute is useful only for non-initial XML references in a reference list. A document referenced in the first XML reference in a list or standalone XML reference will start on a new sheet of media.
billing_code	This attribute is used to coordinate billing transactions with specific PBR jobs. The billing_code attribute is likely to be used by mechanisms that are outside of the Basic Print Profile (e.g., billing_code="approved123").

Table 21: XML Reference Attributes

8.4.2.1 Print Reference Hyperlink

The following URI specification illustrates the special form of a URI, which is used to support PBR hyperlinks in web pages:

```
PRINT:<reference url="http://www.print.bluetooth.org/photo10.jpg" cookie="xyzcv1234" />
```

This type of hyperlink is used for embedding PBR references into web pages. A content service may use this format to add printable links to the web content they provide. When the user selects this kind of link, a special function is invoked on the

browser to process the reference. This kind of hyperlink offers a number of advantages to both the user and Content Provider.

The Content Provider can support access to protected information without requiring the explicit transfer of trust to the printer. This is achieved because the access credentials and/or cookies required to access the content can be encoded in the reference specification as defined in Section 8.4.2. A Print hyperlink can be generated when a page is requested and, as such, can include time-based information or time-sensitive information or support one-time use references. The Content Provider can also control the processing of the information into a printable form.

For example:

A user browses to a content service that sells stock images via the Internet. The user logs into the service and browses the images available. After purchasing the image, the content service can embed a “Print This!” hyperlink in the purchase page for the image:

```
<A href="PRINT:/<reference url=&quot;http://www.print.bluetooth.org/photo10.jpg&quot;
cookie=&quot;xyzcv1234&quot;/>" >Print This!</A>
```

The content service can place required information into the reference (e.g., cookies, credentials, etc.) that associates this reference to the purchase of the image. The content service can then implement print policies via the PRINT hyperlink; for example, the first time this reference is used to obtain a print-ready form of the image, the content service complies. The content service may not allow more than one print request (requiring the user to obtain a new reference to print a second time) or may automatically bill the user on a per-print basis.

The Print hyperlink defines a new protocol scheme, “PRINT”. This scheme requires a protocol plug-in in the browser in order to process the reference information contained in the hyperlink. The protocol plug-in must implement a platform-specific policy on how to deal with print references. A local protocol handler for a print reference hyperlink can use Print-By-Reference by simply converting the Print hyperlink into an OBEX reference hyperlink (defined in Section 9.4). Since there is a direct semantic mapping between a Print hyperlink and an OBEX hyperlink, this conversion is simply a matter of changing the protocol scheme from PRINT to OBEX and adding the PBR_UUID.

For example, the Print hyperlink:

```
PRINT:/<reference url="http://www.print.bluetooth.org/photo10.jpg"
cookie="xyzcv1234"/>
```

becomes:

```
OBEX:PBR_UUID/<reference url="http://www.print.bluetooth.org/photo10.jpg"
cookie="xyzcv1234"/>
```

Once the reference link has been converted it is transferred to the Printer using methods described in Section 9.4.

8.4.3 Reference List

A reference list is syntax defined to represent a list of XML references that can be sent as one transaction. The general syntax of this list is as follows:

```
<referencelist>
        <reference url=".... />
        <reference url=".... />
        ...
        ...
        <reference url=".... />
</referencelist>
```

Each reference in the list represents a separate print document and may be of different format.

For example, a user may print three completely separate documents using the following reference list:

```
<referencelist>
            <reference url="ftp://ftp.lexmark.com/pub/standards/xhtml-print.txt"
            type="application/text"/>
            <reference url="http://www.print.bluetooth.org/photo10.jpg"
            cookie="xyzcv1234" type="image/jpeg" new_sheet="false"/>
            <reference url="https://www.bluetooth.com/developer/specification/
            Bluetooth_11_specification.pdf" type="application/pdf"/>
</referencelist>
```

8.5 OBEX Header Usage Mapping

To support the device interactions described in the previous sections, a mapping to OBEX header usage is required.

8.5.1 Type Headers

The following OBEX type header values are defined for use by Print-By-Reference. The reference formats for each of the types listed are described in Section 8.4.

Type Header Value	Description
text/x-ref-simple	Simple Reference
text/x-ref-xml	XML Reference
text/x-ref-list	Reference List

Table 22: Reference Types

8.5.2 HTTP Headers

The Sender may also place HTTP headers (e.g., Authenticate) in the OBEX request. An OBEX HTTP header is a byte sequence representing a single HTTP 1.X header including the CRLF termination. If more than one HTTP header is used, each HTTP header shall be placed in a separate OBEX HTTP header. If the receiver finds any HTTP headers in the OBEX request it should include these headers in any follow-up HTTP request performed while processing the reference. Details for the use of HTTP Authenticate are provided in Section 8.7.

The HTTP Accept header specifies the RUI formats supported by the Sender.

The HTTP Accept Language header specifies the preferred locale of the Sender.

8.5.3 OBEX Response Codes

OBEX defines a set of response codes that are a direct mapping to HTTP response codes. The table in Section 16.3 lists all OBEX response codes that an OBEX server can use to reply to an OBEX request and the HTTP mapping of those codes. Included in this table is a column that indicates specific PBR usage of a subset of these response codes.

8.6 Send Reference Operation

The OBEX CONNECT request shall be used to establish session parameters between the Sender and Printer before the OBEX PUT request takes place. A CONNECT request shall include a Target header that is set to the PBR_UUID. The CONNECT response includes a Who header and a Connection Id header that provides confirmation of the service connection. See below.

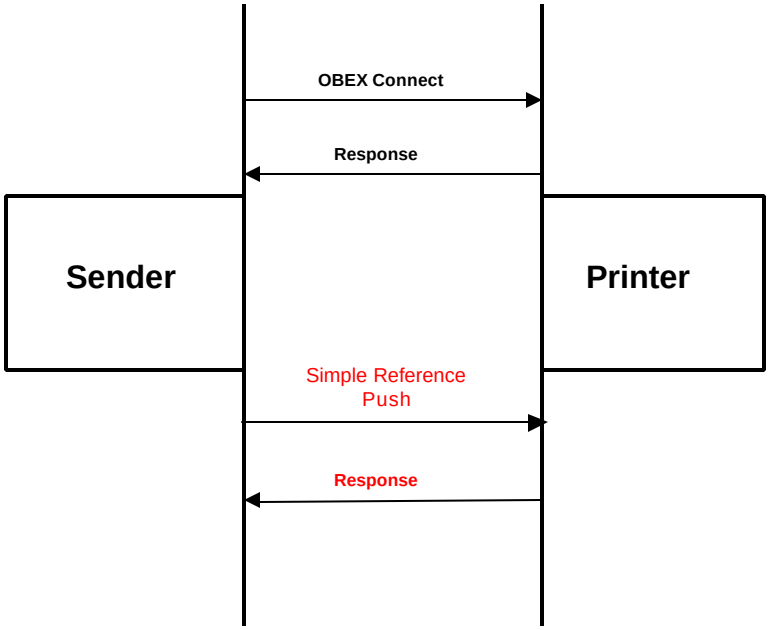

Figure 9: Reference Transfe r

After connecting, the Sender passes the reference to the Printer using an OBEX PUT request. The reference information is placed in the Body of this operation. The Printer then processes the reference and returns status back to the Sender. This minimal interaction assumes that the Printer processes the reference using a default policy defined by the Printer. The Printer may use a Print Service to retrieve the information referenced and format this information for printing. The Print Service may use a number of other services to achieve the end result.

Sender Request:	**Bytes**	**Meaning**
Opcode	0x82 0x00nn 0xCB 0xnnnnnnnn 0x42 0x0015 text/x-ref-simple 0x49 0x00CB 0x...	**PUT**, Single packet request, so final bit set. Packet Length HI for **Connection Id** header. HI for **Type** header. **Type** header length. Reference type<null-terminated US-ASCII> HI for **End-of-Body** header. 200 byte segment length Reference bytes
Printer Response:		
Opcode	0xA0 0x0003	**SUCCESS**, final bit set. Length of response packet.

Table 23: Basic Reference Exchange

Table 23 shows the use of the OBEX PUT action. The OBEX Type header shall be used to specify the type of reference being sent. The OBEX Target header shall be used in the OBEX CONNECT to specify that the PBR service is the destination of the subsequent PUT requests.

If an error occurs, an appropriate OBEX response code (see Section 16.3) shall be returned and the URL of the reference shall be returned in the OBEX Body header of the response. Additional attributes of the reference shall not be returned. If the entire reference does not fit in the Body header of the response, the reference shall be truncated.

8.7 Reference Exchange Security

Four security services are defined for the exchange and processing of a reference. For some services, specific support is required from the Content Provider. Support for security is mandatory for Printers but optional for Senders. The implementation of a specific security policy is optional. The following security mechanisms are defined:

1. Printer access control
2. Access control to referenced content
3. Privacy of reference content
4. Privacy of reference

8.7.1 Printer Access Control

Printer authentication and access control shall be supported via the standard OBEX challenge response protocol. Any OBEX request (CONNECT, PUT, GET, etc.) can be challenged by the OBEX server, requiring the OBEX client to present authentication information. It is this interaction that is used to support authentication between the Sender and Printer. (Note: This authenticates the reference exchange but not necessarily access to the network resource.) If the initial CONNECT request is challenged by the Printer, the Sender may offer the correct credentials and replay the CONNECT request. This interaction is shown in Figure 10.

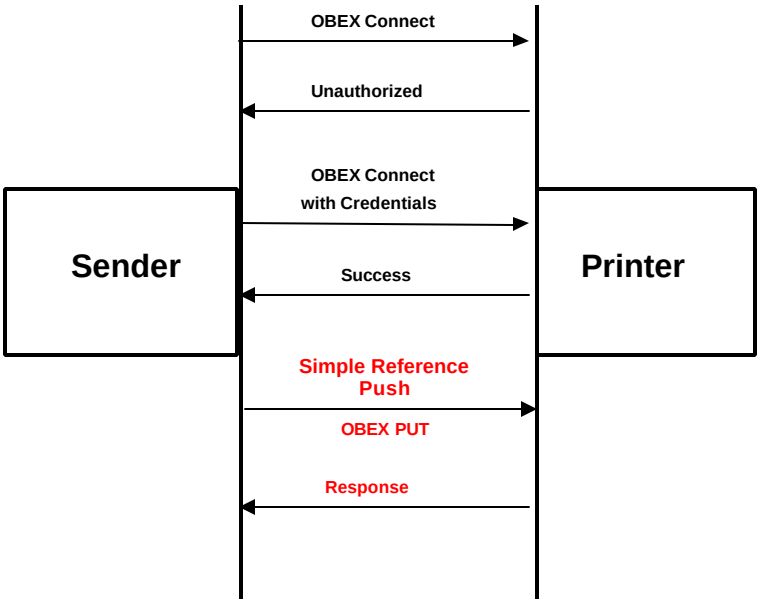

Figure 10: Printer Access Control

This interaction allows the Printer to protect itself from unauthorized access. This may be used for generally restricting the use of a Printer or restricting specific printer operations (job control, printer configuration, etc.).

8.7.2 Access Control to Referenced Content

Print-By-Reference supports two mechanisms for controlling access to referenced content. Both mechanisms leverage the basic challenge/response capability defined for web-based access control over HTTP (RFC 2617, [33]).

In the first case, the security credentials may be included in the reference itself. The XML reference specification includes the on401 and on407 attributes that may be used to associate credentials with a reference. When these fields are specified, the Printer uses the supplied credentials in response to an HTTP 401 (authenticate) or 407 (proxy authenticate) challenge to an HTTP content access. If the credentials are not specified in the reference, the Sender shall be notified, and may provide the required credentials in a new request.

In the second case, PBR leverages support that already exists in both OBEX and HTTP. When a Printer processes a reference, the content provider may choose to challenge the request using the HTTP 401 or (if acting as a proxy) 407 (unauthorized) status response. After receiving this challenge, the Printer passes this challenge back to the Sender via the OBEX 0x41(0xC1) response code. The Sender may then look for the HTTP authentication information in the OBEX headers of the response. This shall consist of one or more WWW-Authenticate headers in the OBEX 0x41(0xC1) response packet. If there is more than one WWW-Authenticate header they shall be in order of preference. In order to gain access to the content, the Sender may

respond to the challenge by replaying the original request, either with an OBEX HTTP Authorization header, or by adding an on401 credential to the XML-specified reference (see Section 8.4.2). The syntax of the WWW-Authenticate and HTTP Authorization headers is described in RFC 2617, [33]. These headers shall be sent as OBEX HTTP headers in the OBEX request.

The sequence of events is shown in Figure 11 and Figure 12.

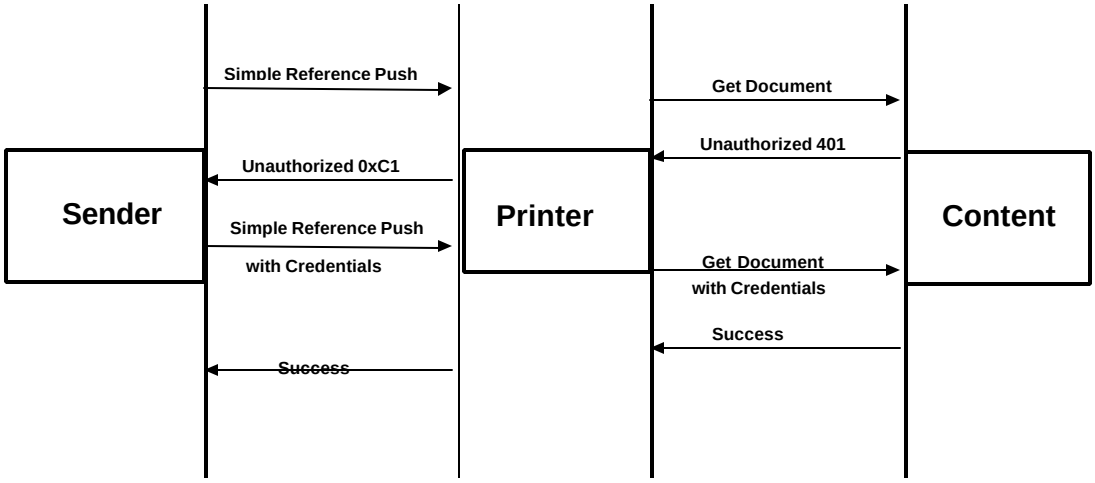

Figure 11: Security Challenge

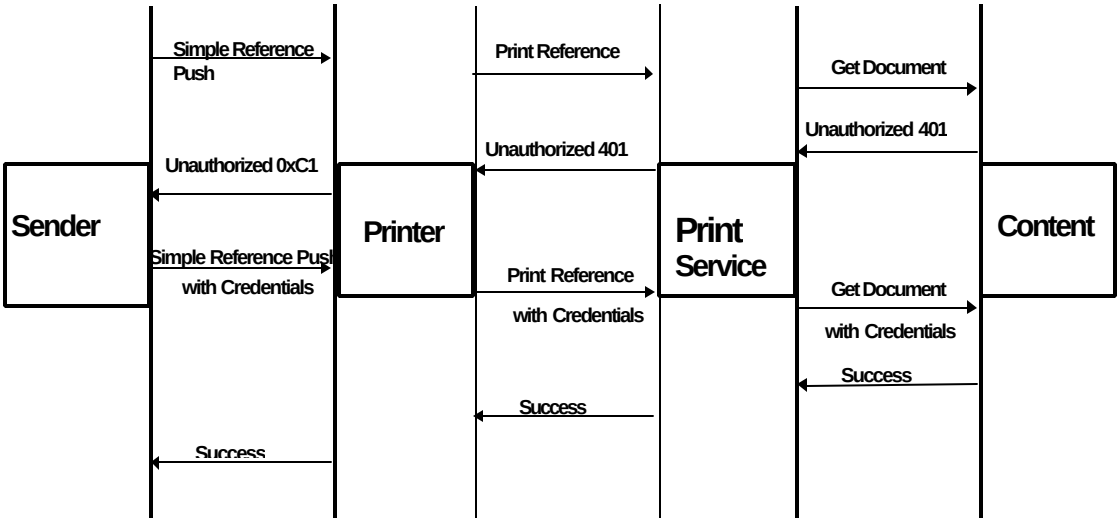

Figure 12: Security Challenge with Print Service

8.7.2.1 Reference Lists

Since a reference list represents multiple documents, the credentials required to access each reference can be different. Each reference can individually include the

credentials using the appropriate attributes in the XML reference specification as described in Section 8.4.2. If a reference does not include the credentials, then any security challenge encountered during the processing of that reference shall be sent back to the Sender for a response. This means that in some cases the Sender may respond to multiple authentication challenges to authenticate access in a particular security domain. The Printer may optionally cache security authorization headers such that if more than one reference in a list belongs to the same security domain, it need not repeatedly contact the Sender. (Note: This is similar to the way browsers currently operate.) In this case it would automatically add the cached information in response to the challenge. (Implementors Note: This cache should expire at the end of a print session and not be retained across multiple print sessions from the same Sender.)

If an error occurs on a reference in the list, the Printer shall save it and continue processing all remaining references and remember those that could not be printed. Once all of the links in a reference list that can be printed have been printed, the Printer shall return the response. If a single error occurs, the appropriate OBEX response code is returned and the URL of the reference is returned in the Body header of the response. Any additional attributes of the reference shall not be returned.

If more than one error occurs, the Printer shall return the OBEX response code (0x26, Partial content) indicating that multiple errors occurred and shall return the URL of the first reference that failed in the Body header of the OBEX response. Any additional attributes of the reference shall not be returned. If the entire URL does not fit in the Body header of the response, the URL is truncated.

8.7.3 Privacy of Referenced Content

Content privacy deals with protecting the content on the communications link between the Printer and the Content Provider. To support content privacy this link can be encrypted to ensure that the information is not sent in a clear form. This link can be safely encrypted by the use of HTTPS or TLS between the Content Provider and Printer. The use of HTTPS or TLS to protect the privacy of a resource can only be established if the Content Provider supports HTTPS or TLS access and the reference specifies HTTPS or TLS as the access protocol (rather than HTTP). Since support for HTTPS or TLS by the Printer is optional, a Printer shall refuse to process any references to HTTPS or TLS content if not supported and return an appropriate OBEX response code (e.g., 0x51(0xD1) – not implemented).

8.7.4 Privacy of the Reference

HTTPS or TLS protects the privacy of the content being accessed on the network link between the Printer and Content Provider. This scheme can not, however, protect the transmission of the reference itself as it is sent across the Bluetooth link. Privacy on the Bluetooth link is provided by Bluetooth security mechanisms for communication. These mechanisms are detailed in the Bluetooth Core Specification [1].

8.8 Job-Based Send Reference Operation

This method uses the CreateJob operation defined in Section 7.1.2. The result of a CreateJob operation includes a JobId that shall be used by subsequent operations. The CreateJob is followed by a SendReference operation that includes the reference information to be printed. The SendReference operation uses exactly the same OBEX headers as the SendDocument operation (defined in Section 7.1.3) except that the Body header of a SendReference operation includes the reference information instead of data to be printed. As with the SendDocument, one and only one SendReference operation is allowed per CreateJob operation.

8.9 Reference Hyperlinks Mapping to OBEX

Section 8.4.2.1 defines a hyperlink specification to enable the support of embedded print references. This allows a Content Provider to embed PBR functionality directly into a web page. This section describes how this support can be mapped to the OBEX transport protocol.

The OBEX mapping is described using the following example:

```
<A href="PRINT:/<reference url=&quot;http://www.print.bluetooth.org/photo10.jpg&quot;
cookie=&quot;xyzcv1234&quot;/>" >Print This!</A>
```

The hyperlink defined above sends a reference to the PBR service on the Printer.

The mapping to OBEX is defined as follows:

Sender		**Printer**
User Selects Hyperlink PRINT:/<reference url="http://www.print.bluetooth.org./photo10.jpg" cookie="xyzcv1234" />		
	OBEX CONNECT **Target** header contains PBR_UUID	
GetRUI	OBEX **GET** → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type** header contains: text/x-ref-xml **HTTP** (Accept) header contains a list of RUI formats supported by the Sender. **HTTP** (Accept-Language) header contains the locale of the Sender. **Name** header contains the Reference Printing Top URL from the Printer's Service Record (UTF-16). **Body** header contains <reference url="http://www.bluetooth.org.com/photo10.jpg"	**Process Request**

	cookie="xyzcv1234" />	
	← Response **Body** header contains the top-level control document for the PBR service on the Printer supporting PBR job control or status.	**Done**
Process Response		

Table 24: RUI Reference OBEX Mapping

The interaction shown in Table 24 uses the RUI service defined in Section 9. The use of RUI in the above interaction is required since the browser controls the transaction, therefore some web-based response to the hyperlink selection is required to provide status to the user and allow the Sender to control the print operation.

It is also possible to bookmark the reference and send it via the PBR Simple Reference Push or CreateJob/Send Reference operations outside of the browser.

9 Reflected User Interface (Optional)

This section describes a user control model for Bluetooth printing that may be utilized by both the Direct Printing and Print-By-Reference components of the BPP. The use of Reflected User Interface (RUI) is optional, and can only be supported by suitably configured Senders and Printers. This method of control is complementary to the Job-Based Transfer model described in Section 7.1.2. Interoperability in RUI is based on the ability of the Sender to render a document (user interface) encoded in a mark-up language such as HTML, CHTML, or WML. The RUI document is generated by the Printer or Print Service when the Sender sends a request. Reflected UI does not require the Sender to have prior knowledge of the control elements and data structures involved in job control, printer capabilities, and status reporting since these are embedded into the mark-up document generated by the Printer.

The Reflected User Interface usage model assumes a browser component on the Sender. This browser, once invoked, is used to control all aspects of a print job or other RUI operation. The intention is to leverage existing browser platforms rather than require a specific browser implementation for printing.

The Bluetooth Printing Profile defines two main uses for RUI:

1. Printer Administration Interface – The default or top-level user interface for the device; used to provide user access to printing, administration, configuration, and monitoring services. If the Printer is a multifunction device then the interface may also include user access to the other non-printing functions.

2. Transactional control for Direct Printing and/or Reference Printing – This provides user access to print job control (e.g., setting of printer capabilities, security, billing, etc.) during a Direct Printing and/or PBR print request.

The Printer Administration Interface may also be accessed via other protocols such as: HTTP and WAP.

A Printer shall indicate support for RUI either by including a Printer Administrative User Interface Service Record (Table 41) if an administrative user interface is provided, or by including the Direct Printing RUI Boolean or the Reference Printing RUI Boolean in the Basic Printing Service Record (Table 39). The Service Record shall also include the RUI formats supported by the Printer and the top-level URL for the RUI option supported. This information informs the Sender that an RUI interaction with the Printer is possible. The Sender initiates an RUI interaction by using the OBEX GET request. All RUI interactions use the basic OBEX GET. Table 25 shows the basic RUI interaction.

Sender		Printer
	OBEX CONNECT **Target** header contains UUID of the service required (DPS_UUID, PBR_UUID, RUI_UUID). See Section 9.1	
GetRUI	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type** header contains x-obex/RUI if the Body header is empty, otherwise the type of data contained in the Body header of the request. **HTTP** (Accept) header contains list of RUI formats supported by the Sender. **HTTP** (Accept-Language) header contains the locale of the Sender. **Name** header contains the URL naming the user interface page being accessed, or may be empty if the Body header contains form data. Request Data is contained in **Body** header.	**Process Request**
	← Continue GET → (only if request is larger the negotiated packet size)	
	← Response **Body** header of the response contains an RUI document encoded in a form supported by the Sender. This document is used to display a user interface via a browser that can render the encoding	**Done**
Process Response		

Table 25: RUI Request/Response

9.1 RUI Naming

An RUI request may be directed to a specific service or function within a printing system. The naming mechanism used by RUI is a combination of the targeted service (i.e., the Target header in the OBEX CONNECT) and Name header included in the GET request. The Target header in the OBEX CONNECT is used to indicate the desired RUI service. Table 26 shows the valid service UUIDs defined for RUI in this Profile.

UUID	Service	Purpose
[RUI_UUID]	Reflected User Interface (Administrative RUI)	This is the administrative reflected user interface provided by the printer. It would typically provide user access to the main functions and services supported by the printer. This interface may not be printer-specific in devices that support operations other than printing.
[DPS_UUID]	Direct Printing Service (DPS)	This reflected user interface supports access to the Direct Printing service that supports the printing of information stored on the Sender.
[PBR_UUID]	Reference Printing Service (PBR)	This reflected user interface supports access to the Print-By-Reference service that can be used to print information stored in the network.

Table 26: RUI Service Mapping

It is possible that each service may provide other named RUIs that support a particular sub-function of the service. For example, the default RUI service may include an interface for the administration of a Printer as well as an interface that shows the current ink status.

Access to these interfaces shall be specified using the OBEX Name header in the GET request.

In general, access to lower-level, named interfaces can be supported via hyperlinks in higher-level interface documents. Once the default interface has been retrieved, the user can access the lower-level interfaces via these hyperlinks. The number, definition, and scope of the named interfaces provided by a printing system are vendor-dependent and as such not defined in this Profile.

9.2 RUI Document Formats

A Reflected User Interface is a document that is encoded using a UI mark-up language. UI mark-up languages form the basic models for user interaction with information and services on the world-wide-web. To be interoperable, the Sender and Printer need to agree on the RUI mark-up language format. The Printer specifies the supported RUI document formats in the Service Record. A Sender shall use this information to determine whether an RUI interaction is possible with the Printer.

When a Sender sends an RUI request to the Printer it shall specify its RUI format preferences. RUI leverages the same technique used by existing web browsers by setting the HTTP Accept header. The HTTP Accept header allows the Sender to specify the list of RUI document formats it can support in preference order. This information is sent in the OBEX GET request as an HTTP header. Section 14.1 of RFC 2616 [30], provides the necessary details on the use of this header.

The UI mark-up language is required to support the basic constructs that enable device and job control information to be passed between the Printer and the Sender.

The most basic construct is the notion of hyperlinks. Hyperlinks allow navigation to other control pages within a Reflected UI. Hyperlinks can also be associated with commands that do not require data input (the command string is represented in the hyperlink). (For example, a command hyperlink to cause the printer to go Offline.) Commands form a basic element of control to invoke actions on the Printer.

Another basic construct is the notion of a form. A form is a collection of data fields and state variables that represent an operation, transaction, or function of a service. The basic operation of a form allows the user to enter information into the data fields provided and then submit the form information (data fields and state information) for processing. This is analogous to the basic dialog control model used in many window-based user interfaces. (For Example, a form that allows the user to enter the number of copies to be printed.)

Currently there are three web UI mark-up languages that support the necessary constructs for RUI based control.

HTML is the most common and widely-used UI mark-up language. HTML has supported the notion of hyperlinks and forms since Version 1.0.

Compact HTML (CHTML) is a variant of HTML designed for small information appliances. CHTML uses the same hyperlink and form constructs as HTML with the same syntax, attributes and tags.

The Wireless Mark-up Language (WML) is the other relevant mark-up language. This language was developed for the WAP architecture and targets devices with very simple UI capabilities. WML is mostly used to support access to the web on cell phones. WML supports both the notions of hyperlinks and forms; the syntax and operation is different from HTML but the functionality is similar.

9.3 Browser Considerations

Reflected UI requires the presence of a browser on the Sender to parse and render the RUI control document provided by the Printer (or Print Service). In order to support RUI interactions over OBEX a new URI format is specified. The new URI

format indicates to the browser that hyperlinks and form submissions shall use the OBEX transport protocol instead of an IP -based protocol traditionally used with HTTP-based URIs.

This is shown in Figure 13. In this case the control doc ument contains two hyperlinks. The first hyperlink (http://printersrus.com/printers1.html) is a traditional HTTP link that is accessed over IP. The second hyperlink (OBEX:RUI_UUID/PrinterStatus) defines an OBEX link that indicates to the browser that this information shall be sent over the OBEX connection on Bluetooth.

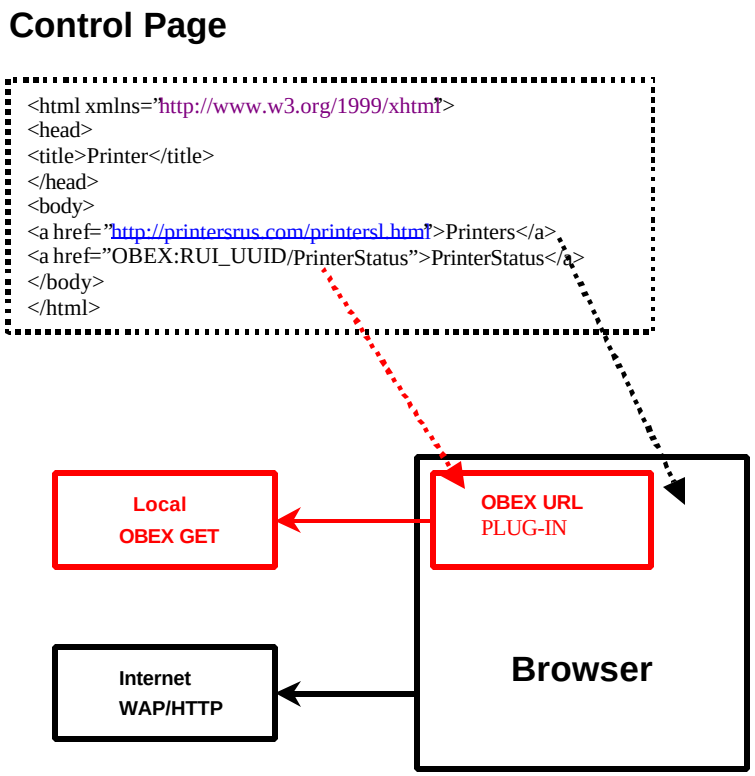

Figure 13: OBEX URL Processing

9.4 OBEX URI Syntax

A device that supports RUI shall also support the OBEX URI.

The generic syntax for a URI is defined in RFC 2396 [34]. The URI syntax depends on the scheme (protocol) used. The general form of an absolute URI is:

<Scheme>:<scheme-specific-part>

An absolute URI contains the name of the scheme being used (<scheme>) followed by a colon (":") then a string (scheme-specific-part) whose interpretation depends on the scheme.

The OBEX URI has the following general form:

OBEX:<service-uuid>/[<path>][?<query>]

The scheme is defined as OBEX followed by a mandatory identifier that indicates which OBEX service should handle the URI <service-uuid>. The service identifier is followed by a forward slash ("/") and a path and an optional query string.

The path portion [<path>] of the URI can be hierarchical in nature with each part of the hierarchy delimited by a forward slash ("/"). The path is used to indicate the name of a resource provided by the OBEX service.

The query portion [?<query>] of the URI shall be preceded by a question mark ("?") and consists of data to be associated with the URI. The characters that follow the question mark shall conform to the URI characters and escape sequences defined in RFC 2396 [34]. The path and query syntax is the same as traditional HTTP-based URLs.

9.5 OBEX URI Protocol Mapping

The OBEX URI is mapped onto the GET request as defined for Reflected UI in Table 25. The protocol handler for the OBEX URI (shown in Figure 13) parses the URI and creates a GET request from the components. The following examples show this mapping in more detail.

9.5.1 Top-Level Control Page Hyperlink

A top-level control page hyperlink refers to the initial or entry-point interface provided by a Reflected User Interface service. It is analogous to the homepage concept in traditional web browsing. The URL for a service's top-level interface shall be included in the Service Record if that service supports RUI. The example below shows the Service URI for an access to the top-level interface.

The Administrative RUI, PBR, and Direct Printing services may be selected by using the corresponding UUID.

The example mapping to OBEX for a top-level access to the Printers Administrative RUI page follows:

Sender		**Printer**
	OBEX CONNECT **Target** header contains UUID of ReflectedUI service.	
User Selects Hyperlink OBEX:0x00001121/top-level-name		
GetRUI	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type** header contains x-obex/RUI. **HTTP** (Accept) header contains a list of RUI formats supported by the Sender. **HTTP** (Accept-Language) header contains the locale of the Sender. **Name** header contains the Top URL from the Printer Administrative User Interface Service Record (UTF-16).	**Process Request**
	← Response **Body** header contains Top-Level control document for the Administrative RUI service on the Printer.	**Done**
Process Response		

Table 27: Administrative Service Hyperlink OBEX Mapping

9.5.2 Named Control Page Hyperlink

A named control page hyperlink includes the addition of a path that represents the name of a specific RUI resource within a service.

OBEX:[Insert UUID here]/top_level_page/my_named_page

The URI shown above includes the path “top_level_page/my_named_page” that is the name for a specific interface within the RUI (Administrative, Print-By-Reference, or Direct Printing) service on the Printer.

The mapping to OBEX is as follows:

Sender		**Printer**

	OBEX CONNECT **Target** header contains UUID of service, (DPS,PBR,RUI).	
User selects Hyperlink OBEX:[Insert UUID here] /top_level_page/my_named_page GetRUI	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type** header contains x-obex/RUI if the Body header is empty, otherwise the type of the data contained in the Body header of the request. **HTTP** (Accept) header contains list of RUI formats supported by the Sender. **HTTP** (Accept-Language) header contains the locale of the Sender. **Name** header contains the complete URL. (UTF-16).	**Process Request**
	← Response **Body** header contains the Remote User Interface document for the targeted service, which is "my_named_page".	**Done**
Process Response		

Table 28: Named Control Page Hyperlink OBEX Mapping

9.5.3 Form Submission

When the OBEX URI is used as part of a form submission, the information in the form is collected together and sent with the GET request. The examples below show both HTML/cHTML and WML form submissions.

9.5.3.1 HTML

```
<form method="POST" enctype="application/x-www-form-urlencoded"
 action="OBEX:[insert RUI_UUID here]/ExampleForm" name="example">
   <p> </p>
      <p> Some Text : <input type="text" name="T1" size="20"></p>
   <p><input type="submit" value="Submit" name="B1"><input type="reset"
   value="Reset" name="B2"></p>
   <input type="hidden" name="StateVariable1" value="123"><input
   type="hidden" name="StateVariable2" value="example">
</form>
```

When the submit button on this form is selected, the elements are gathered up and result in a URI of the form:

OBEX:[insert UUID here]/ExampleForm?T1=[text]&StateVariable1=123&StateVariable2=example

where [text] is the actual text entered by the user.

The mapping to OBEX is as follows:

Sender		**Printer**
	OBEX CONNECT **Target** header contains UUID for RUI service.	
User submits the form OBEX:[Insert UUID here]/ExampleForm?T1=[text]&StateVariable1=123&StateVariable2=example		
GetRUI	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type** header contains the value of the encoding attribute specified in the Form or application/x-www-form-urlencoded. **HTTP** (Accept) header contains list of RUI formats supported by the Sender. **HTTP** (Accept-Language) header contains the locale of the Sender. **Name** header is empty. The **Body** header contains the <path> and <query> portions of the URL. "ExampleForm?" T1=[text]&StateVariable1= 123&StateVariable2=example	**Process Request**
	← Response **Body** header contains the response document from the form submission (e.g., status of request or another control page).	**Done**
Process Response		

Table 29: Form Submission OBEX Mapping

9.5.3.2 WML

```
<card id="example" title="Example">
 <setvar name="StateVariable1" value="123">
 <setvar name="StateVariable2" value="example">
 <P>Some Text:
    <input name="T1" value="" />
 </P>
 <go href=OBEX:[insert RUI_UUID here]/ExampleForm method="POST">
    <postfield name="T1" value="$T1"/>
```

```
    <postfield name="StateVariable1" value="$StateVariable1"/>
    <postfield name="StateVariable2" value="$StateVariable2"/>
 </go>
</card>
```

The WML card shown above yields the same result as the HTML Form described previously. The mapping to OBEX is also the same (Table 29).

9.5.3.3 Form Encoding Types

The encoding of the form data is specified by the UI mark-up language used. Common form encoding types for HTML include “multipart/form-data” and “application/x-www-form-urlencoded”. Form encoding types for other UI mark-up languages are defined by the language.

9.5.4 OBEX Response Codes

OBEX defines a set of response codes that are a direct mapping to HTTP response codes. The table in Section 16.3 lists all OBEX response codes that an OBEX server can use to reply to an OBEX request and the HTTP mapping of those codes. Included in this table is a column that indicates specific RUI usage of a subset of these response codes.

9.6 Locale

Reflected UI supports language localization through the specification of the Sender locale in the RUI request. This information provides a hint to the Printer or Print Service that the Sender prefers to receive localized control pages. This information is only a hint; support for localization at the Printer or Print service is optional. If the Print Service does not support transfer of control information in the specified locale, it should return the control information in its default form.

The Sender shall specify its preferred locale through the use of the HTTP Accept-Language header in the OBEX GET request. The use of the HTTP Accept-Language header is specified in Section 14.4 of RFC 2616 [30].

9.7 Administrative Reflected UI Service

This Reflected UI service allows a printer to provide an administrative or general user interface that supports user access to the function(s) of the device. In this Profile it is referred to as an Administrative RUI or just RUI (rather than DPS RUI or PBR RUI) and its support shall be indicated by including a separate Service Record. This interface may not be specific to printing and may offer hyperlinks to many operational aspects of a device. The Administrative RUI for a printer may also provide access to the DPS RUI and PBR RUI (if supported) via hyperlinks in its top-level control page. The Administrative RUI may be accessed via OBEX using the GET operation as described above, using the RUI_UUID in the OBEX Target header of the CONNECT request. If the Printer has web presence served from a traditional web server,

embedded in the Printer, the Sender may use any access mechanism available including HTTP or WAP to obtain this control page.

9.8 PBR Reflected UI Service

The PBR service implements support for the printing of information hosted in the network. The Print-By-Reference service can use a Reflected UI to provide transactional control as part of a PBR print job. This basic interaction is shown in Figure 14.

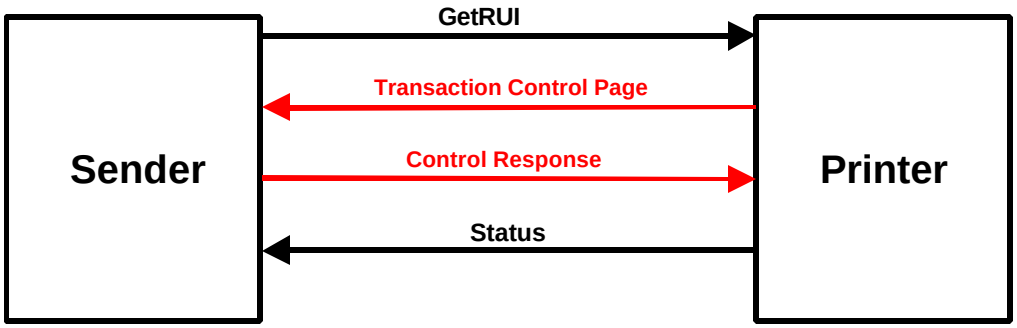

Figure 14: Simple Transaction Control

When the opportunity to perform transactional control exists, the Sender sends a GetRUI request. The use of GetRUI signifies to the printer that the Sender wishes to use reflected UI for transactional control. The Sender shall set the HTTP Accept and Accept-Language headers of the OBEX GET request indicating the RUI format required and locale preferred for control pages that are returned. All other OBEX headers are the same as those used in PBR.

After receiving the GetRUI request, the Printer shall provide an appropriately encoded control page in the Body header of the OBEX GET response. The Printer defines the content and structure of this page. Figure 15, shows an example of a transactional control page.

Figure 15: Example Control Page

In this case, the RUI control page allows the setting of Printer options for the particular Printer being accessed. A Printer with different features can generate a different user interface to expose those features (e.g., stapling, collation etc.).

The control page in Figure 15 is coded as an HTML form. The form contains graphical elements that support data entry by the user to set or modify the various options. The form also contains hidden state variables (e.g., JobId) used by the Printer to associate the RUI with a particular transaction. The type and format of the state information is defined by the Printer and may be different across printer implementations.

After the user has filled out the information in the Reflected UI shown above, the form is submitted. This action gathers up the form data including the state information and submits the form via OBEX using the procedure defined in Section 9.5.3.

The Printer continues to process the reference sent in the original request, perhaps optionally utilizing a remote Print Service over interface B shown in Figure 7.

This interaction may yield another control page generated by the remote Print Service itself. While the details of interface B in Figure 7 are outside the scope of this Profile,

this interface shall understand the RUI capabilities of the Sender and the transaction state in order to provide a control page. The control page provided by the Print Service is returned to the Sender (in response to the previous form submission) and the same user actions take place to respond to this control page. Once the print job has completed (or failed) a status page is sent to the Sender. This final page completes the print operation. This interaction is shown in Table 30.

Sender	**Interface A** **OBEX**	**Printer**	**Interface B**	**Print Service**
	OBEX CONNECT **Target** header = PBR_UUID.			
GetRUI Sender sends a reference to the printer indicating support for RUI.	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type** header = Reference encoding type. **HTTP** (Accept) header = RUI formats acceptable to Sender. **HTTP** (Accept-Language) header = Language preferred by Sender. **Name** header = Reference Printing Top URL (UTF-16). **Body** header = Remote Reference.	Printer receives reference, allocates state information for the print job, and associates this with a Job Identifier (format defined by Printer).		
	←Response Control Page	Printer generates RUI form to allow user to set Printer options. Form may include graphical elements for data entry plus print job state information held in form variables.		
Sender launches browser to display the RUI page				
User enters data into form to select options and submits the form.	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type** header = Encoding type of the form data. **HTTP** (Accept) header = RUI formats acceptable to Sender. **HTTP** (Accept-Language) header = Language preferred by Sender. **Name** header = Defined by Printer.	Printer receives form information, retrieves Job Identifier to access state information. Printer retrieves options set by user and proceeds to process the print job.		

Sender	Interface A OBEX	Printer	Interface B	Print Service
	Body header = Form data.			
		Printer uses a remote Print Service to process the reference. Sends job information to Print Service.	Send →	Print Service accepts print job and starts to process the reference.
	← Response Control Page	Printer sends RUI onto user.	← Response Control Page	Print Service Requires confirmation from user. Generates RUI that is sent back to the Printer.
User confirms Print Service information and submits form.	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type** header = Encoding type of the form data. **Name** header = Defined by Print Service. **Body** header = Form data.	Printer forwards response to Print Service.	Send →	Print service completes print job and sends printout to Printer.
			← Response	
		Printer prints out print job.		
	← Response Status Page	Printer generates status page and returns this to user.		
User collects printout and closes browser.				

Table 30: PBR Transaction Control with Print Service

Note: In the case that the optional external Print Service is not used, all interactions occur between the Sender and the Printer.

9.9 Direct Printing Reflected UI Service

Direct Printing Service of BPP supports the printing of information stored on the Sender. Direct Printing can use Reflected UI to provide transaction control information much as PBR does. This allows the user to set print job options and be provided with graphical status and error information through the use of RUI pages.

The mechanism used to support Direct Printing-based transaction control leverages heavily the procedure defined for Reference Printing. The Sender starts the transaction by sending a Direct Printing GetRUI request using the OBEX GET

request. The Body header of this initial request contains an XML reference for the information on the Sender. This is essentially a reference, local to the Sender, with the syntax defined in section 8.4.2. Standard RUI OBEX header information shall also be specified along with a Target header (in the OBEX CONNECT) set to the DPS_UUID. After receiving this request, the Printer returns the Reflected UI for setting print job options in the same manner described in PBR. When all of the RUI print job interactions have finished, the printer shall use the GetReferencedObjects operation defined in Section 7.1.6 over the RUI Referenced Jobs Channel, to retrieve the document stored on the Sender. This interaction is shown in Table 31.

Sender	**Interface A** **OBEX**	**Printer**
	OBEX CONNECT **Target** header = DPS_UUID	
GetRUI User initiates a print request for a local file. A reference is created to this file and sent in a GetRUI request to the Direct Printing Service.	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type header** = Reference encoding type. **HTTP** (Accept) header = RUI formats acceptable to Sender. **HTTP** (Accept-Language) header = Language preferred by Sender. **Name** header = Direct Printing Top URL (UTF-16). **Body** header = Local Reference	Printer receives reference, allocates state information for the print job, and associates this with a Job Identifier (format defined by Printer).
		Printer generates RUI form to allow uses to set Printer options. Form includes graphical elements for data entry plus print job state information held in form variables.
	←Response Control Page	
Sender launches browser to display the RUI page		
User enters data into form to select options and submits the form	OBEX GET → **Connection Id** header contains the identifier assigned as a result of the CONNECT. **Type header** = Encoding type of the form data. **Name** header = Defined by Printer. **HTTP** (Accept) header = RUI formats acceptable to Sender.	Printer receives form information retrieves Job Identifier to access state information. Retrieves the options set by user and starts to process the print job.

Sender	**Interface A** **OBEX**	**Printer**
	HTTP (Accept-Language) header = Language preferred by Sender. **Body** header = Form data.	
Sender serves up the local document	←GetReferencedObjects Send Block → Block based transfer of document controlled by printer.	Printer accesses reference.
		Printer prints out the document.
	←Response Control Page	Printer sends status page back to user.
User collects print-out		

Table 31: Direct Printing Service Reflected UI Interaction

10 Common Object Formats for BPP

This section describes some of the common object formats that may be sent from Senders to Printers. These data formats are valid for any of the print scenarios described in the Basic Print Profile.

In some cases, details and guidelines regarding how these objects should be printed are included.

10.1 Format Overview

Table 32 indicates a few common formats that may be used in the Bluetooth Basic Printing Profile.

Format	Support in Sender	Support in Printer
XHTML-Print	M	M
Basic Text	O	O
vCard	O	O
vCalendar	O	O
vMessage	O	O

Table 32: Object Formats

10.2 XHTML-Print

Bluetooth Basic Printing Profile-compliant Printers shall meet the requirements of XHTML-Print [22]. For a Sender, at least one application shall support printing using XHTML-Print. It is recommended that all applications on a Sender that require printing support at least the XHTML-Print data format.

10.3 Basic Text

This object format is optional for Senders and Printers.

The basic text format is a UNICODE character stream with no mark-up that is UTF-8-encoded. US-ASCII compatibility for character codes 0x00 to 0x7F is achieved by using UTF-8.

To make sure that the text is readable on the Printer, the Sender should check the "Character Repertoires Supported" entry in the Basic Printing Service Record before sending the text. If the Printer does not support the targeted character repertoire, it is up to the Sender to decide whether to send the document to the Printer. If sent, the Printer should make its “best effort” attempt to print all glyphs; i.e., it should not hang up or exhibit failure behavior, other than the failure to print unsupported glyphs.

If the character stream contains CR/LF characters, they shall remain in the print data stream to allow output of simple preformatted text. If the text does not contain any CR/LF, or if the row being rendered is too long to fit on the actual media, then the Printer s hall insert control codes for text wrapping. (Behavior when the character stream contains only CR characters with no LF or only LF characters with no CR is device-dependent.)

A fixed-pitch font shall be used to allow the Sender to print preformatted text in the form of columns using multiple white spaces as a tabulation method. A fixed-pitch font also makes it easier to calculate where text wrapping should take place.

If the Printer does not know the medium size, a medium with the maximum width of (iso_a4_210x297mm) and the maximum length of (na_letter_8.5x11in) should be assumed. This ensures that printed information is always within the limits of the printable area when using either A4 or Letter-sized medium.

10.3.1 Basic Text Line-Breaking Guidelines

It is recommended that line-breaking behavior follow the guidelines presented in the Unicode Standard [32] and the Line Breaking Properties Annex [31].

10.4 vCard

A Printer that indicates support for vCard shall accept data according the vCard specification cited in [11]. It is recommended that vCard information be transmitted using UTF-8 encoding. The MIME media-type reported in the Service Record is: (text/x-vcard:2.1).

Multiple vCard objects may be transferred using one SendDocument action.

Limitations

Nested vCards are not supported.

10.4.1 vCard Layout Parameters

When sending vCard data to the printer, layout information may be included in the Description header of the OBEX PUT operation. This information may be included as part of either the SendDocument operation (if a CreateJob is used) or the simple FilePush operation.

The format of this layout information in the Description header is defined below.

Type	Parameter Name	Support in Sender	Support in Printer
I4	CardsPerPage	O	O
	Description: The number of vCards to be included per page when a stream of vCards is passed to the printer.		
String	CardLayout	O	O
	Description: The layout of the vCard data. • ***business-card*** *– Each vCard is printed in business card layout.* • ***phone-book*** *– All vCards are printed out in a phone book style; e.g., only a list of names and phone numbers*		

Table 33: vCard Layout Parameters

Syntax

<*Parameter Name*>*Parameter Value*</*Parameter Name*>

Example

<*CardsPerPage*>*16*</*CardsPerPage*>

Note: The OBEX specification requires UTF-16 encoding for the Description header field.

The Printer should inspect the document type before trying to interpret the content of the OBEX Description header field. The document type shall be indicated in the Type header field of the OBEX PUT request; and optionally as part of the CreateJob operation, if the job-based model is used.

Note: An alternate vCard format specification is cited in [25]. Support for vCards compliant with this specification is optional. The MIME media-type for this alternate format is: (text/x-vcard:3.0).

10.5 vCalendar

A printer that indicates support for vCalendar shall accept data according to the vCalendar 1.0 content format specification in [12]. It is recommended that vCalendar information be transmitted using UTF-8 encoding. The MIME media-type reported in the Service Record is: (text/x-vCalendar:1.0).

Multiple vCalendar objects may be transferred using one SendDocument operation.

Limitations

- Appointments shall be sent in chronological order.

- Only vCalendar objects received by the printer shall be printed; that is, no recurring events shall be printed more than once, if not transmitted more than once.

10.5.1 vCalendar Layout Parameters

When sending vCalendar data to the Printer, layout information may be included in the Description header of the OBEX PUT request. This information may be included as part of either the SendDocument operation (if a CreateJob is used) or the simple FilePush operation. The format of this layout information in the OBEX Description header is defined below.

Type	Parameter Name	Support in Sender	Support in Printer
String	CalendarFormat	O	O
	Description: The layout on the page of the vCalendar data. • ***single –*** *The vCalendar data is processed and printed as single appointments or events.* • ***daily –*** *The vCalendar data is processed and printed in a daily layout.* • ***weekly*** *– The vCalendar data is processed and printed in a weekly layout.* • ***monthly*** *– The vCalendar data is processed and printed in a monthly layout.*		
I4	CalendarFormatsPerPage	O	O
	Description: The number of vCalendar objects to be printed per page (e.g., if CalendarFormat = daily, then CalendarFormatsPerPage = the number of daily calendar entries per page).		

Table 34: vCalendar Layout Parameters

Syntax

<Parameter Name>Parameter Value</Parameter Name>

Example

<CalendarFormat>weekly</CalendarFormat>

Note: The OBEX specification requires UTF-16 encoding for the Description header.

The Printer should inspect the document type before trying to interpret the content of the OBEX Description header. When CreateJob/SendDocument operations are used, the document type shall be indicated in the Type header of the OBEX PUT request

and optionally as part of the CreateJob operation. When the FilePush operation is used, the document type shall be included in the Type header of the OBEX PUT request.

Note: An alternate vCalendar format specification is cited in [26]. Support for vCalendars compliant with this specification is optional. The MIME media-type for this alternate format is: (text/calendar:2.0).

10.6 vMessage

A Printer that indicates support for vMessage shall accept data according to vMessages Version 1.1 as specified in [9]. It is recommended that vMessage information be transmitted using UTF-8 encoding. The MIME media-type reported in the Service Record is: (text/x-vmessage:1.1).

Multiple vMessage objects may be transferred using one SendDocument action.

10.7 Mandatory Image Format

As required by XHTML-Print [22], Bluetooth Basic Printing Profile-compliant printers shall support the printing of JPEG images as part of XHTML-Print documents. The MIME media-type specified shall be (image/jpeg).

11 Transport Layer Details

11.1 OBEX Operations Used

Table 35 shows the OBEX operations that are used in the Basic Printing Profile.

Operation No.	OBEX Operation	Sender	Printer
1	CONNECT	M	M
2	DISCONNECT	M	M
3	PUT	M	M
4	GET	O	M
5	ABORT	M	M

Table 35: OBEX Operations

11.2 OBEX Headers

Table 36 shows the specified OBEX headers that are used in the Basic Printing Profile.

Header no.	OBEX Headers	Client	Server
1	Count	O	O
2	Name	M	M
3	Type	M	M
4	Length	M	M
5	Time	O	O
6	Description	M	M
7	Target	C1	M
8	HTTP	O	O
9	Body	M	M
10	End of Body	M	M
11	Who	O	M
12	Connection Id	O	M
13	Authenticate Challenge	M	O
14	Authenticate Response	M	O
15	Application Parameters	O	M
16	Object Class	X	X

Table 36: OBEX Headers Used in the Basic Printing Profile

C1: The OBEX Target header's use in an OBEX CONNECT is optional only for the FilePush operation. For all other operations, including SimpleReferencePush, the OBEX Target header shall be used. It is strongly recommended that clients should use the Target header for all operations since some devices that support the Basic Printing Profile may not support the Direct Printing Service as their default OBEX server.

Note: For BPP operations, OBEX headers that contain no information, (i.e., empty) may be left out of the OBEX request or included with no content.

11.3 Initialization of OBEX

See Section 5.3 in [10].

Implementors Note: For implementations of the Bluetooth Core Specification prior to Version 1.1, OBEX packet sizes may have to be tuned to RFCOMM's buffering capability for proper flow control.

11.4 RFCOMM Channels and OBEX Target Applications

11.4.1 RFCOMM Channels

The Basic Print Profile requires a minimum of one RFCOMM channel from Sender to Printer with optionally an additional two channels from Sender to Printer and two from Printer to Sender. See Figure 4, and Figure 5, in Section 5.1.

The channel assignments and their associated OBEX Target connections are described in the following sections.

11.4.1.1 OBEX Target Header UUIDs

OBEX UUIDs used in the Target header of the OBEX CONNECT are mapped to Bluetooth Assigned Numbers as follows:

OBEX UUID	Bluetooth UUID
DPS_UUID	DirectPrinting
PBR_UUID	ReferencePrinting
RUI_UUID	ReflectedUI
STS_UUID	PrintingStatus
REF_OBJ_UUID	DirectPrintingReferencedObjectsService

Table 37: Basic Printing Profile UUID Mapping

These Bluetooth UUIDs can be found in [16].

11.4.1.2 Job Channel

This shall be the first channel established between Sender and Printer.

The RFCOMM Channel assigned to the Job Channel is part of the Basic Printing Service Record. If the Administrative User Interface Service is supported over the Job channel as well, the RFCOMM channel number assigned to the Job Channel shall be the same as the RFCOMM channel number assigned in the Administrative User Interface Service Record.

The Sender may connect or target up to three different Target UUIDs on this channel:

1. DPS_UUID (Direct Printing Service)

2. PBR_UUID (Reference Printing Service)

3. RUI_UUID (Administrative User Interface Service)

See Section 5.4 in [10] for a description of OBEX session establishment. The Printer determines whether or not OBEX authentication is to be used. If so, OBEX authentication is used by the Printer to authenticate the Sender. OBEX authentication is not used by the Sender to authenticate the Printer.

If the Printer, by default, initiates OBEX authentication, interoperability cannot be guaranteed with Senders without a user interface. Therefore it is recommended that Printers provide a way to disable OBEX authentication in this case.

The Sender shall be able to supply a password and possibly a user ID according to Section 3.5 of the IrOBEX specification[8]. This authentication may re-use the Bluetooth PIN code at the link layer; however, it shall be entered separately. The Sender shall (except when using the Simple Push model of the Direct Printing Service) use the OBEX Target header when establishing the OBEX connection.

The Target UUID assignments may be found in the Bluetooth Assigned Numbers Document [16].

If the Printer is unable to accept the job or the connection request when it is received, the printer shall respond with an appropriate OBEX response code from Section 16.3.

11.4.1.3 Status Channel

If the Sender requires status concurrently with an active Job Channel, then the Sender can open a second OBEX connection on a separate RFCOMM channel. This

RFCOMM channel number is assigned in the Basic Printing Service Record. This connection may occur any time after the Job Channel is established.

See Section 5.4 in [10] for a description of OBEX session establishment. When establishing secondary channel connections, OBEX authentication is not used. For security reasons, the printer shall only accept the OBEX connection to the Basic Printing Status service from the Sender that currently has an open Job Channel.

The Sender shall use the Target header when establishing the connection to the Status channel. The STS_UUID shall be used in the OBEX Target header for the Status channel. The Target UUID assignments may be found in the Bluetooth Assigned Numbers Document [16].

The Sender shall only use the Status Channel for the following operations:

- GetPrinterAttributes
- GetJobAttributes
- CancelJob
- GetEvent

If the Sender currently has an active (i.e., waiting for response) GetEvent operation on the Status Channel, it shall OBEX ABORT the OBEX request before issuing another operation on that channel.

The Status Channel shall be OBEX Disconnected before disconnecting the Job Channel.

11.4.1.4 Object Channel

If the print job contains referenced images that are to be retrieved by the printer, then the Printer shall OBEX CONNECT to the Sender to open a “reverse” channel.

The RFCOMM channel number assigned to the Object Channel shall be contained in the Sender’s Basic Printing Referenced Objects Service Record.

See Section 5.4 in [10] for a description of OBEX session establishment. When establishing secondary channel connections, OBEX authentication is not used. For security reasons, the Sender should only accept OBEX connections, on the Object channel, from Printers that it is currently communicating with via an open Job Channel.

The Printer shall use the Target header when establishing the OBEX connection. The REF_OBJ_UUID is used in the OBEX Target header for connections on the Object

Channel. The Target UUID assignments may be found in the Bluetooth Assigned Numbers Document [16].

The Printer shall OBEX DISCONNECT the Object Channel upon receiving a DISCONNECT on the Job Channel.

11.4.1.5 Pulling Referenced Objects

After connecting to the Sender, the Printer shall use OBEX GET requests to retrieve one or more objects, or parts of those objects, from the Sender.

The Name header shall be set to the same value found in the file referencing the object, after translating the value from UTF-8 encoding to UFT-16.

Example

If a reference to an object is `<img src=”img5.jpg” …/>`,

the OBEX Name header shall be set to “img5.jpg”.

For the Printer to be able to request a part of an object, three tags in the Application Parameters header may be used. Two of these tags allow the Printer to OBEX GET an object from the Sender starting at a byte offset in the object and asking only for a limited number of bytes starting from that byte offset. The third tag encodes the size of the referenced object. The Application Parameters header tags are described in Section 11.12.

11.4.1.6 RUI Referenced Job Channel

When the Reflected User Interface (See Section 9) option is used, jobs may be retrieved by the Printer from the Sender. This is done using the same operation (GetReferencedObjects) as is used over the Object Channel, but instead of retrieving objects referenced as part of the job, the entire job content is retrieved. Additionally, if the reference is local to the Sender, the referenced job may include referenced objects (i.e., local images) that are retrieved using the Object Channel.

The RFCOMM channel number assigned to the RUI Referenced Job Channel is in the Sender’s Basic Printing Referenced Objects Service Record.

See Section 5.4 in [10] for a description of OBEX session establishment. When establishing secondary channel connections, OBEX authentication is not used. For security reasons, the Sender should only accept OBEX connections, to the Bluetooth Basic Printing Referenced Jobs service, from Printers that it is currently communicating with via an open Job Channel.

The Printer shall use the Target header when establishing the OBEX connection. The REF_OBJ_UUID is used in the OBEX Target header for connections on the RUI Referenced Job Channel. The Target UUID assignments are found in the Bluetooth Assigned Numbers Document [16].

In the case that the Sender is not able to honor the Printer's request for referenced content (i.e., that part of the file does not exist or is currently unavailable), then the Sender shall respond with an appropriate OBEX response code, listed in Section 16.3.

11.5 Pushing Data

See Section 5.5 in [10].

11.6 Pulling Data

See Section 5.6 in [10].

11.7 Disconnection

See Section 5.6 in [10].

11.8 Mapping Application Operations to OBEX Headers

OBEX headers are used in the following manner for each of the application-layer actions.

11.8.1 Simple Push Transfer Operations

Operation	Type Header	Body Header	Description Header (UTF-16-Encoded)	Name Header (UTF-16-Encoded)	Application Parameters Header
FilePush	PUT REQUEST: MIME Media Type of document	PUT REQUEST: Document Data	PUT REQUEST: Optional, document type-dependent information	PUT REQUEST: Optional, document name	PUT REQUEST: NO JobId TAG

Operation	Type Header	Body Header	Description Header (UTF-16-Encoded)	Name Header (UTF-16-Encoded)	HTTP Header
SimpleReferencePush	PUT REQUEST: text/x-ref-simple text/x-ref-xml text/x-ref-list	PUT REQUEST: URL of reference			Authenticate Credentials, if required.
		URL of failing attempt			Authentication Challenge
Channel: Job Target Header Usage (in OBEX CONNECT): DPS_UUID for FilePush. PBR_UUID for SimpleReferencePush. Channel: Object REF_OBJ_UUID for GetReferencedObjects.					

11.8.2 Job-Based Data Transfer Operation Mapping

Operation	BPP Channel	Type Header	Body Header	Description Header	Name Header (UTF-16-Encoded)	Application Parameters Header
CreateJob	Job	GET REQUEST: x-obex/bt-SOAP	GET REQUEST: CreateJob SOAP request			
			GET RESPONSE: CreateJob SOAP response			GET RESPONSE: JobId
SendDocument	Job	PUT REQUEST: MIME Media-Type of document	PUT REQUEST: print content	PUT REQUEST: Optional, document type dependent information	PUT REQUEST: Optional, document name	PUT REQUEST: JobId

Operation	BPP Channel	Type Header	Body Header	Description Header	Name Header (UTF-16-Encoded)	Application Parameters Header
GetJobAttributes	Job or Status	GET REQUEST: x-obex/bt-SOAP	GET REQUEST: GetJobAttributes SOAP request			
			GET RESPONSE: GetJobAttributes SOAP response			
GetPrinter-Attributes	Job or Status	GET REQUEST: x-obex/bt-SOAP	GET REQUEST: GetPrinterAttributes SOAP request			
			GET RESPONSE: GetPrinterAttributes SOAP response			
CancelJob	Job or Status	GET REQUEST: x-obex/bt-SOAP	GET REQUEST: CancelJob SOAP request			
			GET RESPONSE: CancelJob SOAP response			
GetEvent	Status	GET REQUEST: x-obex/bt-SOAP	GET REQUEST: GetEvent SOAP request			
			GET RESPONSE: GetEvent SOAP response			
GetReferenced-Objects	Object	GET REQUEST: x-obex/ referencedobject			GET REQUEST: URI	GET REQUEST: Offset, Count, FileSize
			GET RESPONSE: Image data			GET RESPONSE: FileSize

Operation	BPP Channel	Type Header	Body Header	Target Header (OBEX CONNECT)	Application Parameters Header	HTTP Header
SendReference	Job	PUT REQUEST: text/x-ref-simple text/x-ref-xml text/x-ref-list	PUT REQUEST: URL of the reference	UUID of service: PBR_UUID	PUT REQUEST: JobId	PUT REQUEST: Authenticate Credentials
			PUT RESPONSE: URL of failing attempt			
Target Header Usage (in OBEX CONNECT): STS_UUID for operations on Status Channel DPS_UUID or PBR_UUID for operations on Job Channel. (depending on service) REF_OBJ_UUID for operations on Object Channel						

11.8.3 Enhanced Layout Operation Mapping

Operation	BPP Channel	Type Header	Body Header	Description Header	Name Header (UTF-16-Encoded)	Application Parameters Header
CreatePreciseJob	Job	GET REQUEST: x-obex/bt-SOAP	GET REQUEST: CreatePreciseJob SOAP request			
			GET RESPONSE: CreatePreciseJob SOAP response			GET RESPONSE: JobId
GetMargins	Job	GET REQUEST: x-obex/bt-SOAP	GET REQUEST: GetMargins SOAP request			
			GET RESPONSE: GetMargins SOAP response			
Target Header (in OBEX CONNECT): DPS_UUID or PBR_UUID depending on service.						

11.8.4 Reflected User Interface Operation Mapping

11.8.4.1 General form for GetRUI Operation

Operation	BPP Channel	Type Header	Body Header	Target Header (in OBEX CONNECT)	Name Header (UTF-16-Encoded)	HTTP Header
GetRUI (General for all usage cases)	Job or RUI* See Section (5.1.3.2)	GET REQUEST: x-obex/RUI , Reference encoding type from Section (8.5.1), Form encoding type.	GET REQUEST: Optional: Form Data, Local Print Reference, or Network Print Reference	UUID of service: RUI_UUID, DPS_UUID, or PBR_UUID	Conditional: Name of a specific URL interface.	Authenticate Credentials Accept: Supported RUI formats Accept-Language: Preferred locale
			GET RESPONSE: Control Page Info.			

11.8.4.2 Header Usage for Printer Administration via Remote UI

Operation	BPP Channel	Type Header	Body Header	Target Header (in OBEX CONNECT)	Name Header (UTF-16-Encoded)	HTTP Header
GetRUI (Initial)	Job or RUI* See Section (5.1.3.2)	GET REQUEST: x-obex/RUI	GET REQUEST:	UUID of service: RUI_UUID	Name of a Top-Level URL.	Accept: Supported RUI formats Accept-Language: Preferred locale
			GET RESPONSE: Control Page Info.			

Operation	BPP Channel	Type Header	Body Header	Target Header (in OBEX CONNECT)	Name Header (UTF-16-Encoded)	HTTP Header
GetRUI (Subsequent)	Job or RUI* See Section (5.1.3.2)	GET REQUEST: Encoding type of the form data	GET REQUEST: Form Data	UUID of service: RUI_UUID	Optional: Name of a specific interface	
			GET RESPONSE: Status.			

11.8.4.3 Header Usage for Direct Printing via Remote UI

Operation	BPP Channel	Type Header	Body Header	Target Header (in OBEX CONNECT)	Name Header (UTF-16-Encoded)	HTTP Header
GetRUI (Initial)	Job	GET REQUEST: Reference encoding type from Section (8.5.1)	GET REQUEST: Local Reference	UUID of service: DPS_UUID	Name of a Top-Level URL.	Accept: Supported RUI formats Accept Language: Preferred locale
			GET RESPONSE: Control Page Info.			
GetRUI (Subsequent)	Job	GET REQUEST: Encoding type of form data	GET REQUEST: Form Data	UUID of service: DPS_UUID	Optional: Name of a specific interface	
			(Response pending)			
Action by the Printer						
Operation	BPP Channel	Type Header	Body Header	Target Header (in OBEX CONNECT)	Name Header (UTF-16-Encoded)	Application Parameters Header
GetReferenced-Objects	RUI Ref. Job	GET REQUEST: x-obex/ referencedobject		REF_OBJ_UUID	GET REQUEST: URI	GET REQUEST: Offset, Count, FileSize
			GET RESPONSE: data			GET RESPONSE: FileSize
Response from Printer on Job Channel for pending GetRUI operation						
	Job		GET RESPONSE: Status.			

11.8.4.4 Header Usage for Reference Printing via Remote UI

Operation	BPP Channel	Type Header	Body Header	Target Header (in OBEX CONNECT)	Name Header (UTF-16-Encoded)	HTTP Header
GetRUI (Initial)	Job	GET REQUEST: Reference encoding type from Section (8.5.1)	GET REQUEST: PBR Reference	PBR_UUID	Name of Top-Level URL.	Authenticate Credentials Accept: Supported RUI formats Accept-Language: Preferred locale
			GET RESPONSE: Control Page Info. (printer)			
GetRUI (Subsequent, Print Service control page)	Job	GET REQUEST: Encoding type of the form data	GET REQUEST: Form Data	PBR_UUID	Optional: Name of a specific interface	
			GET RESPONSE: Control Page Info. (print service)			
GetRUI (Subsequent Printer)	Job	GET REQUEST: Encoding type of the form data.	GET REQUEST: Form Data	PBR_UUID	Optional: Name of a specific interface.	Authenticate Credentials
			GET RESPONSE: Status			

11.9 SOAP Message Exchange

For operations listed in Section 7 that require the exchange of SOAP encoded messages, an OBEX GET request shall be used.

11.9.1 SOAP Stream Requirements

The HTTP/1.1 header strings shown as part of the SOAP data shall be included in the Body header as part of the SOAP request or response (not in the OBEX HTTP header). Case sensitivity and line spacing of SOAP messages follow the conventions specified in HTTP/1.1 [30]. The order of the body elements in insignificant. Required elements shall occur exactly once (i.e., no duplicates), and optional elements may occur at most once.

HTTP Header	Requirement	Description
CONTENT-LENGTH	Required	Contains the length of the SOAP body, in bytes.
CONTENT-TYPE	Required	Contains ‘text/xml; charset=”utf-8”’.

SOAPACTION	Required	Contains the service type, hash-mark, and name of action to be invoked; all enclosed in double quotes as shown in the examples associated with each SOAP encoded operation.
CONTENT-LANGUAGE	Optional	Contains user's preferred language (per RFC 1766 [38]), which the printer may use in some manner in communications with the user.

Figure 16: HTTP Headers in SOAP Messages

Figure 17 below illustrates a typical session: In this example, the request and response phases of the GET both use three packets to convey the SOAP request and response.

Client Request:	**Bytes**	**Meaning**
Opcode	0x03	**GET**, Final bit not set.
	0xnnnn	Length of packet.
	0x42	Hi for **Type** header.
	0x0012	Length of **Type** header.
	"x-obex/bt-SOAP"	Type of object, null terminated US-ASCII text.
	0x48	Hi for **Body** header.
	0xnnnn	Length of **Body** header.
	0x……	**Body** (first chunk of SOAP request)
Server Response:		
Response Code	0x90	CONTINUE, Final bit set.
	0x0003	Length of response packet.
Client Request:		
Opcode	0x03	**GET**, Final bit not set.
	0xnnnn	Length of packet.
	0x48	Hi for Object **Body** header.
	0xpppp	Length of **Body** header.
	0x….	**Body** (second chunk of SOAP request)
Server Response:		
Response Code	0x90	CONTINUE, Final bit set.
	0x0003	Length of response packet.
Client Request:		
Opcode	0x83	**GET**, Final bit set.
	0xnnnn	Length of packet.
	0x49	Hi for Object **End-of-Body** header.
	0xpppp	Length of **End-of-Body** header.
	0x….	**End-of-Body** data (final chunk of SOAP request)

This completes the request-phase of the OBEX GET. The GET response-phase now proceeds:

Server Response:		
Response Code	0x90 0xnnnn 0x4C 0x0009 0x03 0x04 0x00000001 0x48 0xpppp 0x……	CONTINUE, Final bit set Length of response packet Hi for **Application Parameter** header Length of **Application Parameter** header Tag for JobId Length of JobId JobId – type I4 Hi for Object **Body** header Length of **Body** header Body object (first chunk of SOAP response)
Client Request:		
Opcode	0x83 0x0003	**GET**, Final bit set Length of packet
Server Response:		
Response Code	0x90 0xnnnn 0x48 0xpppp 0x……	CONTINUE, Final bit set Length of response packet Hi for Object **Body** header Length of **Body** header Body object (second chunk of SOAP response)
Client Request:		
Opcode	0x83 0x0003	**GET**, Final bit set Length of packet
Server Response:		
Response Code	0xA0 0xnnnn 0x49 0xpppp 0x……	SUCCESS, Final bit set Length of response packet Hi for Object **End-of-Body** header Length of **End-of-Body** header **End-of-Body** object (final chunk of SOAP response)

Figure 17: SOAP-Encoded Message Exchange Using OBEX GET

11.10 Status Reporting During a Job

If the Printer encounters an error during a SendDocument or GetReferencedObjects operation it has several options to react upon such an error depending on the existence of a Status Channel, print buffer capabilities, etc. These alternatives are described below.

In general, status should be reported at the highest level available. (i.e., use operation status codes if the operation can be completed, else OBEX response codes for transport layer status).

1. The Sender uses the Simple Push model.
   - If the Printer has sufficient buffer space it may continue responding to the Sender's PUT requests with a CONTINUE response code until the complete object is received. The Printer may indicate, in the OBEX Description header of the response, information on the encountered error. The Sender may indicate this information to the user.
   - The Printer may return an OBEX error response code to abort the transmission. This option might be used if the Printer does not have sufficient buffer space for the remaining part of the document and the error is regarded as too severe to not be resolved by a user immediately.
   - The Printer may not return an OBEX response to the last submitted PUT request of the Sender and keep the last PUT hanging (i.e., not completed). It is recommended that the Printer indicate this to the user (e.g., flashing LED). In addition, the Sender may indicate this to the user as well, who has the choice to either fix the error or abort the transmission. If the transmission is aborted, the Sender shall disconnect the OBEX session and close the Bluetooth connection. Following the OBEX session disconnect, the Printer shall cancel the job. If the error is fixed before the OBEX session is disconnected, the Printer shall continue by responding to the last-submitted PUT request.

     Note: This can be a typical scenario where the Printer encounters an error (e.g., out of paper) that can be fixed by the user immediately to avoid sending the whole document again, as would need to occur if the transmission were aborted.

2. The Sender uses the Job-Based Transfer model but has not submitted a GetEvent operation.
   - The Printer and Sender may react in the same way as in the case of the Simple Push model.
   - If the Printer does not respond to the latest PUT request, the client may submit a GetEvent, GetJobAttributes, or GetPrinterAttributes on the Status Channel to retrieve status information. Depending on the user's interaction, the Sender can disconnect the OBEX session. If the error on the Printer is fixed before the

OBEX session is disconnected, the Printer shall respond to the last-submitted PUT request.

- During the SendDocument operation the client may always submit a GetEvent, GetJobAttributes, or GetPrinterAttributes to retrieve status information (e.g., if the Printer provides information in the Description header of the OBEX response and the Sender wants to request more detailed status information).

3. The Sender uses the Job-Based Transfer model and has submitted a GetEvent operation on the Status Channel.

   - The Printer may not respond to the last submitted PUT request but the Printer shall respond to the last submitted GetEvent request of the GetEvent operation indicating the changed status of the Printer. Upon receiving that GetEvent response, the client may disconnect the OBEX session and close the Bluetooth connection. If the Bluetooth connection is closed, the Printer shall cancel the job. Alternatively, the Sender may keep the OBEX session open and submit a new GetEvent request (belonging to the same GetEvent operation). If the status of the Printer has changed (e.g., the error has been corrected) the Printer shall submit a new GetEvent response indicating the changed status. If the printer error is corrected, the Printer shall, in addition to providing the GetEvent response, respond to the last submitted PUT request.
   - If the Printer has sufficient buffer space it may continue responding to the Sender's PUT requests with a CONTINUE response code until the complete object is received. In addition, the Printer shall respond to the last submitted GetEvent request indicating the changed status of the Printer. Upon receiving that GetEvent response, the Sender may disconnect the OBEX session and close the Bluetooth connection. If the Bluetooth connection is closed, the Printer shall cancel the job. Alternatively, the Sender may keep the OBEX session open, continue submitting PUT requests on the Job Channel, and submit a new GetEvent request (belonging to the same GetEvent operation). If the status of the Printer has changed (e.g., the error has been corrected) the Printer shall submit a new GetEvent response indicating the changed status if the OBEX session was not disconnected by the Sender (e.g., if the SendDocument operation was finished).

4. The Sender uses the RUI-based model (i.e., the GetReferencedObjects operation is used by the Printer and not the SendDocument operation by the Sender).

   - The Printer may not submit the next GetReferencedObjects request but respond to the last submitted GetRUI request of the GetRUI operation indicating the changed status of the Printer. Upon receiving that GetRUI response, the Sender may disconnect the OBEX session and close the Bluetooth connection and the Printer shall disconnect the RUI Referenced Job Channel and cancel the job. Alternatively, the Sender may keep the OBEX session and submit a new GetRUI request (belonging to the same GetRUI operation). If the status of the Printer has changed (e.g., the error has been

corrected) the Printer may submit a new GetRUI response indicating the changed status. If the Printer error is corrected, the Printer shall continue submitting GetReferencedObjects requests to the Sender.

- If the Printer has sufficient buffer space it may continue submitting GetReferencedObjects requests until the complete object is received. In addition, the Printer may respond to the last submitted GetRUI request indicating the changed status of the Printer. Upon receiving that GetRUI response, the Sender may disconnect the OBEX session and close the Bluetooth connection. If the Bluetooth connection is closed, the Printer shall disconnect the RUI Referenced Job Channel and cancel the job. Alternatively, the Sender may keep the OBEX session open and submit a new GetRUI request (belonging to the same GetRUI operation). If the status of the Printer has changed (e.g., the error has been corrected) the Printer may submit a new GetRUI response indicating the changed status if the OBEX session was not disconnected by the Sender.

11.11 Typical Message Sequence

This section describes a typical session for a simple print job, where the client does not wish to monitor status as the job proceeds. Following the OBEX CONNECT, a CreateJob action is used to obtain a JobId and this is then followed by a SendDocument action.

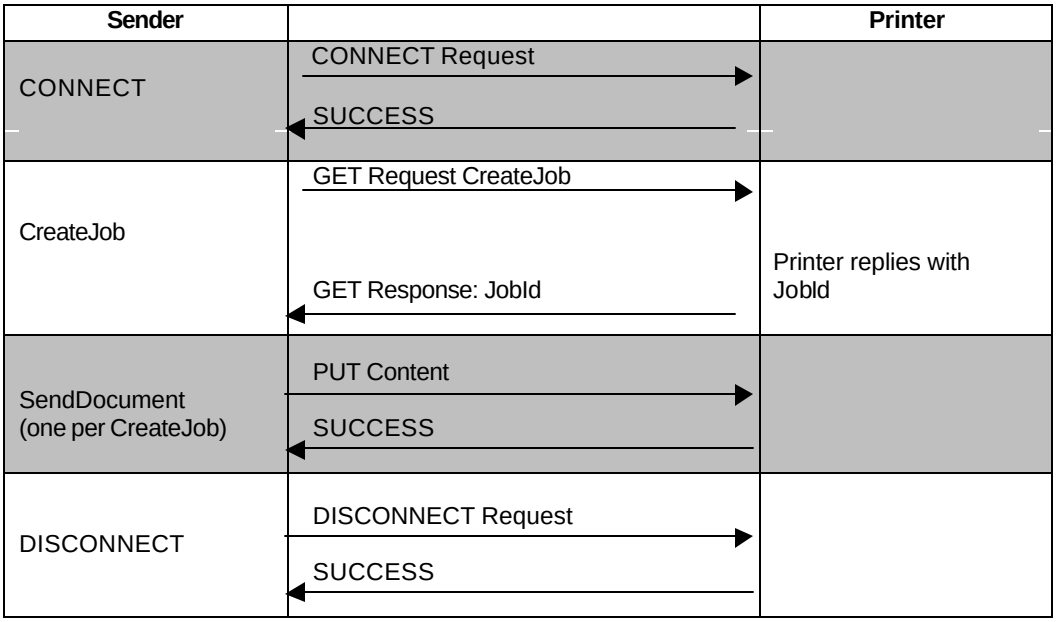

Figure 18: Session Signaling Diagram

11.12 OBEX Application Parameters Header Usage

This section defines the Application Parameters header tags used in the Basic Printing Profile.

Tag No.	Description	Length of Value in Bytes	Format	Value
1	Offset	4 bytes	An unsigned integer	The byte offset into the image or file.
2	Count	4 bytes	A signed integer	The number of bytes of the image or file to be sent.
3	JobId	4 bytes	An unsigned integer	The job identifier of the print job.
4	FileSize	4 bytes	A signed integer	The size (in bytes) of object or file.

Table 38: OBEX Application Parameters Header Tags

All Application Parameters header tag values defined in Table 38 are transferred in standard network byte order (Big-Endian), with more significant (high-order) bytes being transferred before less-significant (low-order) bytes.

12 Service Discovery

12.1 SD Service Records

A Printer that supports the Basic Printing Profile shall register one or more Printer Service Records, including the entries listed in Table 39.

A Sender that provides a Basic Printing Profile Referenced Objects service shall register a Service Record, including the entries listed in Table 40.

A Printer that also supports a Printer Administration Reflected User Interface shall register one or more P rinter Administrative Reflected User Interface Service Records, including the entries listed in Table 41.

The “Status” column of each of the Service Record tables indicates if the entry is mandatory or optional.

12.1.1 Printer Service Record

If a Bluetooth device supporting the Bluetooth Basic Printing Profile can serve more than one Printer, it may register more than one Service Record according to the table below. The Service Records shall have different values for the RFCOMM channel number to be used when connecting to the printing service, in addition to other entries in the Service Records as appropriate. Senders are not required to enable the selection of which service (Printer) to use if more than one Service Record is found.

Item	Definition	Type Size	Value*	AttrID	Status	Default Value
Service Class ID List				See [16]	M	
Service Class #0		UUID	ReferencePrinting		O	See [16]
Service Class #1		UUID	PrintingStatus		M	See [16]
Service Class #2		UUID	DirectPrinting		M	See [16]
ServiceID**	See [6]	UUID	Varies	See [16]	O	
Protocol Descriptor List				See [16]	M	
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	Job Channel	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	

Item	Definition	Type Size	Value*	AttrID	Status	Default Value
Service Name	Displayable Text name	String	Varies	See [16]	O	See Section 12.2.1
Bluetooth Profile Descriptor List				See [16]	O	
Profile ID #0	Supported Profile	UUID	BasicPrinting			See [16]
Version #0	Profile Version	uint16	Varies			0x0100
Additional Protocol Descriptor Lists Protocol Descriptor List #0	See [6], Errata 2186			See [16]	M	
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	Status Channel	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	
Document Formats Supported	See Section 12.2.2	String	See Section 12.2.2	0x0350	M	
Character Repertoires Supported	See Section 12.2.3	Uint128	See Section 12.2.3	0x0352	M	
XHTML-Print Image Formats Supported	See Section 12.2.4	String	See Section 12.2.4	0x0354	M	
Color Supported	Printer supports full color output	Boolean	See Section 12.2.5	0x0356	O	false
1284ID	Printer's 1284 ID	String	Printer model specific 1284 ID string. See [23]	0x0358	M	
Printer Name	User-Friendly Printer Name	String	Varies	0x035A	O	unknown
Printer Location	Physical Printer Location	String	Varies	0x035C	O	unknown

Item	Definition	Type Size	Value*	AttrID	Status	Default Value
Duplex Supported	Printer Supports Duplex	Boolean		0x035E	O	false
Media Types Supported	Printers Paper Types	String	See Section 12.2.6	0x0360	O	unspecified
MaxMediaWidth	Maximum paper width (mm)	Uint16	See Section 12.2.7	0x0362	O	See Section 12.2.7
MaxMediaLength	Maximum paper length (mm)	Uint16	See Section 12.2.7	0x0364	O	See Section 12.2.7
Enhanced Layout Supported		Boolean		0x0366	O	false
RUI Formats Supported	List of Supported RUI formats in order of preference for Job config.	String	See Section 14.1 of RFC 2616 [30]	0x0368	O	unknown
Reference Printing RUI Supported		Boolean		0x0370	O	false
Direct Printing RUI Supported		Boolean		0x0372	O	false
Reference Printing Top URL		URL	See Section 12.2.8	0x0374	C1	
Direct Printing Top URL		URL	See Section 12.2.8	0x0376	C2	

Table 39: Basic Printing Service Record (Printer)

** Values that are of the type UUID are defined in the Assigned Numbers specification [16].*

*** The ServiceID entry is used to uniquely identify a Basic Printing Service Record if more than one record is exposed by a single SDP Server.*

C1: Mandatory if Reference Printing RUI Supported Boolean is reported true.

C2: Mandatory if Direct Printing RUI Supported Boolean is reported true.

12.1.2 Referenced Objects Service Record

This service record shall be registered in the SDDB of the Sender if it supports a Referenced Objects Service.

Item	Definition	Type Size	Value*	AttrID	Status	Default Value
Service Class ID List				See [16]	M	
Service Class #0		UUID	DirectPrinting-ReferencedObjects-Service		M	
ServiceID**	See [6]	UUID	Varies	See [16]	O	
Protocol Descriptor List				See [16]	M	
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	Object Channel	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	
Service Name	Displayable Text name	String	Varies	See [16]	O	See Section 12.2.1
Bluetooth Profile Descriptor List				See [16]	O	
Profile ID #0	Supported Profile	UUID	BasicPrinting			See [16]
Version #0	Profile Version	uint16	Varies			0x0100
Additional Protocol Descriptor Lists	See [6], Errata 2186			See [16]	M	
Protocol Descriptor List #0						
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	RUI Referenced Job Channel	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	

Table 40: Basic Printing Referenced Objects Service Record (Sender)

*** The ServiceID entry is used to uniquely identify a Basic Printing Referenced Objects Service Record if more than one record is exposed by a single SDP Server.*

12.1.3 Printer Administrative User Interface Service Record

This service record shall be registered in the SDDB if the Printer supports a Reflected User Interface Service that is used for overall Printer administration and configuration, whether or not it also supports reflected UIs for either Direct Printing or Reference Printing.

Item	Definition	Type Size	Value*	AttrID	Status	Default Value
Service Class ID List				See [16]	M	
Service Class #0		UUID	ReflectedUI		M	See [16]
ServiceID**	See [6]	UUID	Varies	See [16]	O	
Protocol Descriptor List				See [16]	M	
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	Channel	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	
Service Name	Displayable Text Name	String	See Section 12.2.1	See [16]	O	See Section 12.2.1
RUI Formats Supported	List of supported RUI formats in order of preference for Printer Admin.	String	See Section 14.1 of RFC 2616 [30]	0x0368	M	
Printer Admin RUI Top URL		URL	See Section 12.2.8	0x0378	M	

Table 41: Printer Administrative User Interface Service Record

*** The ServiceID entry is used to uniquely identify a Printing Administrative User Interface Service Record if more than one record is exposed by a single SDP Server.*

12.2 Service Record Attribute Details

12.2.1 Service Name

The Service Name string provides a displayable text name that can be directly transmitted to the user. A suggested value for Printers that include this optional string and choose not to localize is “Basic Printing”.

12.2.2 Document Formats Supported

Each Data Format, Page Description Language (PDL), or Image format supported by the printer should be included in the Document Formats Supported list if support is to be discovered by a Bluetooth Basic Print Profile client device.

The Document Format string includes the MIME media-type followed by an optional US-ASCII string, representing any applicable version of the data format. The MIME media-type shall be separated from the version string with the colon character. Some common examples of this include:

Document Format	MIME Media Type:Version
XHTML-Print	application/vnd.pwg-xhtml-print+xml:1.0
Basic Text	text/plain
vCard	text/x-vcard:2.1
vCard 3.0	text/x-vcard:3.0
vCalendar	text/x-vcalendar:1.0
iCalendar 2.0	text/calendar:2.0
vMessage	text/x-vmessage:1.1
PostScript 2	application/PostScript:2
PostScript 3	application/PostScript:3
PCL5E	application/vnd.hp-PCL:5E
PCL3C	application/vnd.hp-PCL:3C
PDF	application/PDF
JPEG	image/jpeg
GIF89a	image/gif:89A

Table 42: Common Media-Types

Note: The MIME media-types for certain vObjects are not officially registered with IANA; however, the indicated non-standard media-types are commonly accepted.

Vendor-specific document formats currently without MIME media-types should be registered through IANA [24]. A current list of registered media-types is also available through IANA.

For entry into the Service Record, multiple document formats shall be represented by a comma-delimited list of MIME media-type:version strings. For example, a printer supporting XHTML-Print as well as PCL5E and PDF could include either of the following strings in the Service Record:

1. application/vnd.pwg-xhtml-print+xml:1.0,application/vnd.hp-PCL:5E,application/PDF

2. application/vnd.hp-PCL:5E,application/PDF,application/vnd.pwg-xhtml-print+xml:1.0

Note: While the version of the document format indicates support for a specific version of that format, the lack of a version number does not necessarily indicate the lack of support for that version. (e.g., application/PostScript may support both PostScript Versions 2 and 3).

12.2.3 Character Repertoires Supported

This field is provided to enable a Sender to determine which characters or glyphs a Printer supports for access from XHTML-Print, and the optional Basic Text, vCard, vCalendar, and vMessage formats. Support for glyphs that are indicated in this field does not guarantee support in other data formats.

Character Repertoires Supported is a 128-bit data value, where each bit represents the set of glyphs comprised within the corresponding character repertoire. A bit set to 1 (one) indicates that all characters of the corresponding character repertoire can be printed. A bit set to 0 (zero) indicates that a part or all of the characters in the corresponding character repertoire cannot be printed.

Note: A bit set to one indicates that the associated glyphs are available via UTF-8 encoding only; it does not imply that they are accessible via the corresponding character set encoding.

Table 43 shows the correspondence between bits in the Character Repertoires Supported field and character repertoires. The character set standards referenced in Table 43 generally indicate both character symbols (glyphs) and a mapping to an integer character code. The character code mapping is unused for this application. This cross-standard referencing is necessary because the Unicode mapping of interesting repertoires of character symbols to UTF-8 encoding is not a contiguous block of numbers. References for most of the indicated Character Repertoires may be found in the IANA Character Sets list [35].

The most current version of the bit assignments for the Character Repertoires Supported field may be found in the Host Operating Environment Identifiers section of the Bluetooth Assigned Numbers Document [16]. Unassigned bits will be assigned by the maintainer of [16] according to procedures described by the Bluetooth SIG. The general guideline is that each bit should indicate a subset of the 4-byte Unicode space of use to providers of Senders, Printers, fonts, and Internet content, with appropriate support from national standards groups. It is strongly recommended that new character repertoires also be filed with IANA (see [36]).

The capability to print 7-bit US-ASCII characters is not listed as part of the following table; however, that capability is mandatory for all Printers supporting any part of this Profile.

Bit Number	Character Repertoire	Description
Bit0	ISO-8859-1	Latin alphabet No. 1
Bit1	ISO-8859-2	Latin alphabet No. 2
Bit2	ISO-8859-3	Latin alphabet No. 3
Bit3	ISO-8859-4	Latin alphabet No. 4
Bit4	ISO-8859-5	Latin/Cyrillic alphabet
Bit5	ISO-8859-6	Latin/Arabic alphabet
Bit6	ISO-8859-7	Latin/Greek alphabet
Bit7	ISO-8859-8	Latin/Hebrew alphabet
Bit8	ISO-8859-9	Latin alphabet No. 5
Bit9	ISO-8859-10	Latin alphabet No. 6
Bit10	ISO-8859-13	Latin alphabet No. 7
Bit11	ISO-8859-14	Latin alphabet No. 8
Bit12	ISO-8859-15	Latin alphabet No. 9
Bit13	GB_2312-80	Chinese (People's Republic of China)
Bit14	Shift_JIS	Japanese
Bit15	KS_C_5601-1987	Korean
Bit16	Big5	Chinese (Taiwan)
Bit17	TIS-620	Thai
Bits18-127	Reserved	(These bits will be allocated by the Bluetooth SIG. The Printer should set them to zero if not yet allocated, or if relevant character repertoire is not supported.)

Table 43: Character Repertoires Supported

12.2.4 XHTML-Print Image Formats Supported

The XHTML-Print Image format shall be represented by a string that includes the MIME media type followed by a US-ASCII string representing any applicable version of the image format. See Section 12.2.2 for details on formatting and versioning of this string.

An image format shall only be included in this list if it can be supported as part of an XHTML-Print document. If an image format is also supported as a standalone document format, it may also be included in the Document Formats Supported list.

12.2.5 Color Supported

This Boolean indicates the support for full color output. Printers that support highlight color or support grayscale (without also supporting full color) shall not include this Boolean with a value of “true”. If not included the value is assumed “false”.

12.2.6 Media Types Supported

Identifies the types of “paper” that the printer can support. The Printer does not necessarily have to have this type of media loaded. Refer to the Media Standardized Names standard [27] for a complete listing of all Media Type Names.

For entry into the Service Record, multiple media types shall be represented by a comma-delimited list of media type values. For example, a printer supporting plain copy paper as well as envelopes and cardstock could indicate:

(stationery,cardstock,envelope)

If this string is not present in the service record, the media type “unspecified” shall be assumed.

12.2.7 MaxMediaWidth and MaxMediaLength Attributes

These values indicate the dimensions of the largest medium that can be used in the Printer. If a Printer does not include these values in the Service Record, a medium with the maximum width of (iso_a4_210x297mm) and the maximum length of (na_letter_8.5x11in) should be assumed.

12.2.8 Reflected User Interface Top URL Attributes

These values indicate the URL of the top-level (default) page for the supported RUI. The URL has the following general form:

OBEX:<service-uuid>/<path>

12.3 SDP Protocol Data Units

Table 44 shows the specified SDP PDUs (Protocol Data Units), which are required in the Basic Printing Profile.

PDU no.	SDP PDU	Sender	Printer
1	SdpErrorResponse	M	M
2	SdpServiceSearchAttributeRequest	M	M
3	SdpServiceSearchAttributeResponse	M	M
4	SdpServiceSearchRequest	O	M
5	SdpServiceSearchResponse	O	M
6	SdpServiceAttributeRequest	O	M
7	SdpServiceAttributeResponse	O	M

Table 44: SDP PDUs

13 Link Manager

13.1 Authentication, Encryption, and Bonding

Link-level authentication and encryption are mandatory to support and optional to use.

If the Printer initiates authentication it may do so by using a fixed PIN. The PIN may be changed using a proprietary method in the Printer. The use of a fixed PIN is explained in Section 14.2.1 of [1]. How the fixed PIN is communicated to the user of the Sender is not specified.

Neither the Printer nor the Sender is required to initiate authentication, but all devices shall be able to respond to an authentication request.

Senders without a user interface should not initiate link-level authentication. If the Printer, by default, initiates OBEX authentication, interoperability cannot be guaranteed with Senders without a user interface. Therefore, it is recommended that Printers provide a way to disable OBEX authentication in this case.

14 Generic Access Profile Interoperability Requirements

The requirements of this Profile for GAP interoperability are the same as those stated in GOEP [10], except that the Non-pairable Mode for the Server is optional (instead of Mandatory as stated in the GAP requirements).

15 Acronyms And Abbreviations

Abbreviation or Acronym	Meaning
ASCII	American Standard Code for Information Exchange
BB	BaseBand
BPP	Basic Printing Profile
CHTML	Compact Hypertext Markup Language
CoD	Class of Device/Service
CRLF	Carriage Return Linefeed
CSS	Cascading Style Sheets
DPS	Direct Printing Service
FTP	File Transfer Protocol
GOEP	Generic Object Exchange Profile
HCI	Host Controller Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP with Secure Socket Layer (SSL)
IANA	Internet Assigned Numbers Authority
IPP	Internet Printing Protocol
L2CAP	Logical Link and Control Adaptation Protocol
LC	Link Controller
LM	Link Manager
LMP	Link Manager Protocol
MIB	Management Information Base
MIME	Multipurpose Internet Mail Extensions
MSC	Message Sequence Chart
OBEX	Object Exchange Protocol
OSI	Open Systems Interconnection
PBR	Print-By-Reference or Reference Printing option of BPP
PDA	Personal Digital Assistant

Abbreviation or Acronym	Meaning
PDL	Page Description Language
PIM	Personal Information Management
QoS	Quality of Service
RFCOMM	Serial Cable Emulation Protocol
RUI	Reflected User Interface option of BPP
SD	Service Discovery
SDDB	Service Discovery Database
SDP	Service Discovery Protocol
SOAP	Simple Object Access Protocol
TLS	Transport Layer Security
UI	User Interface
URI	Uniform Resource Indicators
URL	Uniform Resource Locator
UUID	Universally Unique Identifier
WML	Wireless Markup Language
XML	Extensible Markup Language

16 Appendix

16.1 Bluetooth General and Device Specific Inquiry (CoD Field)

The Class of Device/Service (CoD) field facilitates discovery of relevant Bluetooth devices via Bluetooth Inquiry mechanisms described in [13].

A Printer Device that supports BPP shall set the “Rendering” and “Object Transfer” Major Service Class bits, which are described in the Bluetooth Assigned Numbers Document [16]. Other Major Service Class bits may be set as appropriate.

A Printer Device that Supports BPP should set the Major Device Class to “Imaging” as specified in [16]. Note that a Printer Device that supports BPP in addition to other Bluetooth Profiles may advertise a different Major Device Class than “Imaging”. For instance, a device with a printer proxy function might present a Major Device Class of “Computer” or “LAN/Network access point”. Such a device shall be detected by Sender Devices that are looking for available BPP Printer Devices. I.e., a Sender Device shall not filter only for devices that have “Imaging” as their Major Device Class.

If the “Imaging” Major Device Class is advertised, the “Printer” bit shall be set in the Minor Device Class field. Other Minor Device Class bits may be set as appropriate.

BPP does not require any particular bits to be set in the CoD field of a Sender Device.

The following is an example of an algorithm that makes use of the CoD field to locate Printers supporting BPP:

1. Are “Rendering” and “Object Transfer” bits set in the Major Service Class field? If not, ignore this Bluetooth device.

2. If the Major Device Class is “Imaging”, is the “Printer” bit set in the Minor Device Class field? If not, ignore this Bluetooth device.

3. Request the SDP records from the device and see if the UUID “DirectPrinting” is present in the Service Class ID list. If so, the device supports Printing via BPP.

16.2 Operation Status Codes

The following status codes are taken from the IPP specification [19].

Code Value (hex)	Code Meaning
0x0000	successful-ok
0x0001	successful-ok-ignored-or-substituted-attributes
0x0002	successful-ok-conflicting-attributes
0x0400	client-error-bad-request
0x0401	client-error-forbidden
0x0402	client-error-not-authenticated
0x0403	client-error-not-authorized
0x0404	client-error-not-possible
0x0405	client-error-timeout
0x0406	client-error-not-found
0x0407	client-error-gone
0x0408	client-error-request-entity-too-large
0x0409	client-error-request-value-too-long
0x040a	client-error-document-format-not-supported
0x040b	client-error-attributes-or-values-not-supported
0x040c	client-error-uri-scheme-not-supported
0x040d	client-error-charset-not-supported
0x040e	client-error-conflicting-attributes
0x040f	client-error-compression-not-supported
0x0410	client-error-compression-error
0x0411	client-error-document-format-error
x00412	client-error-document-access-error
x00418	client-error-media-not-loaded
0x0500	server-error-internal-error
0x0501	server-error-operation-not-supported
0x0502	server-error-service-unavailable
0x0503	server-error-version-not-supported
0x0504	server-error-device-error
0x0505	server-error-temporary-error

Code Value (hex)	Code Meaning
0x0506	server-error-not-accepting-jobs
0x0507	server-error-busy
0x0508	server-error-job-canceled
0x0509	server-error-multiple-document-jobs-not-supported

At a minimum, the Printer shall be able to respond with at least three different error codes. (one successful code, one client error code, and one server error code).

The Sender may choose to group specific error codes into one of three categories. (successful, client error, or server error)

16.3 OBEX Response Codes

OBEX Response Code	**HTTP Response Code**	**OBEX Definition**	**BPP Code Interpretation (if different)**
0x00 to 0x0F	None	Reserved	
0x10 (0x90)	100	Continue	
0x20 (0xA0)	200	OK, Success	
0x21 (0xA1)	201	Created	
0x22 (0xA2)	202	Accepted	
0x23 (0xA3)	203	Non-Authoritative Information	
0x24 (0xA4)	204	No Content	
0x25 (0xA5)	205	Reset Content	
0x26 (0xA6)	206	Partial Content	**PBR:** Multiple Errors Occurred. More than one error occurred in processing a reference list. References could not be enumerated.
0x30 (0xB0)	300	Multiple Choices	
0x31 (0xB1)	301	Moved Permanently	
0x32 (0xB3)	303	Moved Temporarily	
0x33 (0xB3)	303	See Other	

OBEX Response Code	HTTP Response Code	OBEX Definition	BPP Code Interpretation (if different)
0x34 (0xB4)	304	Not Modified	
0x35 (0xB5)	305	Use Proxy	
0x40 (0xC0)	400	Bad Request – server could not understand the request.	**PBR:** Invalid Target ID. The receiving device does not support the target UUID. **RUI:** Unsupported RUI Service. An RUI request included an unknown target UUID.
0x41 (0xC1)	401	Unauthorized	**PBR/RUI:** Authorization Required. Credentials are required to access the Reference/User Interface.
0x42 (0xC2)	402	Payment required	
0x43 (0xC3)	403	Forbidden: operation is understood but refused	
0x44 (0xC4)	404	Not found	**DPS:** Object or File does not exist. **PBR:** Reference does not exist or cannot be accessed at this time. **RUI:** Invalid RUI Name. The interface associated with the name in the OBEX Name header of the request does not exist.
0x45 (0xC5)	405	Method not allowed	**PBR:** Specified Print Service cannot be found or is unavailable.
0x46 (0xC6)	406	Not acceptable	**PBR:** Malformed Reference. The reference cannot be parsed or the reference type listed in the request is not supported.
0x47 (0xC7)	407	Proxy authentication is required	
0x48 (0xC8)	408	Request timed out	**PBR/RUI:** Timeout. Access to the Reference/User Interface timed out before content could be accessed.
0x49 (0xC9)	409	Conflict	
0x4A (0xCA)	410	Gone	
0x4B (0xCB)	411	Length Required	
0x4C (0xCC)	412	Precondition failed	**DPS:** Object or File is not currently available: future retries may succeed. **RUI:** File not ready: retry request later.
0x4D (0xCD)	413	Requested entity is too large	**DPS:** Offset of requested object or file too large (GetReferencedObjects operation).
0x4E (0xCE)	414	Requested URL is too large	
0x4F (0xCF)	415	Unsupported media type. **Note:** Refers to MIME media-type.	**PBR:** Reference cannot be rendered. The target of the reference cannot be rendered or only partially rendered. **RUI:** Unsupported RUI Format. The Sender specified a format for RUI pages that is not supported by the Printer.
0x50 (0xD0)	500	Internal receiver error	**DPS:** Misc. internal Server error.

OBEX Response Code	HTTP Response Code	OBEX Definition	BPP Code Interpretation (if different)
0x51 (0xD1)	501	Not implemented.	**PBR:** Unsupported Protocol. The URI that was sent in the PBR request specifies a protocol scheme that is not supported.
0x52 (0xD2)	502	Bad Gateway	
0x53 (0xD3)	503	Service Unavailable	**PBR/RUI:** Offline. The printer is Offline and not accepting requests at this time.
0x54 (0xD4)	504	Gateway Timeout	
0x55 (0xD5)	505	HTTP protocol version is not supported	
0x60 (0xE0)	----	Database Full	
0x61 (0xE1)	----	Database Locked	

17 References

17.1 Normative References

[1] Bluetooth Core Specification, Baseband Specification

[2] Bluetooth Core Specification, LMP Specification

[3] Bluetooth Core Specification, L2CAP Specification

[4] Bluetooth Core Specification, RFCOMM with TS 07.10

[5] ETSI, TS 07.10, Version 6.3

[6] Bluetooth Core Specification, SDP Specification

[7] Bluetooth Core Specification, IrDA Interoperability

[8] Infrared Data Association, IrDA Object Exchange Protocol (IrOBEX), Version 1.2 with Published Errata, April 1999

[9] Infrared Data Association, IrMC (Ir Mobile Communications) Specification with Published Errata, Version 1.1, February 1999

[10] Bluetooth Profile Specification, Generic Object Exchange Profile

[11] The Internet Mail Consortium, vCard - The Electronic Business Card Exchange Format, Version 2.1, September 1996

[12] The Internet Mail Consortium, vCalendar - The Electronic Calendaring and Scheduling Exchange Format, Version 1.0, September 1996

[13] Bluetooth Profile Specification, Generic Access Profile Specification

[14] Bluetooth Profile Specification, File Transfer Profile Specification

[15] Bluetooth Profile Specification, Synchronization Profile Specification

[16] Bluetooth Profile Specification, Assigned Numbers Specification, http://www.bluetooth.org/assigned-numbers.htm

[17] Bluetooth Profile Specification, Object Push Profile Specification

[18] Bluetooth Core Specification, Appendix IX

[19] RFC 2911, Internet Printing Protocol/1.1: Model and Semantics, http://www.ietf.org/rfc/rfc2911.txt

[20] Bluetooth Profile Specification, Still Imaging Profile, Phase 1

[21] Bluetooth Profile Specification, Hardcopy Cable Replacement Profile

[22] XHTML-Print, http://www.xhtml-print.org.

[23] IEEE Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers. IEEE Std. 1284-2000

[24] Internet Assigned Numbers Authority (IANA) http://www.iana.org/

[25] RFC 2426, vCard MIME Directory Profile, http://www.ietf.org/rfc/rfc2426.txt

[26] RFC 2445, Internet Calendaring and Scheduling Core Object Specification (iCalendar), http://www.ietf.org/rfc/rfc2445.txt

[27] PWG-ISTO STD.5101.1. Media Standardized Names. ftp.pwg.org/pub/pwg/standards/pwg5101.1.pdf Currently in Draft D0.11 form at, ftp://ftp.pwg.org/pub/pwg/media-sizes/pwg-media-11.pdf

[28] W3C Recommendation “Extensible Markup Language (XML) 1.0”, http://www.w3.org

[29] W3C Note “Simple Object Access Protocol (SOAP) 1.1”, http://www.w3.org

[30] RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, http://www.ietf.org/rfc/rfc2616.txt

[31] Unicode Standard Annex #14, Line Breaking Properties, http://www.unicode.org/unicode/reports/tr14

[32] The Unicode Standard, Version 3.0, (Reading, Massachusetts: Addison-Wesley Developers Press 2000), ISBN 0-201-61633-5.

[33] RFC 2617, HTTP Authentication: Basic and Digest Access Authentication, http://www.ietf.org/rfc/rfc2617.txt

[34] RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax, http://www.ietf.org/rfc/rfc2396.txt

[35] IANA “Character Sets” document, http://www.iana.org/assignments/character-sets

[36] RFC 2978, “IANA Charset Registration Procedures”, http://ietf.org/rfc/rfc2978.txt

[37] IEEE Standards Style Manual, http://standards.ieee.org/guides/style/2000Style.pdf

[38] RFC 1766, Tags for the Identification of Languages, http://www.ietf.org/rfc/rfc1766.txt