BLUETOOTH DOC	Date / Year-Month-Day 2012-07-24	Approved Adopted	Revision V12	Document No BIP_SPEC
Prepared By BARB	E-mail Address barb-main@bluetooth.org			N.B.

BASIC IMAGING PROFILE

Abstract**:**

This profile defines the requirements necessary for *Bluetooth*® devices to support the Basic Imaging Profile usage models. The requirements are expressed by defining the features, functions, and underlying profiles which are required for interoperability among Bluetooth devices in the Basic Imaging Profile usage models.

Revision History

Revision	Date	Comments
0.5	26-Sep-2000	Major document structure enhancement
0.7	27-Jan-2001	Features finalization
0.9	18-May-2001	Functions finalization
0.95	25-Sep-2001	Feedback from prototypes implementation and testing included
1.0 Draft	05-Feb-2002	Feedback from first open interoperability testing event and correction of remaining typos
1.0 Final	15-Nov-2002	Final draft
1.0 Final_a	25-Jul-2003	Further changes after BQRB review
D11r00	15 Aug 2005	Updated for core release v1.2 or later
D11r01	13 Sept 2005	Editorial Updates
D11r02	28 Oct 2005	Editorial Updates
D11r03	10 Nov 2005	Editorial Updates
D11r04	30 Nov 2005	Editorial Updates
D11r05	10 July 2007	Editorial updates for 2.1 + EDR
D11r06	01 Aug 2008	Initial version implementing functionality in OBEX Application Enhancements DIPD
D11r07	26 Aug. 09	Updates after WG review
D11r08	10-oct-09	Correct GOEP L2CAP PSM in SDP record
D11r09	12-Jan-10	Cleanup for BARB review
D11r10	23-Feb-10	Address BARB review comments
D11r11	23-Feb-10	Minor review fixups by Tim
D11r12	19-Mar-10	Reference GOEP Selection procedure (due to RWH); exclude reliable sessions per OBEX WG telecom 12 Mar
D11r13	03-June-10	Started with TechEd-returned copy. Checked all must/shalls and connection/sessions from BARB approved CR. R08/r07 void versions
D11r14	29-June-10	Include BARB review feedback into correct baseline
D11r15	05-July-10	Include correct references from previous baseline
V11r00	26 Aug 10	Adopted by the Bluetooth SIG Board of Directors
D12r00	2012-01-25	ESR04: E710 Annex D
D12r01- D12r04	2012-05-09- 2012-05-28	Entered corrections according to TB's comments. Applied latest templates. Spell checked. Changed version numbers in Revision History to match what is in V11 released version. Address reviewer's comments: 1) Change 0x0101 to 0x0102 (3x) 2) Fix hyperlinks for references. Remove comments. Change remaining two instances of 0x0101 to 0x0102 Fixed Normative References to start at [1]. Fixed cross references in 4.5.12 and 4.5.14 to reference [1], not [6].
V12	2012-07-24	Adopted by the Bluetooth SIG Board of Directors

Contributors

Name	Company
Victor Zhodzishsky	Broadcom
Sherry Smith	Broadcom
Akane Yokota	Canon
Kenichi Fujii	Canon

Name	Company
Kazuaki Abe	Casio Computer
Matsunaga Kazuhisa	Casio Computer
Ryohei Yamamoto	Casio Computer
Tatsuo Arai	Casio Computer
Patric Olsson	(currently connectBlue AB)
Erwin Weinans	Ericsson
Marcel Wong	Ericsson
Maria Rang	Ericsson
Hiroshi Tanaka	Fujifilm
Mikio Watanabe	Fujifilm
Akinori Yoshioka	Fujifilm software
Holt Mebane	HP
Paolo Fontani	HP
Kevin Hendrix	iAnywhere Solutions
Franc Camara	Microsoft
Karl Heubaum	Microsoft
Tim Howes	Nokia Corporation
Martin Roter	Nokia Mobile Phones
Stephane Bouet	Nokia Mobile Phones
Mandar Gokhale	Qualcomm
Greg Burns	Qualcomm
Len Ott	Socket Mobile
Takayasu Sanada	Toshiba
Takuya Kawamura	Toshiba
Yosuke Tajika	Toshiba

Disclaimer and Copyright Notice

The copyright in this specification is owned by the Promoter Members of *Bluetooth*® Special Interest Group (SIG), Inc. ("*Bluetooth* SIG"). Use of these specifications and any related intellectual property (collectively, the "Specification"), is governed by the Promoters Membership Agreement among the Promoter Members and *Bluetooth* SIG (the "Promoters Agreement"), certain membership agreements between *Bluetooth* SIG and its Adopter and Associate Members (the "Membership Agreements") and the *Bluetooth* Specification Early Adopters Agreements (1.2 Early Adopters Agreements) among Early Adopter members of the unincorporated *Bluetooth* SIG and the Promoter Members (the "Early Adopters Agreement"). Certain rights and obligations of the Promoter Members under the Early Adopters Agreements have been assigned to *Bluetooth* SIG by the Promoter Members.

Use of the Specification by anyone who is not a member of *Bluetooth* SIG or a party to an Early Adopters Agreement (each such person or party, a "Member"), is prohibited. The legal rights and obligations of each Member are governed by their applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement. No license, express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to *Bluetooth* SIG or any of its members for patent, copyright and/or trademark infringement.

THE SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the *Bluetooth* technology ("*Bluetooth* products") may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of *Bluetooth* products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their *Bluetooth* Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their *Bluetooth* products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses**. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.**

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST *BLUETOOTH* SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserve the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

Document Terminology

The Bluetooth SIG has adopted Section 13.1 of the IEEE Standards Style Manual, which dictates use of the words ``shall'', ``should'', ``may'', and ``can'' in the development of documentation, as follows:

- The word shall is used to indicate mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (shall equals is required to).
- The use of the word must is deprecated and shall not be used when stating mandatory requirements; must is used only to describe unavoidable situations.
- The use of the word will is deprecated and shall not be used when stating mandatory requirements; will is only used in statements of fact.
- The word should is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required; or that (in the negative form) a certain course of action is deprecated but not prohibited (should equals is recommended that).
- The word may is used to indicate a course of action permissible within the limits of the standard (may equals is permitted).
- The word can is used for statements of possibility and capability, whether material, physical, or causal (can equals is able to)

Contents

Foreword

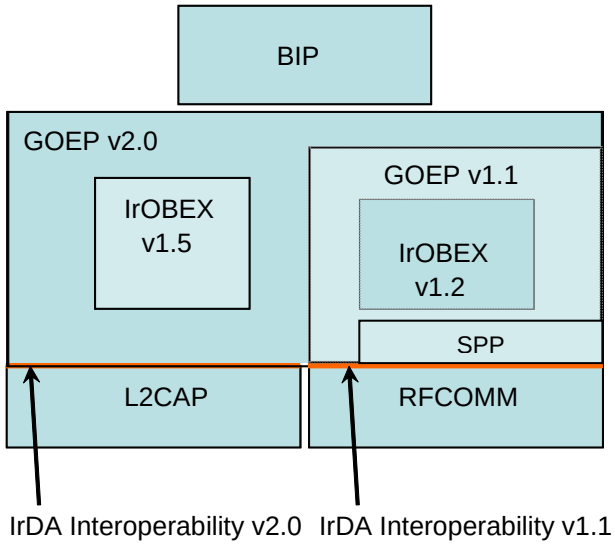
The File Transfer Profile defines a mechanism enabling Bluetooth devices to exchange files in a generic fashion. The foundation of the Basic Imaging Profile is a series of constructs that enable Bluetooth devices to negotiate the size and encoding of imaging data to be exchanged. Given the variety of image formats currently in use, it is impossible – without using a negotiation mechanism – to guarantee that images transferred via Bluetooth (even when correctly received) can actually be used by a receiving device.

Interoperability cannot be guaranteed unless all Bluetooth imaging devices support at least one common image format. Therefore, this profile requires that all Bluetooth imaging devices be capable of receiving JPEG thumbnail images and/or providing JPEG thumbnail versions of their stored images (either by performing format conversions or by using a thumbnail image created when the image is stored). Built on this negotiation mechanism and required imaging format are six features, ranging from very basic image transfer to remote control operations that enable the usage scenarios described in the Bluetooth Imaging Market Requirements Document.

1 Introduction

1.1 Profile Dependencies

In the figure below, 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 or indirectly. For example, the Basic Imaging profile is dependent on the Generic Object Exchange, Serial Port (for compatibility with devices implementing earlier versions of this profile), and Generic Access profiles (GOEP, SPP and GAP respectively).



1.2 Bluetooth OBEX-Related Specifications

This specification is built upon the following specifications:

1. Bluetooth IrDA Interoperability Specification v2.0 [8].
   - Defines how the applications can function over both Bluetooth and IrDA.
   - Specifies how OBEX is mapped over L2CAP and TCP.
2. Bluetooth Generic Object Exchange Profile Specification v2.0 [7]
   - Generic interoperability specification for the application profiles using OBEX over L2CAP.
   - Defines the interoperability requirements of the lower protocol layers (e.g. Baseband and LMP) for the application profiles.

1.3 Symbols and Conventions

1.3.1 Requirement Status Symbols

In this document (especially in the features and functions tables in Section 4.3), the following symbols are used:

"M" for mandatory to support (used for capabilities that shall be used in the profile)

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

"C" for conditional support (used for capabilities that shall be used in case a certain other capability is supported)

"X" for excluded (used for capabilities that may be supported by the device 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 to support. These are capabilities that may, however, degrade the operation of devices following this profile. Therefore these features shall not be activated in the context of an OBEX connection within this profile.

In this specification, the word ‘shall’ or ‘must’ is used for mandatory requirements, the word ‘should’ is used to express recommendations and the word ‘may’ is used for options.

The conformance requirements of a profile relate to the conformance requirements in the base standards in the following ways:

- Unconditional mandatory requirements in a base standard shall remain mandatory in the profile. (Base standard requirements to support a certain procedure do not mean it has to be used in a profile.)
- Unconditional options in a base standard may remain options or may be changed within the profile to become: mandatory, conditional, out of scope (unused), or prohibited.
- If conditional requirements in a base standard can be fully evaluated in the context of the profile, they become unconditional mandatory, unconditional options, out of scope, or prohibited. Otherwise, conditional requirements remain conditional, possibly with evaluated conditions.

1.3.2 Signaling Diagram Conventions

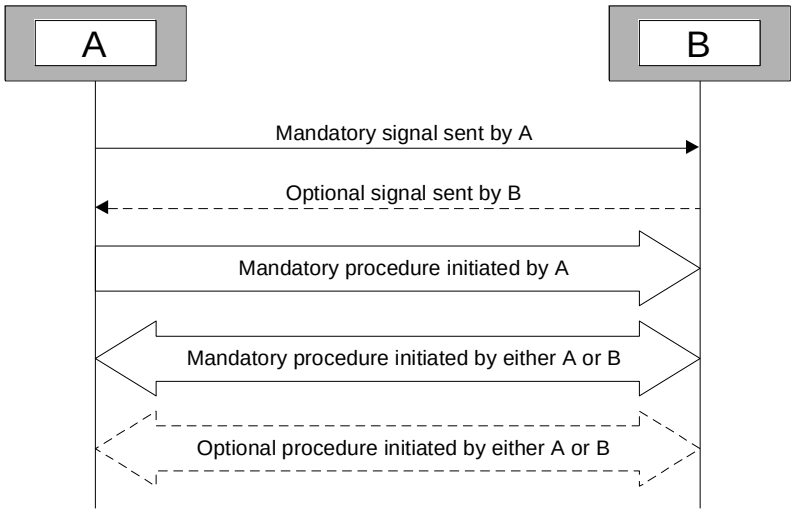


Figure 1.1: Signaling Diagram Conventions

In Figure 1.1, the following cases are shown:

A thin arrow represents a signal or individual message. When dotted, the representation indicates that the signal or message is optional. The arrow points from the issuer of the signal/message to its destination.

A procedure (i.e. a set of messages) is represented with a wide arrow that points from the initiator of the procedure toward the responder. If both sides can initiate the procedure, the wide arrow is double-headed. A dotted representation indicates that the procedure is optional.

2 Profile Overview

2.1 Protocol Stack

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

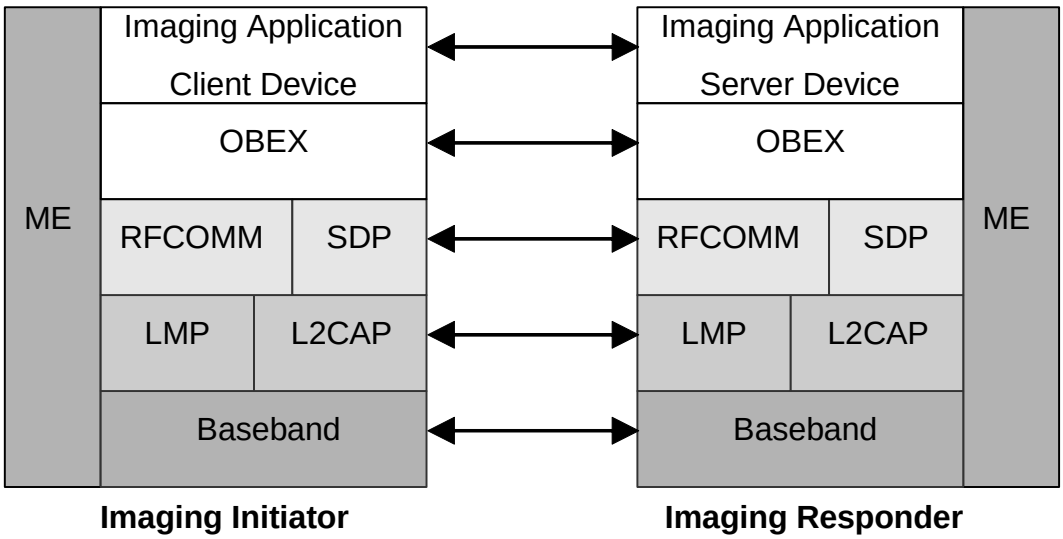


Figure 2.1: Protocol Model

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

ME is the Management Entity which coordinates procedures during initialization and manages the state of the link.

This profile requires backwards compatibility with the previous versions of BIP which are based on GOEP v1.1 [5]. The procedures for backward compatibility defined in Section 5.2 of GOEP v2.0 [7] shall be used.

2.2 Configurations and Roles

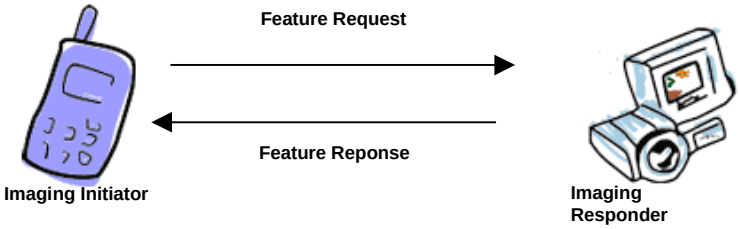


Figure 2.2: The Imaging Initiator is the Initiator of a Basic Imaging Feature and the Imaging Responder is the Responder

The following roles are defined for this profile:

Imaging Initiator: The Imaging Initiator is the device that initiates a Basic Imaging feature. The Imaging Initiator shall provide at least an object exchange client. It shall comply with the GOEP interoperability requirements for the client. To support some features, the Imaging Initiator shall also provide an object exchange server and comply with the interoperability requirements for the server. Features that require support for both the object exchange client and object exchange server on the Imaging Initiator are indicated in Section 4.3.

Imaging Responder: The Imaging Responder is the device that responds to the initiation of a Basic Imaging feature by the Imaging Initiator. The Imaging Responder shall provide at least an object exchange server. The Imaging Responder shall comply with the GOEP interoperability requirements for the server. To support some features, the Imaging Responder shall also provide an object exchange client and comply with the interoperability requirements for the client. Features that require support for both the object exchange client and object exchange server on the Imaging Responder are indicated in Section 4.3.

There is no mandatory mapping between the Imaging Initiator/Imaging Responder roles and BR/EDR master/slave roles.

2.3 User Requirements and Scenarios

The scenarios covered by this profile are:

- Use of a mobile phone to send one or more images to a wristwatch. Scenarios involving imaging devices of different natures but forming a functionally equivalent combination are also enabled by this profile.
- Use of a mobile phone to browse and retrieve the images stored on a digital still camera. These images may then be stored locally or forwarded to a third party via a PLMN network. Scenarios involving imaging devices of different natures but forming a functionally equivalent combination are also enabled by this profile.
- Use of a PC to automatically download the content of a digital still camera as soon as the camera comes into the vicinity of the PC. Scenarios involving imaging devices of different natures but forming a functionally equivalent combination are also enabled by this profile.

- Use of a printer to print one or more images sent from a digital still camera. Scenarios involving imaging source devices of a different nature but with equivalent functionally could also be implemented.
- Use of a mobile phone to control the shutter of a digital still camera and immediately examine the result on the phone's screen. In the present scenario, any other portable imaging device could play the role of the mobile phone.
- Use of a laptop computer to send and control the display of images by a projector device. Scenarios involving imaging source devices of a different nature but equivalent functionality can also be implemented. The role of the projector could potentially be played by any device with display capability.

2.4 Profile Fundamentals

The profile fundamentals are the same as those defined in the GOEP.

LMP authentication and encryption are mandatory to support and optional to use. When using Core Specification v2.1 or later, encryption is mandatory to use; otherwise optional.

Bonding is mandatory to support and optional to use. This profile does not mandate that either the server or the client enter the discoverable mode or the connectable mode automatically even if they are able to do so.

User interaction is always needed to trigger one of the six features defined in Section 3.2, where "user interaction" includes both on-the-fly interaction and use of pre-configured settings.

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 to 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 User Interface Aspects

3.1 Mode Selection

There is a mode associated with the Basic Imaging Profile. It affects how the Imaging Responder is configured.

Bluetooth Imaging Mode enables the Imaging Responder to maintain a connection and receive commands from Imaging Initiators. As a result of entering Bluetooth Imaging Mode:

- Devices that wish to be discoverable for a limited period of time shall be in limited discoverable mode and in connectable mode. It is expected that most implementations will take this approach.
- Devices that wish to be discoverable all the time shall be in general discoverable mode and in connectable mode.
- Devices that wish to be hidden from devices they never talked to before shall be in non-discoverable mode and in connectable mode.

The setting of the CoD bits – together with the registration of all the local imaging applications in the SDP database of the Imaging Responder – may also be influenced by entry into Bluetooth Imaging Mode. Note that the publication of the present profile is accompanied by the introduction of an Imaging specific Major Device Class and associated Minor Device Class values in the Assigned Numbers specification [9].

It is recommended that Bluetooth Imaging Mode be entered and exited via user interaction, but depending on the nature of the Imaging Responder and on the application it is running, this mode may be entered automatically.

On the inquiring side, limited inquiry as described in [1] is the preferred inquiry procedure.

3.2 Features

There are six features associated with the Basic Imaging Profile:

- Image Push
- Image Pull
- Advanced Image Printing
- Automatic Archive
- Remote Camera
- Remote Display

The Image Push feature pushes one or more images to an Imaging Responder device. As a result of being pushed, an image could be displayed, stored, buffered, or printed.

The Image Pull feature browses through the images stored on the Imaging Responder device and downloads one or more of them as requested by the user.

The Advanced Image Printing feature is designed for the usage case where the Imaging Responder is a printer or printing-enabled device. This feature enables an Imaging Initiator device to specify print job options and produces richer output than that obtained by using the Image Push feature.

The Automatic Archive feature triggers the Imaging Responder device to download from the Imaging Initiator device all or part of the images stored on that device. It is up to the Imaging Responder to determine which images to download; the Automatic Archive feature could, for instance, be combined with a synchronization application that would retrieve only new images from the Imaging Initiator device.

The Remote Camera feature allows the user of the Imaging Initiator device to view thumbnail size monitoring images as seen by the Imaging Responder device and trigger the shutter of the Imaging Responder when desired. The Imaging Responder is a device with image capture capability, such as a digital still camera.

The Remote Display feature allows the user to push images to an Imaging Responder with display capability and control the display sequence of those images.

These basic imaging features should be triggered by the user. They should not generally be performed automatically unless such action is the result of a configuration setting that is under user control.

3.3 Example Feature Sequences

The sequences presented in this section are examples. Variations in implementation are possible and allowed.

In the following sequences, with the exception of Automatic Archive, pairing can be performed as necessary and is left to the implementer's discretion. All devices shall support pairing as defined in the GAP. In the case of Automatic Archive, it is highly recommended that pairing be a prerequisite to the use of that feature.

Note that the list of Imaging Responders that is displayed on the Imaging Initiator can be the result of a simple inquiry. It can also result from an inquiry combined with a name request and/or SDP requests, or it can consist of a list of previously paired devices that the Imaging Initiator maintains internally.

3.3.1 Example Image Push Sequence

When an Imaging Initiator wants to push an image to an Imaging Responder the following sequence can occur:

Imaging Initiator	Imaging Responder
	The user sets the device into Bluetooth Imaging Mode.
The user of the Imaging Initiator selects the Image Push feature on the device.	
A list of devices is displayed to the user.	

Imaging Initiator	Imaging Responder
The user selects a device to push the image to. If the selected device does not support the Basic Imaging Profile, the user is prompted to select another device.	
The user selects an image and sends it.	
	When the image is received by the Imaging Responder, the user is asked to accept or reject the image.
The user is notified of the result of the operation.	

3.3.2 Example Image Pull Sequence

When an Imaging Initiator wants to browse the images stored on an Imaging Responder, the following sequence can occur:

Imaging Initiator	Imaging Responder
	The user sets the device into Bluetooth Imaging Mode.
The user of the Initiator selects the Image Pull feature on the device.	
A list of devices is displayed to the user.	
The user selects a device to pull images from. If the selected device does not support the Basic Imaging Profile, the user is prompted to select another device.	
The list of stored pictures is displayed to the user.	
The user can at any moment decide to retrieve an image or a collection of images he or she is interested in.	
The user is notified of the result of the operation.	

3.3.3 Example Advanced Image Printing Sequence

When an Imaging Initiator wants to print an image, the following sequence can occur:

Imaging Initiator	Imaging Responder
	The user sets the device into Bluetooth Imaging Mode.
The user of the Imaging Initiator selects the Advanced Image Printing feature on the device.	
A list of printers is displayed to the user.	
The user selects a printer to push the image to. If the selected printer does not support the Basic Imaging Profile, the user is prompted to select another printer.	
The user selects an image or a collection of images and sends it. The user specifies, for each individual image, the number of copies to produce, and (if necessary) a date stamp, a title stamp, and a frame number stamp.	
	The printer prints the image(s).
The user is notified of the result of the operation.	

3.3.4 Example Automatic Archive Sequence

When an Imaging Initiator implements the Automatic Archive feature, the following sequence can occur:

Imaging Initiator	Imaging Responder
	The user configures the Imaging Responder for Bluetooth Imaging Mode (for example, through the configuration panel of PC operating systems).
The user of the Imaging Initiator selects the Automatic Archive feature on the device. The Imaging Initiator connects automatically to the Imaging Responder (the devices have been previously paired).	
	The Imaging Responder device downloads images from the Imaging Initiator device.
The user is notified of the result of the operation.	

By providing the means to identify new and modified images on the Imaging Initiator, the Automatic Archive feature provides the necessary tools to allow for intelligence in the Imaging Responder's download application so that only the necessary images are retrieved.

This feature can be potentially sensitive from a security point of view. Implementers should ensure that the level of security they chose to provide is appropriate.

3.3.5 Example Remote Camera Sequence

When an Imaging Initiator wants to remotely control an image capture device, the following sequence can occur:

Imaging Initiator	Imaging Responder
	The user sets the device into Bluetooth Imaging Mode.
The user of the Imaging Initiator selects the Remote Camera feature on the device.	
A list of devices is displayed to the user.	
The user selects a device to control. If the selected device does not support the Basic Imaging Profile, the user is prompted to select another device.	
The current view is displayed to the user.	
The user triggers the shutter of the Imaging Responder via the user interface of the Imaging Initiator.	
	The image is captured and stored.
The resulting image is displayed to the user.	

3.3.6 Example Remote Display Sequence

When an Imaging Initiator wants to remotely control an image display device, the following sequence can occur:

Imaging Initiator	Imaging Responder
	The user sets the device into Bluetooth Imaging Mode.
The user of the Imaging Initiator selects the Remote Display feature on the device.	
A list of devices is displayed to the user.	
The user selects a device to send the images to and have them displayed. If the selected device does not support the Basic Imaging Profile, the user is prompted to select another device.	
The user pushes images to the Imaging Responder and/or sends commands to pilot the image display sequence.	
	The images are displayed as specified by the Imaging Initiator.

4 Application Layer

This section describes the feature requirements for devices compliant with the Basic Imaging Profile.

For each feature described in this chapter, an open OBEX Imaging Service session is assumed. This means that all features can be invoked only after a successful connection request/response to the Imaging Service of an Imaging Responder. Whether to close the Imaging Service session after each feature invocation or leave it open for future feature invocations is left to the implementer's discretion.

Note that for some features it is necessary to establish a secondary OBEX connection; for example, the Imaging Responder opens a Referenced Objects connection by issuing a connection request to the Imaging Initiator (these procedures are detailed in Section 5). In such a case, the secondary connection shall be terminated before any other feature can be invoked on the primary OBEX connection.

4.1 Imaging Devices Classification

The requirements imposed on devices by the Basic Imaging Profile depend on the device's declared capabilities. The one common requirement imposed on all devices implementing the Basic Imaging Profile is the ability to exchange imaging data. There are three optional capabilities that enable a device to remotely control another device or to be remotely controlled itself:

The Basic Imaging Profile common capability:

- Generic Imaging: The ability to exchange imaging data.

The optional remote control capabilities:

- Bluetooth Controlled Capturing: The capture of an image, using an optical system, via a Bluetooth interface. (A digital still camera that does not have a shutter that can be triggered via a Bluetooth interface is not considered a controlled capture device.)
- Bluetooth Controlled Printing: The printing of images via a Bluetooth interface that is also used to transfer images.
- Bluetooth Controlled Display: The display of images, controlled via a Bluetooth interface that is also used to transfer the images.

In all four capabilities, imaging data is transferred between a device that acts as source and a device that acts as destination; the role (source or destination) is inconsequential when classifying a device according to these capabilities.

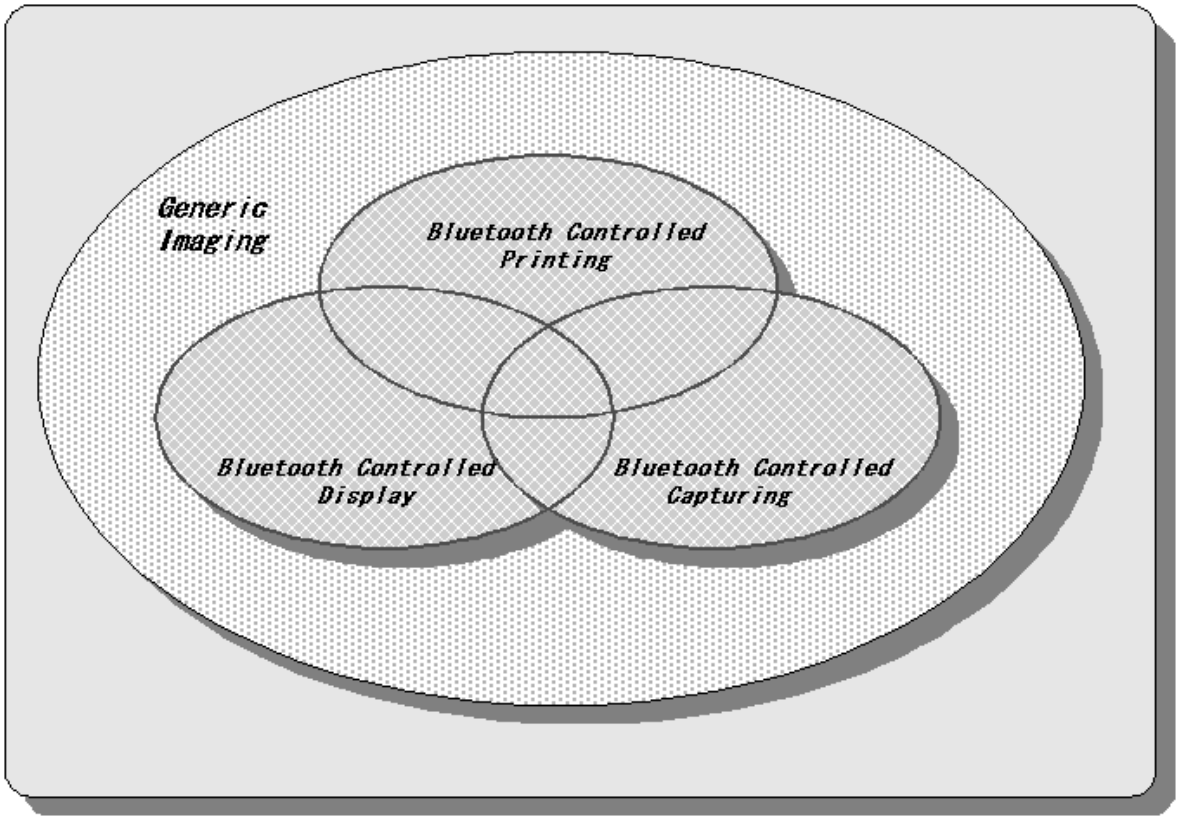


Figure 4.1: Imaging Products Classification

4.2 Imaging Features Overview

This section lists the features provided by this profile and the required degree of support classified by imaging capability.

Feature	Imaging Capability			
	Generic Imaging	**Bluetooth Controlled Printing**	**Bluetooth Controlled Capturing**	**Bluetooth Controlled Display**
Image Push	C1	M	O	O
Image Pull	C1	O	O	O
Automatic Archive	O	O	O	O
Advanced Image Printing	O	M	O	O
Remote Camera	O	O	M	O
Remote Display	O	O	O	M

C1 indicates that at least one of the features shall be supported.

Table 4.1: Imaging Features Overview

Consequently, for a device to qualify as a Basic Imaging Profile-enabled device, it shall at least be able to act as Image Push Client or Image Pull Client or Image Push Server or Image Pull Server.

4.3 Imaging Features

The tables in this section describe the functions associated with each feature and the level of support required for each function. There are two tables per feature, one corresponding to the requirements applicable when the feature is supported by an Imaging Initiator device, and one corresponding to the requirements applicable when the feature is supported by an Imaging Responder device.

Each table lists the functions that comprise a given feature; the columns indicate the level of support required for each function. When a feature requires only a primary OBEX connection (see Section 5), the Imaging Initiator needs to support only the client portion of the OBEX Imaging Service and the Imaging Responder needs to support only the server portion of the OBEX Imaging Service. When a feature requires both primary and secondary OBEX connections, some functions shall be supported by both client and server portions of the OBEX Imaging Service.

Note that a device compliant with this specification shall not use a function in the context of a given feature if that function is not associated with the feature in this section. An undefined function request shall be rejected with a “Bad Request” error code (see Section 5.3).

4.3.1 Image Push Feature

This feature enables an Imaging Initiator to send one or more images to an Imaging Responder.

This feature is comprised of four functions:

Functions as Imaging Initiator	**Function**	**OBEX Imaging Service Client**
	GetCapabilities	O
	PutImage	M
	PutLinkedThumbnail	M
	PutLinkedAttachment	O
Functions as Imaging Responder	**Function**	**OBEX Imaging Service Server**
	GetCapabilities	M
	PutImage	M
	PutLinkedThumbnail	O
	PutLinkedAttachment	O

Table 4.2: Function Overview for Image Push

The PutImage function pushes an image to an Imaging Responder.

When the PutImage function is used to push an image to a device where printing is the only possible outcome for an exchange of imaging data, the PutImage function results in the image being printed. If the Imaging Responder has multiple services capable of receiving images, the channel used to open the OBEX connection determines the outcome of the image exchange. See Section 6 for details.

It is always possible to send a thumbnail (see Section 4.4.2) to an imaging device that supports the Image Push feature as an Imaging Responder.

Although optional to support on an Imaging Initiator device, it is highly recommended that the Imaging Initiator use the GetCapabilities function prior to making an attempt to

push images to an Imaging Responder. The imaging-capabilities object retrieved by the GetCapabilities function describes – among other attributes – the Imaging Responder's level of support for image encodings and sizes. Ignoring this imaging-capabilities object may result in attempts to push images in formats unacceptable to the Imaging Responder.

PutLinkedThumbnail is used to push the thumbnail version of an image that was pushed to the Imaging Responder at an earlier time during the current connection; this action is taken in response to a request from the Imaging Responder when it replies to a PutImage operation. A PutLinkedThumbnail operation is always immediately preceded by a PutImage operation.

PutLinkedAttachment is used to push an attachment associated with an image; PutLinkedAttachment is always preceded by a PutImage operation.

A typical function sequence for the Image Push feature is illustrated below.

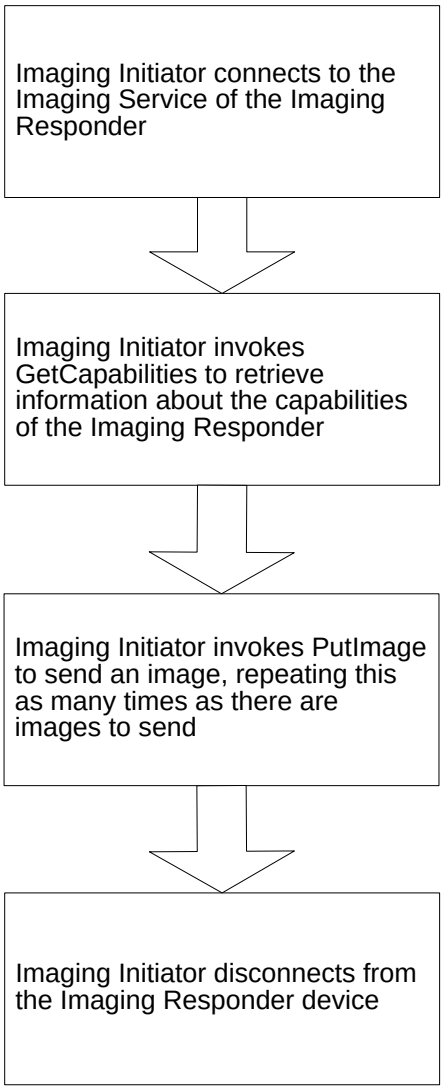


Figure 4.2: Typical Image Push Sequence

4.3.2 Image Pull Feature

This feature enables an Imaging Initiator device to browse the images available on an Imaging Responder and retrieve those of interest.

The Image Pull feature is comprised of seven functions:

Functions as Imaging Initiator	**Function**	**OBEX Imaging Service Client**
	GetCapabilities	O
	GetImagesList	M
	GetImageProperties	O
	GetImage	C1
	GetLinkedThumbnail	C1
	GetLinkedAttachment	O
	DeleteImage	O
Functions as Imaging Responder	**Function**	**OBEX Imaging Service Server**
	GetCapabilities	M
	GetImagesList	M
	GetImageProperties	M
	GetImage	M
	GetLinkedThumbnail	M
	GetLinkedAttachment	O
	DeleteImage	O

Table 4.3: Function Overview for Image Pull

C1 indicates that at least one of those functions shall be supported.

GetCapabilities enables an Imaging Initiator to discover an Imaging Responder's level of support for various imaging capabilities.

GetImagesList returns a list of handles for the images available on the Imaging Responder, together with file information such as creation date and modification date.

GetImageProperties is used to retrieve information regarding the image formats, encodings, etc. available for an image.

GetImage enables an Imaging Initiator to retrieve an image from an Imaging Responder with a specified format, encoding, etc.

GetLinkedThumbnail enables an Imaging Initiator to retrieve the thumbnail version of an image. It is also possible to retrieve a thumbnail using the GetImage function by specifying the thumbnail format – GetLinkedThumbnail is essentially a shortcut useful on very limited devices that deal only with thumbnail images.

GetLinkedAttachment enables an Imaging Initiator to retrieve an attachment associated with an image.

DeleteImage is used to delete an image from the Imaging Responder. It is recommended that the Imaging Responder also delete any attachments associated with the image, but this is left to the implementer's discretion.

A typical function sequence for the Image Pull feature is illustrated below.

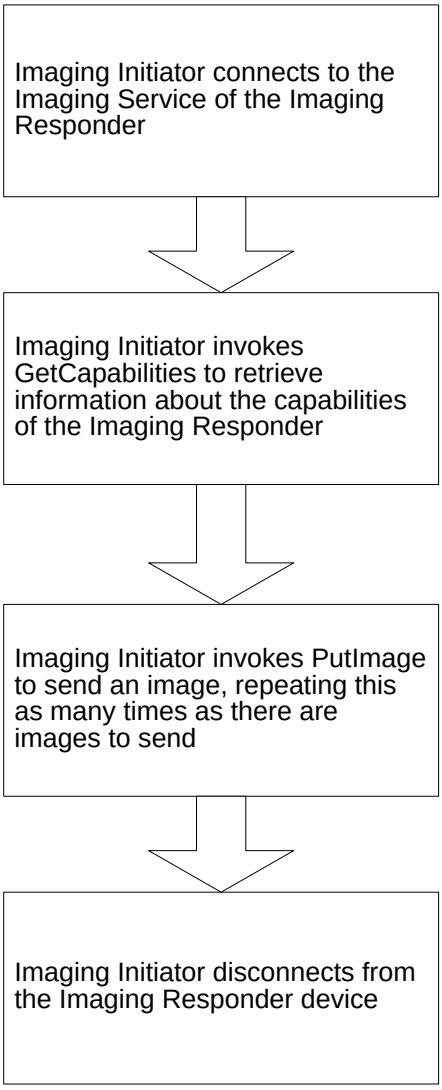


Figure 4.3: Typical Image Pull Sequence

4.3.3 Advanced Image Printing Feature

This feature provides enhanced capabilities for image printing. As described in Section 4.3.1, it is possible to print images using the Image Push feature. Image Push, however, does not provide a mechanism to specify output parameters such as the number of copies to print or how to print several images on a single page. The Advanced Image Printing feature provides these more advanced features.

The Advanced Image Print feature is comprised of four functions:

Functions as Imaging Initiator	Function	OBEX Imaging Service Client	OBEX Referenced Objects Service Server
	GetCapabilities (C)	O	N/A
	GetPartialImage (S)	N/A	M
	StartPrint (C)	M	X

	GetStatus (C)	O	X
Functions as Imaging Responder	**Function**	**OBEX Imaging Service Server**	**OBEX Referenced Objects Service Client**
	GetCapabilities (S)	M	N/A
	GetPartialImage (C)	N/A	M
	StartPrint (S)	M	X
	GetStatus (S)	M	X

Table 4.4: Function Overview for Advanced Image Printing

X indicates that the corresponding role is prohibited.

N/A indicates that the corresponding role is not applicable.

(C) indicates that the function is to be implemented as Client.

(S) indicates that the function is to be implemented as Server.

GetCapabilities enables an Imaging Initiator to retrieve the printing capabilities of an Imaging Responder.

StartPrint enables an Imaging Initiator to trigger a print job. The Imaging Responder then opens a separate OBEX connection to the Referenced Objects Service of the Imaging Initiator.

GetPartialImage is used by the Imaging Responder to retrieve the images that are necessary to perform the print job requested by the Imaging Initiator. It is an enhanced version of the GetImage function, specially designed for devices with print capability but with very little or no buffer capacity.

A typical function sequence for the Advanced Imaging Printing feature is illustrated below.

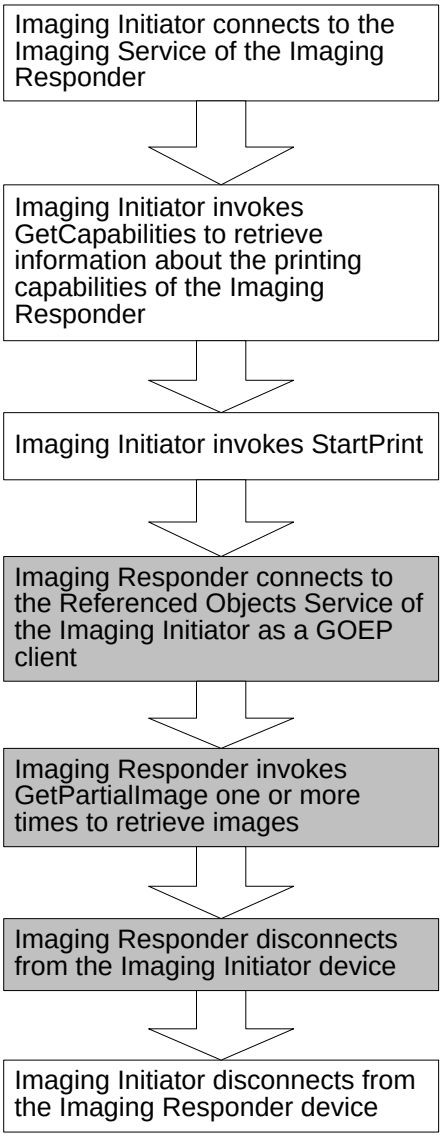


Figure 4.4: Typical Advanced Image Printing Sequence

In Figure 4.4, blocks in white describe actions by the Imaging Initiator device with respect to the connection it establishes to the Imaging Responder's Imaging Service. Blocks in grey describe actions by the Imaging Responder device with respect to the connection it establishes to the Imaging Initiator's Referenced Objects Service.

4.3.4 Automatic Archive Feature

The Automatic Archive feature enables an Imaging Initiator to request that an Imaging Responder with sufficient storage capacity retrieve all or part of its stored images. The typical usage case is that of a digital still camera that requests that a PC drain its recently captured images.

The Automatic Archive feature is comprised of nine functions:

Functions as Imaging Initiator	**Function**	**OBEX Imaging Service Client**	**OBEX Automatic Archive Service Server**
	GetCapabilities (S)	N/A	M
	GetImagesList (S)	N/A	M
	GetImageProperties (S)	N/A	M
	GetImage (S)	N/A	M
	GetLinkedThumbnail (S)	N/A	M
	GetLinkedAttachment (S)	N/A	O
	DeleteImage (S)	N/A	O
	StartArchive (C)	M	X
	GetStatus (C)	O	X
Functions as Imaging Responder	**Function**	**OBEX Imaging Service Server**	**OBEX Automatic Archive Service Client**
	GetCapabilities (C)	N/A	O
	GetImagesList (C)	N/A	M
	GetImageProperties (C)	N/A	O
	GetImage (C)	N/A	C1
	GetLinkedThumbnail (C)	N/A	C1
	GetLinkedAttachment (C)	N/A	O
	DeleteImage (C)	N/A	O
	StartArchive (S)	M	X
	GetStatus (S)	M	X

Table 4.5: Function Overview for Automatic Archive

X indicates that the corresponding role is prohibited.

C1 indicates that at least one of those functions shall be supported.

N/A indicates that the corresponding role is not applicable.

(C) indicates that the function is to be implemented as Client.

(S) indicates that the function is to be implemented as Server.

GetCapabilities enables an Imaging Responder to discover an Imaging Initiator's level of support for various imaging capabilities.

GetImagesList returns a list of handles for the images available on the Imaging Initiator, together with file information such as creation date and modification date.

GetImageProperties is used to retrieve information regarding the formats, encodings, etc. available for an image.

GetImage enables an Imaging Responder to retrieve an image from an Imaging Initiator with a specified format, encoding, etc.

GetLinkedThumbnail enables an Imaging Responder to retrieve the thumbnail version of an image.

GetLinkedAttachment enables an Imaging Responder to retrieve an attachment associated with an image.

DeleteImage is used to delete an image from the Imaging Initiator. It is recommended that the Imaging Initiator also delete any attachments associated with the image, but this is left to the implementer's discretion.

StartArchive is used by the Imaging Initiator to request that the Imaging Responder begin the archiving process.

A typical function sequence for the Automatic Archive feature is illustrated in Figure 4.5.

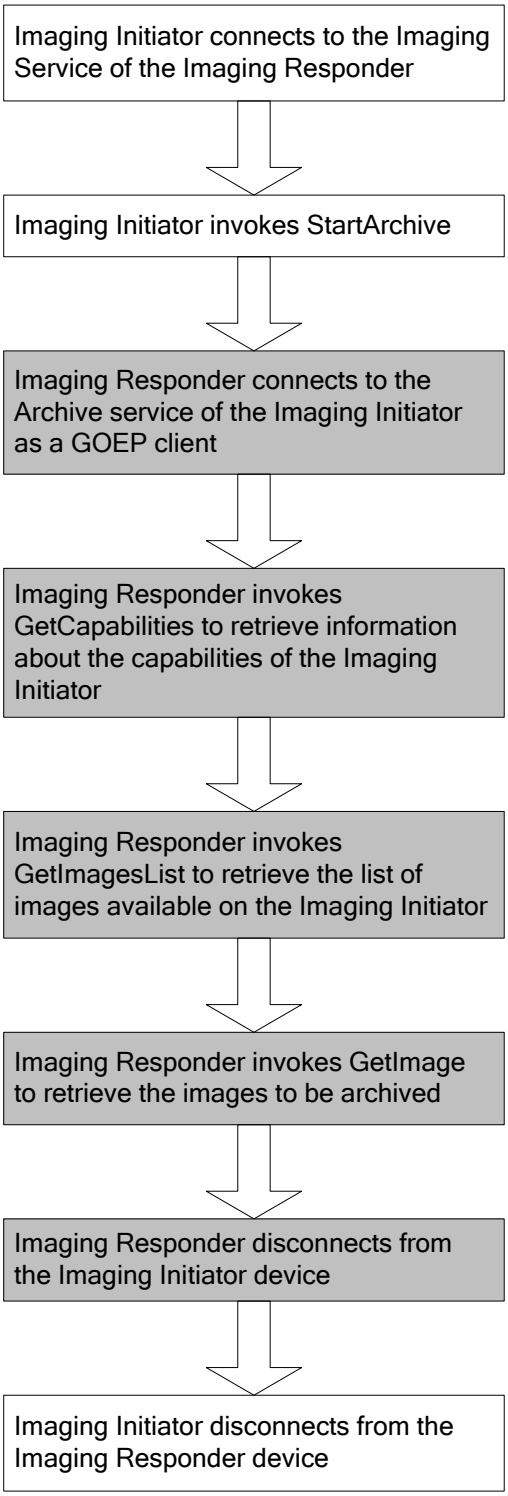


Figure 4.5: Typical Automatic Archive Sequence

4.3.5 Remote Camera Feature

This feature enables a user to view monitoring images as seen by an image capture device, remotely trigger the shutter of the device, and have the device store the resulting image.

The Remote Camera feature is composed of four functions:

Functions as Imaging Initiator	**Function**	**OBEX Imaging Service Client**
	GetMonitoringImage	M
	GetImageProperties	O
	GetImage	O
	GetLinkedThumbnail	O
Functions as Imaging Responder	**Function**	**OBEX Imaging Service Server**
	GetMonitoringImage	M
	GetImageProperties	M
	GetImage	M
	GetLinkedThumbnail	M

Table 4.6: Function Overview for Remote Camera

GetMonitoringImage is used to retrieve the monitoring image object from an Imaging Responder device with image capture capability. The Imaging Initiator can also indicate whether the Imaging Responder should permanently store the image corresponding to this monitoring image.

GetImageProperties is used to retrieve information regarding the image formats, encodings, etc. available for an image.

GetImage enables an Imaging Initiator to retrieve an image from an Imaging Responder with a specified format, encoding, etc.

GetLinkedThumbnail enables an Imaging Initiator to retrieve the thumbnail version of an image. It is also possible to retrieve a thumbnail using the GetImage function by specifying the thumbnail format – GetLinkedThumbnail is essentially a shortcut useful on very limited devices that deal only with thumbnail images.

A typical function sequence for the Remote Camera feature is illustrated in Figure 4.6.

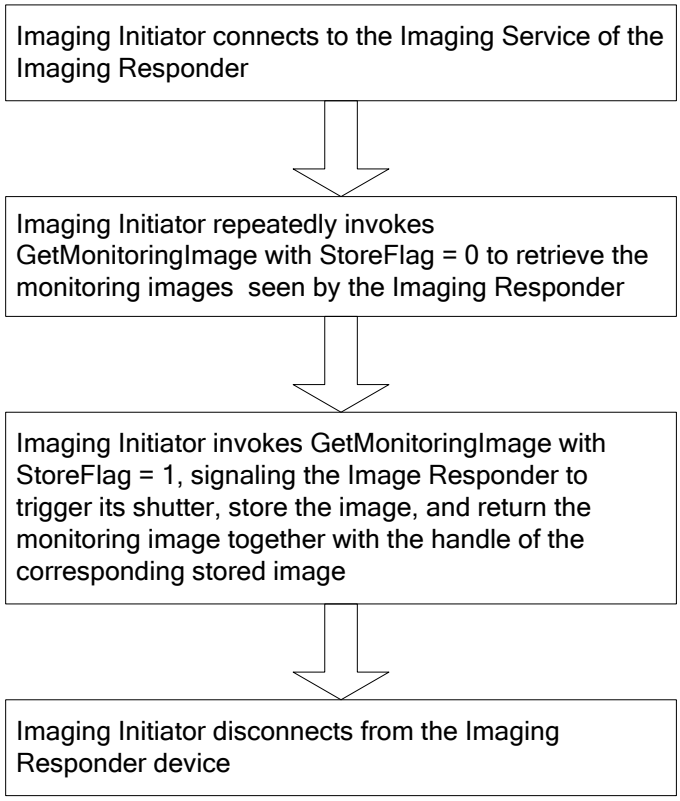


Figure 4.6: Typical Remote Camera Sequence

4.3.6 Remote Display Feature

This feature enables an Imaging Initiator to push images to a display device and control the sequence used to display those images.

The Remote Display feature comprises five functions:

Functions as Imaging Initiator	Function	OBEX Imaging Service Client
	GetCapabilities	O
	PutImage	M
	PutLinkedThumbnail	M
	GetImagesList	O
	RemoteDisplay	M
Functions as Imaging Responder	**Function**	**OBEX Imaging Service Server**
	GetCapabilities	M
	PutImage	M
	PutLinkedThumbnail	O
	GetImagesList	M
	RemoteDisplay	M

Table 4.7: Function Overview for Remote Display

The PutImage function pushes an image to the Imaging Responder. The Imaging Responder should not immediately display the received image, but instead should wait for the Imaging Initiator to invoke the RemoteDisplay function.

Although optional to support on the Imaging Initiator device, it is highly recommended that the Imaging Initiator use the GetCapabilities function prior to making an attempt to push images to an Imaging Responder. The imaging-capabilities object retrieved by the GetCapabilities function describes – among other attributes – the Imaging Responder's level of support for image encodings and sizes. Ignoring this imaging-capabilities object may result in attempts to push images in formats unacceptable to the Imaging Responder.

PutLinkedThumbnail is used by the Imaging Initiator to send the thumbnail version of an image to the Imaging Responder.

GetImagesList returns a list of handles for the images available for display on the Imaging Responder, together with file information such as creation date and modification date. It is recommended that the Imaging Initiator keep track of the handles returned as a result of each PutImage operation so the command to display selected images can be used immediately. Devices that do not wish to store the handles can use the GetImagesList function to retrieve the list of handles.

The RemoteDisplay function is used to control the display of images on the Imaging Responder. This command can be a simple "go to the next image" or "go to the previous image"; the command can also select a specific image.

It is always possible to send images as thumbnails to an imaging device that supports the Remote Display feature as an Imaging Responder.

A typical function sequence for the Remote Display feature is illustrated in Figure 4.7.

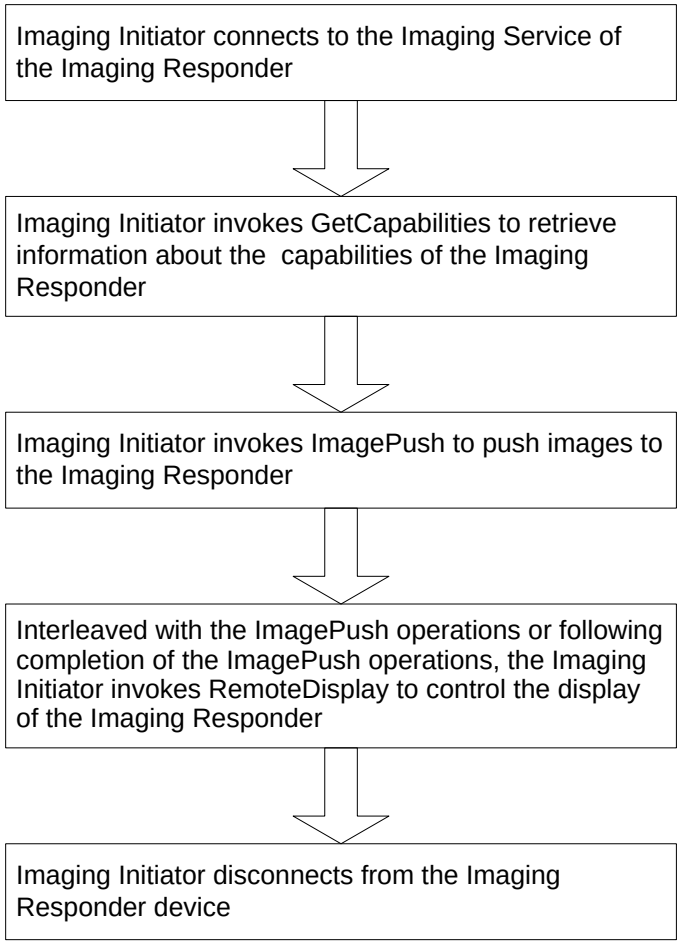


Figure 4.7: Typical Remote Display Sequence

4.4 Imaging Profile Formats, Objects, and Parameters

4.4.1 Storage Formats Support

The present profile does not mandate support for any particular imaging data storage format.

4.4.2 Imaging File Formats Support

The present profile does not mandate support for any particular imaging file format for local storage. The method used to store images in a Bluetooth imaging device is left to the implementer's discretion.

The only requirement, in terms of file formats, for an imaging device to be compliant with the Basic Imaging Profile is that it shall be able to produce a thumbnail (called an imaging thumbnail) for each and every image that it exposes to other devices. This imaging thumbnail may be a result of a conversion performed locally or it may have been created when the associated full-size image was captured.

Other image encodings, namely GIF, PNG, BMP, WBMP, and JPEG2000, are also supported in this profile. There is, however, no interoperability guarantee:

- Interoperability presupposes that both parties (Initiator and Responder) support the same encoding, while the only encoding mandated by this profile is the imaging thumbnail.
- These other encodings have many variants using different encoding parameters and options. Since most of these encodings have one or more de facto variants that codecs are assumed to support, interoperability problems should be rare. However, there are cases – especially for JPEG2000, which has yet to be widely deployed in the imaging industry – where it's possible that an image file exchanged between two devices supporting this profile may not be correctly interpreted even though both devices indicate support for the encoding.

The specifications for JPEG, GIF, WBMP, PNG, BMP, and JPEG2000 are [12], [15], [16], [17], [17], and [18] respectively.

4.4.3 Imaging Thumbnail

The definition of a thumbnail in Bluetooth Imaging (also called an imaging thumbnail in this profile) is as follows:

- JPEG baseline-compliant
- sRGB as default colour space
- Pixel size: 160x120
- Sampling: YCC422
- One marker segment for each DHT and DQT
- Typical Huffman table
- DCF thumbnail file as file format (i.e. EXIF with the thumbnail container in APP1 empty and the imaging thumbnail as basic main image as defined in [14])

If the aspect ratio of the original image differs from 4.3, it is left to the implementer to decide which method to use to produce the thumbnail (padding, filling, cropping, etc.).

4.4.4 Imaging Handles

The Basic Imaging Profile is based on image handles. Image handles are indices (similar to pointers) that are created and assigned by a device to its locally stored images. Image handles are 7 character long strings containing only the digits 0 to 9. Handles are only required to be unique on the source device. Image handles on a device shall be valid and unique throughout the duration of a primary OBEX connection (see Section 5.5). On some implementations, handles may be valid throughout the lifespan of the imaging device, but this is not a requirement of this profile – handles can change each time a new primary OBEX connection is established.

Image handles are always transported in an OBEX User Defined header called "Img-Handle" (see Section 5).

The choice was made to express these handles as character strings rather than binary values in order to facilitate their handling by digital cameras and other devices with relatively simple file storage structures. Annex B: Implementation Guidelines for DCF Devices gives handling guidelines for DCF devices. Other devices that also have a

simple file storage structure (typically no directory nesting) may also apply the same guidelines.

4.4.5 Imaging Attachments

This profile supports the exchange of attachments linked with an image via its handle. All operations related to attachments are marked with an OBEX Type header of “x-bt/img-attachment”.

4.4.6 XML Headers and Objects

All XML headers and bodies are encoded using UTF-8.

Note that XML declarations (for example, <?xml version=”1.0”?>) that normally mark the beginning of XML documents are not to be used in the Basic Imaging Profile. Implementations that need to reuse Basic Imaging Profile XML descriptions for other purposes may have to add XML declarations when interfacing to applications or services outside the scope of the profile.

4.4.6.1 Images-Listing Object (x-bt/img-listing)

The images-listing object describes the images available on an object exchange server in the context of an OBEX connection. Object exchange clients can request images-listing objects from servers and control whether or not the returned list of image handles is ordered (see Section 4.5.6); clients can also request that servers apply a filter to the images-listing object to narrow the list’s scope (see Section 4.4.7.1). The images-listing object is associated with the OBEX Type header “x-bt/img-listing”.

Definition
The images-listing object is based on the folder-listing object defined in [4]. It describes the content of the imaging device in terms of the images that an object exchange client can retrieve in the current OBEX connection.

```
<!DOCTYPE images-listing [
<!ELEMENT images-listing ( image )* >
<!ATTLIST images-listing version CDATA #FIXED “1.0” >
<!ELEMENT image EMPTY>
<!ATTLIST image
  handle CDATA #REQUIRED
  created CDATA #IMPLIED
  modified CDATA #IMPLIED>
]>
```

The elements used in the images-listing object are defined as follows:

Element Name	Meaning
images-listing	The list of image handles available on an object exchange server in the context of the current OBEX connection.
image	Signals the existence of an image that can be referenced by its handle.

Table 4.6: Elements Used in the Images-Listing Object

The attributes used in the images-listing object are defined as follows:

Attribute Name	Meaning
version	The version of the images-listing XML string.
handle	The image's handle.
created	The image's creation time. The format is YYYYMMDDTHHMMSS, where the capital letter 'T' is explicitly inserted between the day and hour fields. It is recommended that whenever possible UTC time is used; when UTC time is used, the letter 'Z' is appended to the end of the string.
modified	The image's modification time. It uses the same format as the created attribute.

Table 4.7: Attributes used in the images-listing Object

Example images-listing XML String

The source device has three images. Image 1000001 was created on the 1st of August 2000 at 6:00 UTC. Image 1000003 was created on the 1st of August 2000 at 6:01:15 UTC and modified on the 8th of August at 07:15:00 UTC. Image 1000004 was created on the 1st of August 2000 at 6:01:37 UTC. The following images-listing object represents the content of the source device:

```
<images-listing  version="1.0" >
<image handle="1000001" created="20000801T060000Z" />
<image handle="1000003" created="20000801T060115Z"
modified="20000808T071500Z" />
<image handle="1000004" created="20000801T060137Z" />
</images-listing>
```

4.4.6.2 Image-Properties Object (x-bt/img-properties)

The image-properties object describes the details of an image, including its available sizes/encodings. The details included in the object can be tailored to the needs of an object exchange client using filtering parameters as explained in Section 4.4.7.2. The image-properties object is associated with the OBEX Type header "x-bt/img-properties".

Definition

The DTD for the image-properties object is as follows:

```
<!DOCTYPE image-properties [
<!ELEMENT image-properties ( native, variant* ,attachment* ) >
<!ATTLIST image-properties
   version CDATA #FIXED "1.0"
   handle CDATA #REQUIRED
   friendly-name CDATA #IMPLIED>

<!ELEMENT native EMPTY>
<!ATTLIST native
   encoding CDATA  #REQUIRED
   pixel CDATA #REQUIRED
   size CDATA #IMPLIED>
```

```
<!ELEMENT variant EMPTY>
<!ATTLIST variant
   encoding CDATA #REQUIRED
   pixel CDATA #REQUIRED
   maxsize CDATA #IMPLIED
   transformation NMTOKENS #IMPLIED “stretch crop fill”>

<!ELEMENT attachment EMPTY>
<!ATTLIST attachment
   content-type CDATA #REQUIRED
   charset CDATA #IMPLIED
   name CDATA #REQUIRED
   size CDATA #IMPLIED
   created CDATA #IMPLIED
   modified CDATA #IMPLIED>
]>
```

Note that the imaging thumbnail format shall be included as an available size/encoding in either the variant-image element or native-image element.

The elements used in the image-properties object are defined as follows:

Element Name	Meaning
image-properties	A detailed description of an image.
native	Describes the native formats in which the image is available.
variant	Describes the variant formats in which the image is available. A variant format is a size/encoding of an image that is produced by an object exchange server on the fly by applying relevant transformations to one or more native forms of the image.
attachment	Describes the attachments associated with an image.

Table 4.8: Elements used in the Image-Properties Object

The attributes used in the image-properties object are defined as follows:

Attribute Name	Meaning
version	The version of the image-properties XML string.
handle	The image’s handle.
friendly-name	A human-readable name for the image. This name can be the file name of the image.
encoding	An encoding method in which the image is available. The encodings supported by this profile include JPEG, GIF, WBMP, PNG, JPEG2000, BMP, and USR-xxx. The tag USR-xxx is used to represent proprietary encodings. The tag shall begin with the string “USR-” but the implementer assigns the characters of the second half of the string. This tag can be used by a manufacturer to enable its devices to exchange images under a proprietary encoding. It is recommended that the name of the manufacturer or its abbreviation be included in the tag (for instance, USR-NOKIA-FORMAT1). It is highly probable that devices manufactured by other companies would not understand this encoding.

Attribute Name	Meaning
pixel	The pixel size or the range of pixel sizes in which the image is available. A fixed pixel size is expressed using the format W(idth)*H(eight). The possible values for W(idth) and H(eight) range from 0 to 65535. If the image can be resized to allow for a modified aspect ratio, the pixel range is indicated using the format W(idth)1*H(eight)1-W(idth)2*H(eight)2, where W(idth)1*H(eight)1 indicates the smallest image size and W(idth)2*H(eight)2 gives the largest image size. 65535*65535 can be used as W(idth)2*H(eight)2 to indicate the absence of an upper limit. For image resizing with a fixed aspect ratio, the pixel range is expressed using the format W(idth)1**- W(idth)2*H(eight)2, where W1 is the smallest width possible and W2*H2 is the largest size possible. For each possible intermediate value of W(idth), the corresponding height is calculated using the formula H=(W*H2)/W2.
transformation	The list of supported image transformation methods. “stretch” indicates that the object exchange server is capable of stretching the image. “fill” indicates the object exchange server can fill the image with padding data. “crop” indicates that the object exchange server can crop the image.
content-type	Gives the MIME content type of the attachment; for example, content-type=“text/plain”. See http://www.isi.edu/in-notes/iana/assignments/media-types/ for content types.
charset	The MIME character set of the attachment; for example, charset=“iso-8859-1”.
name	The file name of the attachment, which shall be provided to the object exchange server to retrieve the attachment.
size	The size in bytes of the image file or an attachment.
maxsize	The estimated maximum size of the image after conversion from the native image. This value is an estimate because it is very difficult to compute what the size of an image’s variant encoding will be without actually performing the conversion. This attribute is useful to assess whether the variant would suit the storage limitations imposed by the device that requested the image-properties object. Although highly recommended to support, this attribute is optional.
created	The attachment’s creation time. The format is YYYYMMDDTHHMMSS, where the capital letter ‘T’ is explicitly inserted between the day and hour fields. It is recommended that whenever possible UTC time is used; when UTC time is used, the letter ‘Z’ is appended to the end of the string.
modified	The attachment’s modification time. It uses the same format as the created attribute.

Table 4.9: Attributes used in the Image-Properties Object

Example image-properties XML String

Figure 4.8 illustrates how to build an image-properties object.

Image 1000001 is available in JPEG and in GIF encodings. In JPEG, it can be delivered in 1280*1024, 640*480, or 160*120 sizes. In GIF, it can be delivered with any resolution between 80*60 and 640*480. The native format of the image is JPEG 1280*1024 and its size is 1 MB (1048576 bytes). Two attachments are available, one text and one audio; their sizes are 5 kB (5120 bytes) and 100 kB (102400 bytes), respectively.

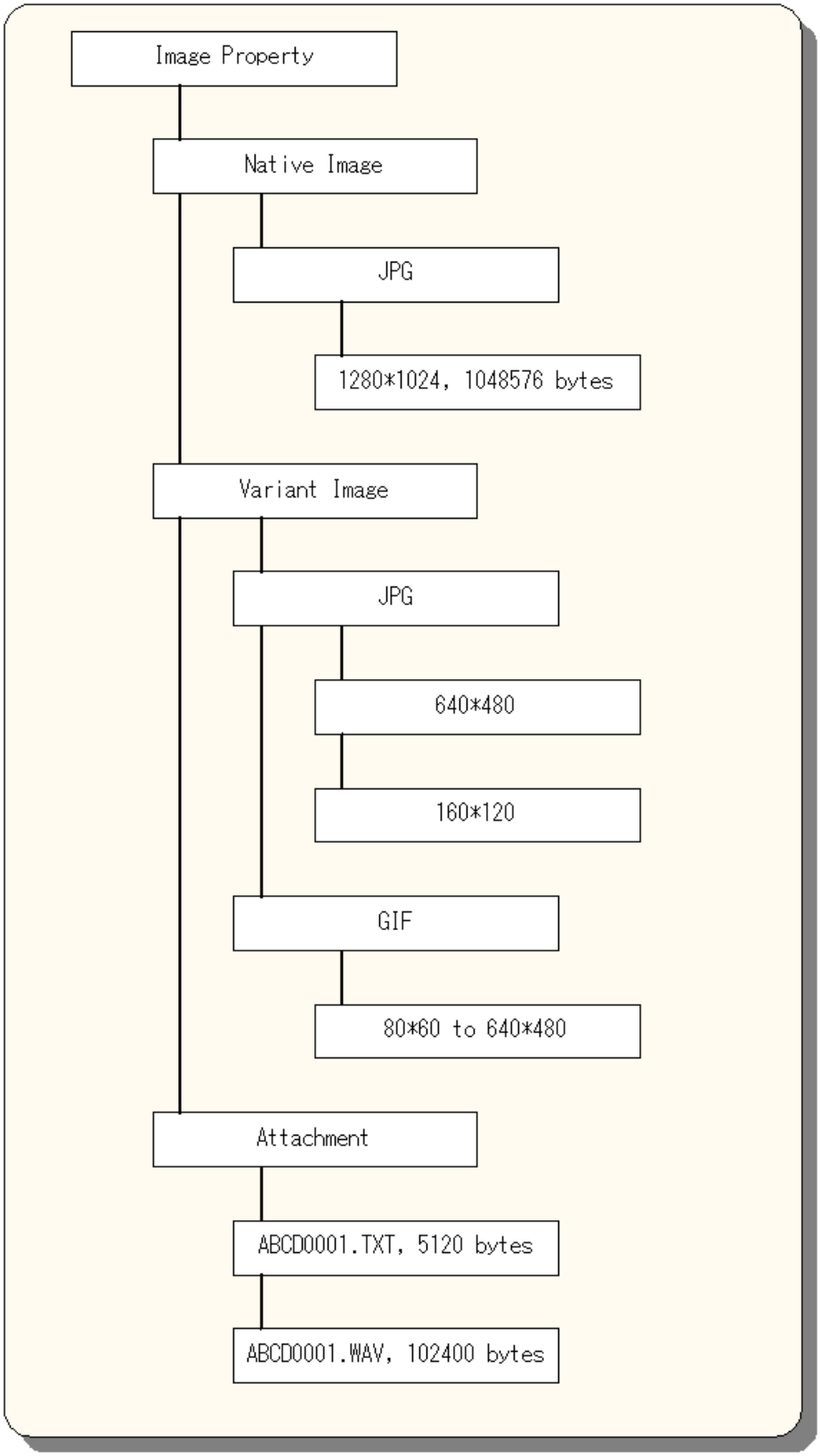


Figure 4.8: Example Image-Properties Representation

The corresponding image-properties XML string follows:

```
<image-properties version="1.0" handle="1000001">
<native encoding="JPEG" pixel="1280*1024" size="1048576"/>
<variant encoding="JPEG" pixel="640*480" />
<variant encoding="JPEG" pixel="160*120" />
<variant encoding="GIF" pixel="80*60-640*480"/>
<attachment content-type="text/plain" name="ABCD0001.txt" size="5120"/>
<attachment content-type="audio/basic" name="ABCD0001.wav" size="102400"/>
</image-properties>
```

4.4.6.3 Imaging-capabilities Object (x-bt/img-capabilities)

The imaging-capabilities object is a complement to the Imaging SDP service record. Imaging-capabilities is a mandatory object that describes in more detail the various options, formats, and attributes that are supported by a device. The imaging-capabilities object is associated with the OBEX Type header "x-bt/img-capabilities".

Definition

The DTD for the imaging-capabilities object is as follows:

```
<DOCTYPE! Imaging-capabilities [

<!ELEMENT imaging-capabilities ( preferred-format? ,image-formats* ,attachment-
formats*, filtering-parameters?, DPOF-options?  ) >
<!ATTLIST imaging-capabilities
   version CDATA #FIXED "1.0" >

<!ELEMENT preferred-format EMPTY>
<!ATTLIST  preferred-format
   encoding CDATA  #REQUIRED
   pixel CDATA #IMPLIED
   transformation NMTOKENS  #IMPLIED "stretch crop fill"
   maxsize CDATA #IMPLIED>

<!ELEMENT image-formats EMPTY>
<!ATTLIST  image-formats
   encoding CDATA #REQUIRED
   pixel CDATA #IMPLIED
   maxsize CDATA #IMPLIED>

<!ELEMENT attachment-formats EMPTY>
<!ATTLIST  attachment-formats
   content-type CDATA #REQUIRED
   charset CDATA #IMPLIED>

<ELEMENT filtering-parameters EMPTY >
<!ATTLIST filtering-parameters
  created CDATA #IMPLIED
```

```
  modified CDATA #IMPLIED
  encoding CDATA #IMPLIED
  pixel CDATA # IMPLIED>

<!ELEMENT DPOF-options EMPTY >
<!ATTLIST DPOF-options
   standard-print CDATA #IMPLIED
   index-print CDATA #IMPLIED
   multiple-image-print CDATA #IMPLIED
   specific-size-print CDATA #IMPLIED
   number-sets CDATA #IMPLIED
   character-stamp CDATA #IMPLIED
   trimming CDATA #IMPLIED>
]>
```

The elements used in the imaging-capabilities object are defined as follows:

Element Name	Meaning
imaging-capabilities	A detailed description of the options, attributes, and formats supported by an imaging device.
Image-formats	Describes the formats in which images can be received by the device.
preferred-format	Describes the format in which an imaging device prefers to receive images. The preferred-format is to be chosen among the formats that the image-formats element indicates as supported on the imaging device.
attachment-formats	Describes supported attachment formats.
filtering-parameters	Describes an imaging device's ability to filter images as it builds images-listing objects. If a device supports filtering based on a particular image attribute, that attribute is listed in this element with the value "1". If a device does not support filtering based on a particular image attribute, that attribute is either listed in this element with the value "0" or is not listed at all (these methods are equivalent and shall be properly understood by an object exchange client that parses imaging-capabilities objects). If multiple attributes are listed in this element, the imaging device shall be capable of filtering using any combination of the supported attributes.
DPOF-options	Describes supported DPOF options. If a device supports a particular DPOF option, an attribute for that option appears in this element with the value "1". If a device does not support a particular DPOF option, an attribute for that option appears in this element with the value "0" or does not appear at all.

Table 4.10: Elements Used in the Imaging-Capabilities Object

The attributes used in the imaging-capabilities object are defined as follows:

Attribute Name	Meaning
version	The version of the imaging-capabilities XML string.

Attribute Name	Meaning
encoding	In the image-formats or preferred-format elements, this attribute describes the encodings supported by the device. The encodings supported by this profile include JPEG, GIF, WBMP, PNG, JPEG2000, BMP, and USR-xxx. The tag USR-xxx is used to represent proprietary encodings. The tag shall begin with the string “USR-” but the implementer assigns the characters of the second half of the string. This tag can be used by a manufacturer to enable its devices to exchange images under a proprietary encoding. It is recommended that the name of the manufacturer or its abbreviation be included in the tag (for instance, USR-NOKIA-FORMAT1). It is highly probable that devices manufactured by other companies would not understand this encoding. In the filtering-parameters element, this attribute indicates whether or not the device can filter the images-listing object based on encoding.
pixel	In the image-formats or preferred-format elements, this attribute describes the pixel size or the range of pixel sizes supported by the device. A fixed pixel size is expressed using the format W(idth)*H(eight). The possible values for W(idth) and H(eight) range from 0 to 65535. If the image can be resized to allow for a modified aspect ratio, the pixel range is indicated using the format W(idth)1*H(eight)1-W(idth)2*H(eight)2, where W(idth)1*H(eight)1 indicates the smallest image size and W(idth)2*H(eight)2 gives the largest image size. 65535*65535 can be used as W(idth)2*H(eight)2 to indicate the absence of an upper limit. For image resizing with a fixed aspect ratio, the pixel range is expressed using the format W(idth)1**- W(idth)2*H(eight)2, where W1 is the smallest width possible and W2*H2 is the largest size possible. For each possible intermediate value of W(idth), the corresponding height is calculated using the formula H=(W*H2)/W2. In the filtering-parameters element, this attribute indicates whether or not the device can filter the images-listing object based on pixel size.
transformation	This attribute describes the preferred transformation method. If an object exchange client is sending an image to an object exchange server and must apply a transformation to the image before sending it, the client can learn the server’s preferred transformation method from this attribute. “stretch” indicates stretching is preferred, “fill” indicates filling with padding data is preferred, and “crop” indicates cropping is preferred.
maxsize	This attribute defines the maximum file size in bytes acceptable for an image. This attribute is used to set an upper limit to the size of image files that can be accepted in a particular format.
content-type	This attribute lists the MIME content-type of a supported attachment type; for example, content-type=“text/plain”. See http://www.isi.edu/in-notes/iana/assignments/media-types/ for content types.
charset	This attribute lists the MIME character set of a supported attachment type; for example, charset=“iso-8859-1”.
created	This attribute indicates whether or not the device can filter the images-listing object based on creation date.
modified	This attribute indicates whether or not the device can filter the images-listing object based on modification date.
standard-print	This attribute indicates support for DPOF standard print (PRT TYP=STD).
index-print	This attribute indicates support for DPOF index print (PRT TYP=IDX).
multiple-image-print	This attribute indicates support for DPOF multiple image print (PRT TYP=MUL).

Attribute Name	Meaning
specific-size-print	This attribute indicates support for DPOF specific size print (PRT TYP = SIZ).
number-sets	This attribute indicates support for the DPOF QTY parameter.
character-stamp	This attribute indicates support for the DPOF DSC parameter.
trimming	This attribute indicates support for the DPOF TRM parameter.

Table 4.11: Attributes used in the imaging-capabilities Object

Table 4.12 indicates the level of support required for each element of the imaging-capabilities object based on the imaging features implemented by a device.

	Image Push	Image Pull	Image Advanced Print	Automatic Archive	Remote Camera	Remote Display
Situation	R	R	R	I	R	R
image-formats	M	N/U	M	N/U	N/U	M
preferred-format	O	N/U	O	N/U	N/U	O
attachment-formats	O	N/U	N/U	N/U	N/U	N/U
filtering-parameters	N/U	O	N/U	O	N/U	O
DPOF-options	N/U	N/U	M	N/U	N/U	N/U

Table 4.12: Minimum requirements for the imaging-capabilities object depending on the feature in which it is used

R: Imaging Responder returns the imaging-capabilities object.

I: Imaging Initiator returns the imaging-capabilities object.

M: Corresponding element shall be included in the imaging-capabilities object.

O: Corresponding element may be included in the imaging-capabilities object.

N/U: The corresponding element may or may not be included in the imaging-capabilities object, but is not used in the feature.

Example Imaging-Capabilities XML String

An example imaging-capabilities object for a device supporting the Image Push and Image Pull features is shown below.

```
<imaging-capabilities version="1.0">
<preferred-format encoding="JPEG" pixel="1280*960" />
<image-formats encoding="JPEG" pixel="160*120" maxsize="5000" />
<image-formats encoding="JPEG" pixel="320*240" />
<image-formats encoding="JPEG" pixel="640*480" />
<image-formats encoding="JPEG" pixel="1280*960" />
<attachment-formats content-type="audio/basic" />
<filtering-parameters
 created="1" modified="1" />
</imaging-capabilities>
```

Note that the preferred-format, image-formats, and attachment-formats elements only make sense for an Image Push or Remote Display scenario (where the Imaging Initiator has to learn which formats the Imaging Responder supports) or an Advanced Image Printing scenario (where the Imaging Initiator prepares the images in a format the

Imaging Responder understands). The filtering-parameters element is used only when the Imaging Initiator uses an imaging feature that includes the GetImagesList function.

An example imaging-capabilities object for a device supporting the Advanced Image Printing feature is shown below.

<imaging-capabilities version="1.0">
<image-formats encoding="JPEG" pixel="640*480-1600*1200" />
<DPOF-options standard-print="1" index-print="1" number-sets="1" trimming="1" />
</imaging-capabilities>

4.4.6.4 Printer-Control Object (x-bt/img-print)

The printer-control object is a text description of a print job based on the DPOF 1.1 standard; the following figure illustrates an example file system structure and its associated DPOF file.

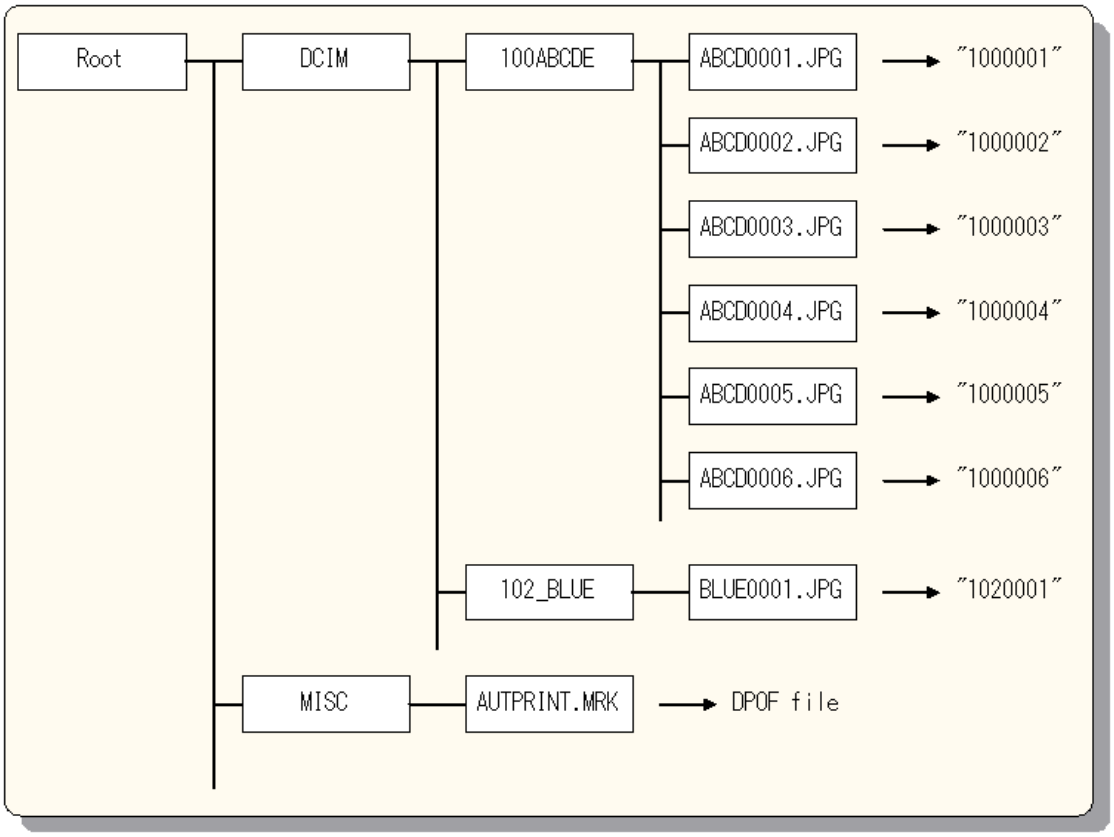


Figure 4.8: Example File System Structure

[HDR]
GEN REV = 01.10
GEN CRT = “Bluetooth camera” -01.00
GEN DTM = 2001:01:01:12:00:00

[JOB]
PRT PID = 001
PRT TYP = STD
PRT QTY = 001
IMG FMT = EXIF2 -J
<IMG SRC = “../DCIM/100ABCDE/ABCD0001.JPG”>
CFG DSC = “100-0001” -ATR FID

[JOB]
PRT PID = 002
PRT TYP = STD
PRT QTY = 002
IMG FMT = EXIF2 -J
<IMG SRC = “../DCIM/100ABCDE/ABCD0002.JPG”>
CFG DSC = “2000.12.24” -ATR DTM
CFG DSC = “100-0002” -ATR FID

[JOB]
PRT PID = 003
PRT TYP = STD
PRT QTY = 001
IMG FMT = EXIF2 -J
<IMG SRC = “../DCIM/100ABCDE/ABCD0003.JPG”>
CFG DSC = “2000.12.25” -ATR DTM
CFG DSC = “100-0003” -ATR FID

[JOB]
PRT PID = 004
PRT TYP = STD
PRT QTY = 003
IMG FMT = EXIF2 -J
<IMG SRC = “../DCIM/102_BLUE/BLUE0001.JPG”>
CFG DSC = “102-0001” -ATR FID

[JOB]
PRT PID = 100
PRT TYP = IDX
PRT QTY = 001
IMG FMT = EXIF2 -J
IMG SRC = “../DCIM/100ABCDE/ABCD0001.JPG”
IMG SRC = “../DCIM/100ABCDE/ABCD0002.JPG”

IMG SRC = "../DCIM/100ABCDE/ABCD0003.JPG"
IMG SRC = "../DCIM/102_BLUE/BLUE0001.JPG"

4.4.6.5 Monitoring-Image Object (x-bt/img-monitoring)

The monitoring-image object is created in the Remote Camera feature by an image capture device, typically a digital still camera. The monitoring-image is the same image that is displayed in a digital still camera's viewfinder or LCD display, and it's typically based on the through-the-lens image but might – in some implementations – imply operating a mechanical shutter. A monitoring-image is not permanently stored on a digital still camera and hence doesn't have an image handle associated with it. When an object exchange client requests a monitoring-image it should specify an OBEX Type header with the value "x-bt/img-monitoring". The returned monitoring-image is in the imaging thumbnail format defined in Section 4.4.2. Note, however, that the monitoring-image is likely to be of lower quality than a true imaging thumbnail or full size image because digital still cameras typically do not perform the same amount of processing on their monitoring/viewfinder images as they do on captured images.

4.4.7 Imaging Descriptors

The imaging descriptors are used to control the content of images-listing objects, describe the characteristics of images as they're transferred, and describe the characteristics of attachments associated with images as they're transferred. Imaging descriptors are transferred in an OBEX User Defined Header called "Img-Description" (see Section 5).

4.4.7.1 Image Handles Descriptor

The image handles descriptor is based on the DTD for the images-listing XML string (see Section 4.4.6.1).

```
<!DOCTYPE image-handles-descriptor [

<!ELEMENT image-handles-descriptor ( filtering-parameters ) >
<!ATTLIST image-handles-descriptor
   version CDATA #FIXED "1.0" >

<!ELEMENT filtering-parameters EMPTY>
<!ATTLIST filtering-parameters
   created CDATA #IMPLIED
   modified CDATA #IMPLIED
  encoding CDATA #IMPLIED
  pixel CDATA # IMPLIED>
]>
```

The elements used in the image handles descriptor are defined as follows:

Element Name	Meaning
image-handles-descriptor	Controls the content of an image-listing object.
filtering-parameters	In a Request:

Element Name	Meaning
	Describes the filtering parameters to be used by an object exchange server when selecting the images in an images-listing object. When the filtering-parameters element contains several filtering parameters, the filtering parameters supported by the Server are logically ANDed in the filtering operation. The other ones are ignored. In a Response: Describes the filtering parameters that were applied to the Images-Listing object. In case no filtering was applied (either because no filtering was requested or because none of the requested filtering parameters are supported by the Server), the Server can return an empty Img-Description header or an Img-Description header with a XML string lacking a filtering-parameters attribute.

Table 4.13: Elements Used in the Image Handles Descriptor

The attributes used in the image handles descriptor are defined as follows:

Attribute Name	Meaning
version	The version of the image-handles-descriptor XML string.
created	The range of image creation dates to be included in the images-listing object. The format is YYYYMMDDTHHMMSS-YYYYMMDDTHHMMSS, where the capital letter 'T' is explicitly inserted between the day and the hour fields. It is recommended that whenever possible UTC time is used; when UTC time is used, the letter 'Z' is appended to the end of the string. It is possible to replace one side of the above expression with a '*' to specify all images created before or after a given date.
modified	The range of modification dates to be included in the images-listing object. It uses the same format as the created attribute.
encoding	This attribute describes the image encoding to be used in the filtering operation. It is not possible to indicate more than one encoding. The encodings that are supported by this profile include JPEG, GIF, WBMP, PNG, JPEG2000, BMP, and USR-xxx. The tag USR-xxx is used to represent proprietary encodings. The tag shall begin with the string "USR-" but the implementer assigns the characters of the second half of the string. This tag can be used by a manufacturer to enable its devices to exchange images under a proprietary encoding. It is recommended that the name of the manufacturer or its abbreviation be included in the tag (for instance, USR-NOKIA-FORMAT1). It is highly probable that devices manufactured by other companies would not understand such an encoding.
pixel	The pixel size or range of pixel sizes to use for the filtering operation. A fixed pixel size is expressed using the format W(idth)*H(eight). The possible values for W(idth) and H(eight) range from 0 to 65535. If the image can be resized to allow for a modified aspect ratio, the pixel range is indicated using the format W(idth)1*H(eight)1-W(idth)2*H(eight)2, where W(idth)1*H(eight)1 indicates the smallest image size and W(idth)2*H(eight)2 gives the largest image size. 65535*65535 can be used as W(idth)2*H(eight)2 to indicate the absence of an upper limit. For image resizing with a fixed aspect ratio, the pixel-range is expressed using the format W(idth)1**- W(idth)2*H(eight)2, where W1 is the smallest width possible and W2*H2 is the largest size possible. For each possible intermediate value of W(idth), the corresponding height is calculated using the formula H=(W*H2)/W2.

Table 4.14: Attributes Used in the Image Handles Descriptor

If a filtering parameter is not to be used, it can be omitted entirely or set to an empty string.

The following example illustrates the use of the image handles descriptor. An object exchange client that wants to retrieve images created on the 1st of January 2000 would construct this image handles descriptor:

< image-handles-descriptor version="1.0" >
< filtering-parameters created="20000101T000000Z-20000101T235959Z" />
< /image-handles-descriptor >

Note carefully that, unlike the images-listing object, the created and modified attributes in the images-handle descriptor are specified as ranges with start and stop dates/times separated by a '-' character.

4.4.7.2 Image Descriptor

The image descriptor is based on the DTD for the image-properties XML string (Section 4.4.6.2). It is used to specify the properties of an image as it is transferred. For the PutImage function, the image descriptor describes the properties of the image being pushed. For the GetImage function, the image descriptor describes the image-properties the object exchange client wants the object exchange server to provide.

```
<!DOCTYPE image-descriptor [

<!ELEMENT image-descriptor ( image ) >
<!ATTLIST image-descriptor version CDATA #FIXED "1.0" >

<!ELEMENT image EMPTY>
<!ATTLIST image
   encoding CDATA #REQUIRED
   pixel CDATA #REQUIRED
   size CDATA #IMPLIED

   maxsize CDATA #IMPLIED
   transformation (stretch | fill | crop ) #IMPLIED
]>
```

The elements used in the image descriptor are defined as follows:

Element Name	Meaning
image-descriptor	A description of an image's properties.
image	An individual image and its properties.

Table 4.15: Elements Used in the Image Descriptor

The attributes used in the image descriptor are defined as follows:

Attribute Name	Meaning
version	The version of the images-descriptor XML string.

Attribute Name	Meaning
encoding	The encoding for an image. The encodings supported by this profile include JPEG, GIF, WBMP, PNG, JPEG2000, BMP, and USR-xxx. The tag USR-xxx is used to represent proprietary encodings. The tag shall begin with the string “USR-” but the implementer assigns the characters of the second half of the string. This tag can be used by a manufacturer to enable its devices to exchange images under a proprietary encoding. It is recommended that the name of the manufacturer or its abbreviation be included in the tag (for instance, USR-NOKIA-FORMAT1). It is highly probable that devices manufactured by other companies would not understand this encoding. This attribute is mandatory. For the PutImage function, it is mandatory to inform the object exchange server of the image’s encoding. For the GetImage operation, this attribute is mandatory but can be empty – an empty value indicates to the object exchange server that any encoding is acceptable.
pixel	The pixel size (or range of acceptable pixel sizes) for an image. This attribute is mandatory. 1. For the PutImage function: It is mandatory to inform the object exchange server of the image’s size. Ranges are not acceptable. Only fixed sizes are acceptable and are expressed using the format W(idth)*H(eight). The possible values for W(idth) and H(eight) range from 0 to 65535. 2. For the GetImage function: This attribute may be empty – an empty value indicates to the object exchange server that any pixel size is acceptable. In addition to a fixed value, it is also possible to pass a range as a value, to inform that any size belonging to this range is acceptable. A free pixel range (not bound by aspect ratio considerations) is indicated using the format W(idth)1*H(eight)1-W(idth)2*H(eight)2, where W(idth)1*H(eight)1 indicates the smallest image size and W(idth)2*H(eight)2 gives the largest image size. 65535*65535 can be used as W(idth)2*H(eight)2 to indicate the absence of an upper limit. If a range of pixel sizes is acceptable but with the extra condition that the aspect ratio stays untouched, the range shall be expressed using the format W(idth)1**- W(idth)2*H(eight)2, where W1 is the smallest width possible and W2*H2 is the largest size possible. For each possible intermediate value of W(idth), the corresponding height is calculated using the formula H=(W*H2)/W2.
size	The size in bytes of the image file. This attribute only makes sense for image-descriptors used for the PutImage function.
maxsize	The maximum size in bytes acceptable for the image file. This attribute only makes sense for image-descriptors used for the GetImage function. The value of the maxsize attribute is to be used by the GetImage Server as target file size (best effort) for the variant image to produce.
transformation	1. For the PutImage function, the transformation applied to an image before it was transferred. 2. For the GetImage function, the transformation that should be applied to an image before it will be transferred. “stretch” indicates image stretching, “fill” indicates filling with padding data, and “crop” indicates image cropping.

Table 4.16: Attributes Used in the Image Descriptor

An object exchange client invoking the PutImage function with a JPEG encoded image with size 1280*960 pixels and a size of 500000 bytes would construct this image descriptor:

< image-descriptor version="1.0" >
< image encoding="JPEG" pixel="1280*960" size="500000"/>
< /image-descriptor >

An object exchange client invoking the GetImage function and desiring a JPEG encoded image with size 1280*960 pixels and 180 Kbytes (184320 bytes) as maximum acceptable size would construct this image descriptor:

<image-descriptor version="1.0" >
< image encoding="JPEG" pixel="1280*960" maxsize="184320" />
</image-descriptor>

Omitting an attribute from an imaging descriptor used in conjunction with a GetImage function signals no preference with respect to the omitted attribute. For example, if an object exchange client specifies only an encoding attribute of JPEG, the object exchange server is free to deliver a JPEG image of any size.

The pixel attribute can also describe a range of acceptable sizes. For example, if an object exchange client invoking the GetImage function wants an image of 640*480 or smaller, it would specify the pixel attribute as "0*0-640*480". If the client wants the aspect ratio of those 640*480 or smaller images to be constant, it would specify the pixel attribute as "0**-640*480".

4.4.7.3 Attachment Descriptor

The attachment descriptor is based on the DTD for the image-properties object (Section 4.4.6.2).

```
<!DOCTYPE attachment-descriptor [
<!ELEMENT attachment-descriptor ( attachment ) >
<!ATTLIST attachment-descriptor version CDATA #FIXED "1.0" >
<!ELEMENT attachment EMPTY>
<!ATTLIST attachment
   content-type CDATA #IMPLIED
   charset CDATA #IMPLIED
   name CDATA #IMPLIED
   size CDATA #IMPLIED
   created CDATA #IMPLIED>
]>
```

The attachment descriptor is used in conjunction with the PushLinkedAttachment function to describe the properties of the attachment to the object exchange server.

The elements used in the attachment descriptor are defined as follows:

Element Name	Meaning
attachment-descriptor	A description of an attachment associated with an image.
attachment	An individual attachment and its properties.

Table 4.17: Elements used in the Attachment Descriptor

The attributes used in the attachment descriptor are defined as follows:

Attribute Name	Meaning
version	The version of the attachment-descriptor XML string.
content-type	The MIME content-type of the attachment; for example, content-type="text/plain". See http://www.isi.edu/in-notes/iana/assignments/media-types/ for content types.
charset	The MIME character set of the attachment; for example, charset="iso-8859-1".
name	The file name of the attachment, which may or may not include a path. This name is intended to be human-readable.
size	The size in bytes of the attachment.
created	The attachment's creation time. The format is YYYYMMDDTHHMMSS, where the capital letter 'T' is explicitly inserted between the day and the hour fields. It is recommended that whenever possible UTC time is used; when UTC time is used, the letter 'Z' is appended to the end of the string.

Table 4.18: Attributes Used in the Attachment Descriptor

An example attachment descriptor follows:

< attachment-descriptor version="1.0" >
< attachment name="DSCF0001.txt" content-type="text/plain" size="5000" />
< /attachment-descriptor >

4.5 Imaging Functions

This section describes each function defined in the Basic Imaging Profile. The request and response message formats associated with each function are described in tables that show the mandatory OBEX fields and headers in the OBEX frame. Note that the position of the fields and headers within the frame as illustrated in the following tables shall be strictly followed. Headers that are not specified in the tables may be used and placed after the headers that are specified, but this profile does not guarantee that other implementations will interpret them as intended.

4.5.1 GetCapabilities Function

The GetCapabilities function is used to retrieve the imaging-capabilities object of an object exchange server.

The GetCapabilities request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-capabilities

Table 4.19: Format of the GetCapabilities Request

The GetCapabilities response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Body/EndOfBody	Imaging-capabilities object

Table 4.20: Format of the GetCapabilities Response

* See Section 5.3 for OBEX error codes.

4.5.2 PutImage Function

The PutImage function is used to push an image to an object exchange server.

Although not a mandatory requirement, it is highly recommended that PutImage attempts to be preceded by retrieving the imaging-capabilities object of the object exchange server so that images are sent to a server in a format the server supports.

An object exchange server may use the PutImage response to request that the client send the thumbnail version of the image it just received. This capability is designed for servers that don't have the ability to convert images into the imaging thumbnail format.

The PutImage request message is formatted as follows:

Fields	Opcode	Put
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-img
	Name	Image name
	Img-Description	Image descriptor
	Body/EndOfBody	Image file

Table 4.21: Format of the PutImage Request

The PutImage response is formatted as follows:

Fields	Response Code	Success or Partial Content or error code*
	Packet Length	Length of the packet
Headers	Img-Handle	Image handle

Table 4.22: Format of the PutImage Response

* See Section 5.3 for OBEX error codes.

The image handle assigned to the newly received image by the object exchange server shall always be returned in the response message. This is done so the object exchange client can send the thumbnail and/or attachments that might be linked to the image using the PutLinkedThumbnail or PutLinkedAttachment functions.

The Partial Content response code indicates that the server requests the thumbnail version of the image just sent by the PutImage function. If an operation involves several request-response messages (i.e., the image being transferred doesn't fit in one OBEX request message), the server shall respond with Partial Content in the very last response packet (where it replaces the Success response code). The intermediate response packets shall carry the Continue response code. It is mandatory for the client to supply the thumbnail version of the image in the operation immediately following the PutImage operation if so requested by the server. In no case shall the connection be

terminated without sending the corresponding thumbnail image to the object exchange server.

4.5.3 PutLinkedThumbnail Function

The PutLinkedThumbnail function is a scaled-down version of the PutImage function that does not use the image-descriptor header. It is only possible to push the thumbnail version of an image; it is not possible to push any other format using this function. The object exchange client uses this function only in response to a PutImage response indicating the object exchange server needs the thumbnail version of an image.

The PutLinkedThumbnail request is formatted as follows:

Fields	Opcode	Put
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-thm
	Img-Handle	Image handle
	Body/EndOfBody	Imaging thumbnail file

Table 4.23: Format of the PutLinkedThumbnail Request

The PutLinkedThumbnail response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet

Table 4.24: Format of the PutLinkedThumbnail Response

* See Section 5.3 for OBEX error codes.

4.5.4 PutLinkedAttachment

The PutLinkedAttachment is used to send attachments associated with an image to an object exchange server after the image has been sent to the server within the context of an OBEX connection. The PutLinkedAttachment function shall be used within the OBEX connection during which the image was sent to the object exchange server.

The PutLinkedAttachment request is formatted as follows:

Fields	Opcode	Put
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-attachment
	Img-Handle	Image handle
	Img-Description	Attachment descriptor
	Body/EndOfBody	Attachment file

Table 4.25: Format of the PutLinkedAttachment Request

The PutLinkedAttachment response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet

Table 4.26: Format of the PutLinkedAttachment Response

* See Section 5.3 for OBEX error codes.

4.5.5 RemoteDisplay Function

The RemoteDisplay function is used to pilot the screen of an object exchange server with display capability. The screen control commands are: NextImage (display the next image), PreviousImage (display the previous image), SelectImage (display a specific image), and CurrentImage (retrieve the handle of the currently displayed image).

It is up to the object exchange server to determine which images to display when the object exchange client sends the NextImage or PreviousImage command. It is recommended that images be displayed in the order they were pushed, assuming they were pushed within the same OBEX connection. An object exchange client may also learn the image order by retrieving the images-listing object, or by examining the Img-Handle header in the response packets.

When the object exchange client wants to display images that have been sent to the server during a previous connection, the client should retrieve the images-listing object from the object exchange server, select an image, and send the SelectImage command to the server.

It is strongly recommended that an object exchange client check the availability of the RemoteDisplay function in the SDP database of the server before attempting to use it. The minimum set of supported attributes should be confirmed: imaging-capabilities should include storage and display and supported features and functions should include image push and remote display.

The RemoteDisplay request message is formatted as follows:

Fields	Opcode	Put
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-display
	Img-Handle	Image handle
	Application Parameters	RemoteDisplay

Table 4.27: Format of the RemoteDisplay Request

The Application Parameters header is formed from one “tag ID” byte, one “length” byte, and one “value” byte (see [4]). The tag ID byte for RemoteDisplay is listed in Section 5.2.1. The values associated with the RemoteDisplay tag ID are as follows:

- NextImage = 0x01
- PreviousImage = 0x02
- SelectImage = 0x03
- CurrentImage = 0x04

If the SelectImage value is used, the Img-Handle header shall be present and must be set to the handle of the image to display. The Img-Handle header must also be present for the NextImage, PreviousImage, and CurrentImage values, but shall be empty.

Devices that implement the Remote Display feature must not automatically display images as they receive them via PutImage requests – instead, they shall wait for a RemoteDisplay request. Note that devices implementing Remote Display shall also set the display and store imaging-capabilities in their SDP service record(s).

The RemoteDisplay response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Img-Handle	Displayed image handle

Table 4.28: Format of the RemoteDisplay Response

* See Section 5.3 for OBEX error codes.

The Img-Handle header contains the handle of the currently displayed image. If there is no currently displayed image, the Img-Handle header shall be present but empty.

4.5.6 GetImagesList Function

The GetImagesList function retrieves the object exchange server's images-listing object.

The GetImagesList request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-listing
	Application Parameters	NbReturnedHandles
		ListStartOffset
		LatestCapturedImages
	Img-Description	Image handles descriptor

Table 4.29: Format of the GetImagesList Request

The Application Parameters headers are formed from one "tag ID" byte, one "length" byte, and *n* "value" bytes (see [4]). The tag IDs for the three Application Parameters headers are as follows:

- NbReturnedHandles
- ListStartOffset
- LatestCapturedImages

The numeric values for these tag IDs are listed in Section 5.2.1.

The NbReturnedHandles tag ID indicates the maximum number of image handles to be returned in the images-listing object. The length field for this header is two bytes; these two bytes of data are represented in the value field in big-endian order. The valid range for the value field is 0 to 65535, inclusive.

If an object exchange client does not want to limit the number of images returned in the images-listing object, it should set the value field of the NbReturnedHandles header to the maximum, 65535. If an object exchange client wants to learn the number of images that would be included in the images-listing object but doesn't want the actual list, it should set the value field of the NbReturnedHandles header to 0. The object exchange server will respond with an empty images-listing object and an NbReturnedHandles header describing the number of images that would have been included in the list.

Note that responding with an images-listing object – or even calculating the number of images that would go into an images-listing object – may take an object exchange server a significant amount of time.

The ListStartOffset tag ID describes a zero-based offset from the beginning of the images-listing object. Its value is encoded in two bytes using big-endian byte ordering. This mechanism can be used to retrieve an object exchange server's images-listing object in pieces; for example, an object exchange client could send a GetImagesList request with the NbReturnedHandles value set to 10 and the ListStartOffset value set to 0 to retrieve a list of the first ten images, followed by a second GetImagesList request with the NbReturnedHandles value again set to 10 and the ListStartOffset value set to 10 to retrieve a list of the next ten images, etc.

Note that when used in a Remote Display feature session, it is highly recommended that the list of images in the images-listing object be listed in the same order as the intended order of display.

The LatestCapturedImages tag ID restricts the scope of the images-listing object to the most recently captured images and controls the order of images within the list. Its value is encoded in one byte and can take only two values, 0x00 or 0x01 (all other values are reserved). Setting the LatestCapturedImages header's value to 0x01 indicates that the images-listing object shall include only locally captured images sorted chronologically by descending capture time. The size of the list is left to the implementer. In particular, the list does not necessarily have to include all the images that have been captured by the object exchange server; the implementer could, for example, decide to keep only a subset of its most recent images sorted in chronological order and return that list. If an object exchange client sets the value of the LatestCapturedImages header to 0x01, the value of the ListStartOffset header shall be 0. Setting the LatestCapturedImages header's value to 0x00 results in an unrestricted, unordered images-listing object.

The Img-Description header contains an image handles descriptor that specifies a filtering mask to apply to the images-listing object. See Section 4.4.7.1 for details about the image handles descriptor. If no filtering is required, an object exchange client can leave the Img-Description header empty, although it shall always be present.

An object exchange server is required to provide an object exchange client with a consistent view of its images-listing object within the context of an OBEX connection. For example, if a client is stepping through the server's images-listing object piecewise via the NbReturnedHandles and ListStartOffset Application Parameters headers, then any other activity on the server that results in adding an image to or removing an image from the server shall not affect the images-listing object. This policy may result in the client having a "stale" view of the images stored on the server, so the client shall be prepared to handle errors encountered in subsequent operations – for example, using the GetImage function to retrieve an image from the server that was deleted at some point in time between the client's learning of the image's existence via a GetImagesList operation and the subsequent GetImage operation will result in an error.

The GetImagesList response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Application Parameters	NbReturnedHandles
	Img-Description	Image handles descriptor
	Body/EndOfBody	Images-listing object**

Table 4.30: Format of the GetImagesList Response

* See Section 5.3 for OBEX error codes.
** The images-listing object in the response message may be a subset of the object exchange server's full image list depending on how the NbReturnedHandles and ListStartOffset headers in the request message were set.

The image handles descriptor included in the response is used to indicate the filtering parameters that were applied when building the images-listing object.

If all the filtering parameters indicated in the GetImagesList request were applied, the image handles descriptors in the request and response messages will typically be identical. However, depending on the implementation of the Server, and provided that the image handles descriptor of the response contains the same filtering parameters as the image handles descriptor in the request, the values attributed those filtering parameters may differ from the original ones. For instance, a Server implementation that does not support the UTC notation used by the Client in the image handles descriptor of the request, might return values for the created or modified filtering parameter in local time format.

4.5.7 GetImageProperties Function

The GetImageProperties function retrieves a description of an image's characteristics. The object exchange client references the image via its image handle and the object exchange server returns the image's image-properties object (see Section 4.4.6.2).

The GetImageProperties request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-properties
	Img-Handle	Image handle

Table 4.31: Format of the GetImageProperties Request

The GetImageProperties response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Body/EndOfBody	Image-properties object

Table 4.32: Format of the GetImageProperties Response

* See Section 5.3 for OBEX error codes.

4.5.8 GetImage Function

The GetImage function is used by an object exchange client to retrieve an image from an object exchange server. The client references the image via its image handle and supplies a description of the image format the server should use.

The GetImage request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-img
	Img-Handle	Image handle
	Img-Description	Image-descriptor

Table 4.33: Format of the GetImage Request

It is possible for the object exchange client to supply an empty Img-Description header, in which case the object exchange server returns the native version of the image. This is useful for clients that don't need to negotiate image encodings and sizes.

The GetImage response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Length	Image file size
	Body/EndOfBody	Image file

Table 4.34: Format of the GetImage Response

* See Section 5.3 for OBEX error codes.

The Length header that indicates the size in bytes of the returned image is mandatory but can be left empty, in case the byte size of the transferred image is not known (this could happen, for instance, if the image is being produced on the fly).

4.5.9 GetLinkedThumbnail Function

The GetLinkedThumbnail function is a scaled-down version of the GetImage function that does not use the image descriptor. It is therefore only possible to retrieve the thumbnail version of an image given its handle. It is not possible to request any other format using this function. This function is designed for devices that deal only with thumbnail images.

The GetLinkedThumbnail request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-thm
	Img-Handle	Image handle

Table 4.35: Format of the GetLinkedThumbnail Request

The GetLinkedThumbnail response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Body/EndOfBody	Imaging thumbnail file

Table 4.36: Format of the GetLinkedThumbnail Response

* See Section 5.3 for OBEX error codes.

The image file that is returned in the response message shall be an imaging thumbnail.

4.5.10 GetLinkedAttachment Function

The GetLinkedAttachment function is used to retrieve an attachment associated with an image from an object exchange server. The presence of one or more attachments can be discovered by retrieving an image's image-properties object via the GetImageProperties function. Attachments are referenced by the associated image's handle and the attachment's file name.

The GetLinkedAttachment request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-attachment
	Img-Handle	Image handle
	Name	Attachment file name

Table 4.37: Format of the GetLinkedAttachment Request

The GetLinkedAttachment response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Body/EndOfBody	Attachment file

Table 4.38: Format of the GetLinkedAttachment Response

* See Section 5.3 for OBEX error codes.

4.5.11 DeleteImage Function

The DeleteImage function is used by an object exchange client to request an object exchange server to delete an image. It is recommended that in addition to deleting the image the server also delete any attachments associated with the image, but this is left to the implementer's discretion.

The DeleteImage request is formatted as follows:

Fields	Opcode	Put
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-img
	Img-Handle	Image handle

Table 4.39: Format of the DeleteImage Request

Note that there is no Body or EndOfBody header in the DeleteImage request frame.

The DeleteImage response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet

Table 4.40: Format of the DeleteImage Response

* See Section 5.3 for OBEX error codes.

4.5.12 StartPrint Function

The StartPrint function is used by an object exchange client to trigger an object exchange server with print capability to execute a print job. The print job is described in the printer-control object sent to the server. As a result of a StartPrint request, the server opens a new OBEX connection (referred to as a secondary connection) to the Referenced Object service of the client (see Section 5 for details).

The StartPrint request is formatted as follows:

Fields	Opcode	Put
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-print
	Application Parameter	ServiceID
	Body/EndOfBody	Printer control object

Table 4.41: Format of the StartPrint Request

The ServiceID (see [1] for a complete definition of this service record attribute) sent in the Application Parameters header is used by the object exchange server to determine which Referenced Objects service record on the client to read. If several Referenced Objects service records are available on the client (for instance, if the client is running several applications that use the Advanced Printing feature and each creates its own Referenced Objects service record), it is necessary to tell the server which Referenced Objects service record to use.

The Application Parameters header is formed from one tag ID byte, one length byte, and *n* value bytes (see [4]). The tag ID byte for ServiceID is listed in Section 5.2.1. The value associated with the ServiceID tag ID is the 128-bit (16-byte) ServiceID UUID described in the Referenced Objects service record of the client.

The StartPrint response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet

Table 4.42: Format of the StartPrint Response

* See Section 5.3 for OBEX error codes.

4.5.13 GetPartialImage Function

The GetPartialImage function is a modified version of the GetImage function that can be used to retrieve either an entire image file or part of an image file, depending on the requirements of the object exchange server. The images retrieved with the GetPartialImage function might be native images or variant images depending on the choice of the object exchange server. The images are not referenced using their image

handles, but rather the path and image file names as described in the IMG SRC tag in the printer-control object; see Section 4.4.6.4.

The GetPartialImage request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-partial
	Name	Image file name
	Application Parameters	PartialFileLength
		PartialFileStartOffset

Table 4.43: Format of the GetPartialImage Request

The Application Parameters header is formed from one tag ID byte, one length byte, and *n* value bytes (see [4]). The PartialFileLength header is the length of the partial file to be returned in the response; it is encoded in four bytes in big-endian byte order. The PartialFileStartOffset header specifies the first byte of the file from which the partial file starts. It is encoded in four bytes in big-endian byte order.

To retrieve an entire image, the PartialFileStartOffset header's value should be set to 0x00000000 and the PartialFileLength header's value should be set to 0xFFFFFFFF.

In the event PartialFileStartOffset + PartialFileLength is greater than the actual size of the image file, the last chunk of the image file (starting from PartialFileStartOffset byte) shall be returned.

The GetPartialImage response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Length	Length of partial file
	Application Parameter	TotalFileSize
		EndFlag
	Body/EndOfBody	Image subfile

Table 4.44: Format of the GetPartialImage Response

* See Section 5.3 for OBEX error codes.

The Length header specifies the length of the returned partial file in bytes; it is encoded in four bytes in big-endian byte order. The TotalFileSize Application Parameters header indicates the size of the entire file in bytes. The TotalFileSize tag ID's numerical value is listed in Section 5.2.1; the header's value field is encoded in four bytes in big-endian byte order. The EndFlag Application Parameters header indicates whether or not the partial image file transported in the response represents the end of the file. The EndFlag tag ID's numerical value is listed in Section 5.2.1; the header's value field is encoded in one byte, and is set to 0x01 when the partial file corresponds to the final part of the image file and is set to 0x00 otherwise – any other value is illegal.

4.5.14 StartArchive Function

The StartArchive function is used by an object exchange client to trigger an object exchange server to start draining the client of its image files (some of or all of the image files depending on the server's download algorithm).

The StartArchive request is formatted as follows:

Fields	Opcode	Put
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-archive
	Application Parameter	ServiceID

Table 4.45: Format of the StartArchive Request

The ServiceID (see [1] for a complete definition of this service record attribute) sent in the Application Parameters header is used by the object exchange server to determine which Automatic Archive service record on the client to read. If several Referenced Objects service records are available on the client (for instance, if the client is running several applications that use the Automatic Archive feature and each creates its own Automatic Archive service record), it is necessary to tell the server which Automatic service record to use.

The Application Parameters header is formed from one tag ID byte, one length byte, and *n* value bytes (see [4]). The tag ID byte for ServiceID is listed in Section 5.2.1. The value associated with the ServiceID tag ID is the 128-bit (16-byte) ServiceID UUID described in the Automatic Archive service record of the client.

The StartArchive response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet

Table 4.46: Format of the StartArchive Response

* See Section 5.3 for OBEX error codes.

4.5.15 GetStatus Function

The GetStatus function is used by the primary object exchange client to monitor a secondary connection (see Section 5.5.1).

The GetStatus request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-status

Table 4.47: Format of the GetStatus Request

The GetStatus response is formatted as follows:

Fields	Response Code	Success or Continue or error code*
	Packet Length	Length of the packet

Table 4.48: Format of the GetStatus Response

* See Section 5.3 for OBEX error codes.

A Success response code in the GetStatus response indicates that the secondary connection has successfully terminated. A Continue response code indicates that the secondary connection is still active. An error code indicates that the secondary connection is being affected by an error.

GetStatus requests can be issued by a primary object exchange client only upon reception of a StartArchive response message. Upon reception of a Success response code, the primary object exchange client shall disconnect the primary connection.

A primary object exchange server shall never issue a Success response code prior to a Disconnect request having been issued by the secondary object exchange client.

The rate at which a primary object exchange client issues GetStatus requests – or whether it bothers to issue them at all – is implementation-dependent. If there is an error condition on the secondary connection, the primary client can terminate the primary session with an Abort or Disconnect request.

4.5.16 GetMonitoringImage Function

The GetMonitoringImage function is used by an object exchange client to retrieve monitoring images from an object exchange server with capturing capability. Monitoring images are retrieved in the monitoring-image format (see Section 4.4.6.5). The client has the option of indicating whether the monitoring-image retrieval operation should be accompanied by releasing the shutter and storing the corresponding full size image on the object exchange server (see Section 4.3.5).

The GetMonitoringImage request is formatted as follows:

Fields	Opcode	Get
	Packet Length	Length of the packet
Headers	ConnectionID	Connection identifier
	Type	x-bt/img-monitoring
	Application Parameters	StoreFlag

Table 4.49: Format of the GetMonitoringImage Request

The Application Parameters header is formed from one tag ID byte, one length byte, and one value byte (see [4]). The numeric value for the StoreFlag tag ID is listed in Section 5.2.1. The StoreFlag's value field can take the values 0x00 (indicating that the server should not store the full size image) and 0x01 (indicating that the server should store the full size image); any other value is illegal.

The GetMonitoringImage response is formatted as follows:

Fields	Response Code	Success or error code*
	Packet Length	Length of the packet
Headers	Img-Handle	Image handle
	Body/EndOfBody	Monitoring image object

Table 4.50: Format of the GetMonitoringImage Response

* See Section 5.3 for OBEX error codes.

When there is no image handle to return, the Img-Handle header shall be empty.

5 OBEX

5.1 OBEX Operations Used

Table 5.1 lists the OBEX operations required by the Basic Imaging Profile. Any other operations listed in the GOEP should not be used in this profile.

OBEX Operation	Ability to Send GOEP Client	Ability to Respond GOEP Server	Prohibited when using RFCOMM
Connect	M	M	
Disconnect	M	M	
Put	M	M	
Get	M	M	
Abort	M	M	
Action	X	X	Y
Session	X	X	Y
SetPath	X	X	Y

Table 5.1: OBEX Operations

5.2 OBEX Headers

Table 5.2 lists the OBEX headers required by the Basic Imaging Profile. Any other headers listed in GOEP should not be used in this profile.

OBEX Header	Client	Server	Prohibited when using RFCOMM
Name	M	M	
Type	M	M	
Length	M	M	
Body	M	M	
End of Body	M	M	
Target	M	M*	
Who	M*	M	
Connection ID	M	M	
Img-Description**	M	M	
Img-Handle**	M	M	
Authenticate Challenge	M	M	
Authenticate Response	M	M	
Application Parameters	M	M	
Single Response Mode	O	O	Y
Single Response Mode Parameters	C1	C1	Y
Session Parameters	X	X	Y
Session Sequence Number	X	X	Y

Table 5.2: OBEX Headers

C1: M if Single Response Mode is supported otherwise X
Ability to parse only (ability to send is not required).
** User-defined header.
Img-Description and Img-Handle are both user-defined headers (as defined in [4]).

Note that the profile does not exclude the headers that are not listed in Table 5.2. Some implementations might choose to use additional headers to enable added value

services. It is also the intention of the Bluetooth SIG to enrich further Bluetooth Imaging with a second profile development phase that could lead to new headers being added to the Basic Imaging Profile functions. Therefore unknown or unsupported headers shall always be skipped and ignored.

5.2.1 Application Parameters Header

The tag IDs used in the Application Parameters header are listed below.

Value	Tag ID	Length	Possible Values
NbReturnedHandles	0x01	2 bytes	0x0000 to 0xFFFF
ListStartOffset	0x02	2 bytes	0x0000 to 0xFFFF
LatestCapturedImages	0x03	1 byte	0x00 (= Boolean False) 0x01 (= Boolean True)
PartialFileLength	0x04	4 bytes	0x00000000 to 0xFFFFFFFF
PartialFileStartOffset	0x05	4 bytes	0x00000000 to 0xFFFFFFFF
TotalFileSize	0x06	4 bytes	0x00000000 to 0xFFFFFFFF
EndFlag	0x07	1 byte	0x00 (= Boolean False) 0x01 (= Boolean True)
RemoteDisplay	0x08	1 byte	0x01 (= NextImage) 0x02 (= PreviousImage) 0x03 (= SelectImage) 0x04 (= CurrentImage)
ServiceID	0x09	16 bytes	UUID
StoreFlag	0x0A	1 byte	0x00 (= Boolean False) 0x01 (= Boolean True)

Table 5.3: Application Parameter Header Tag IDs

All of the Application Parameter header values use big-endian byte ordering.

5.2.2 User-Defined Headers

The user defined headers Img-Handle and Img-Descriptor are defined as follows:

- Img-Handle header ID = 0x30 (null terminated, UTF-16 encoded Unicode text length prefixed with a two-byte unsigned integer)
- Img-Descriptor header ID = 0x71 (byte sequence, length prefixed with a two-byte unsigned integer)
- (The Img-Descriptor header is not null terminated.)

5.2.3 OBEX Headers in Multi-Packet Responses

In the case of multi-packet responses, there is a need to specify which packet contains the headers to be returned to the client. Although the IrOBEX specification does not impose any restrictions in this area, the following rule is used in the Basic Imaging Profile to encourage interoperability:

In the case of a multi-packet PUT response (i.e., the object being transported is large enough to require several packets), the headers, except SRM and SRMP, are placed in the last packet. All intermediate response packets shall contain only the Continue response code or (when necessary) one of the error codes. Note that if the Partial

Content response code is used (see Section 4.5.2), it shall be issued in the last response packet, after the object has been completely received.

In the case of a multi-packet GET response (i.e., the object being transported is large enough to require several packets), all the headers other than the body header are placed in the first packet. If the first packet has enough room to include a portion of the object body, then the first packet ends with the body header that carries this portion of the object. Otherwise, the object is transferred in subsequent packets.

The following figure illustrates a multi-packet PutImage operation. The intermediate Put response messages do not contain any header.

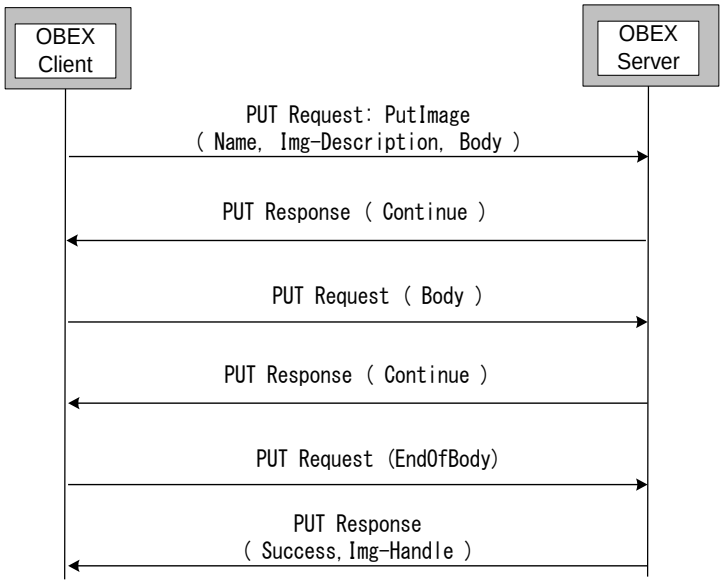


Figure 5.1: Example OBEX Packet Exchange Sequence for PutImage Operation

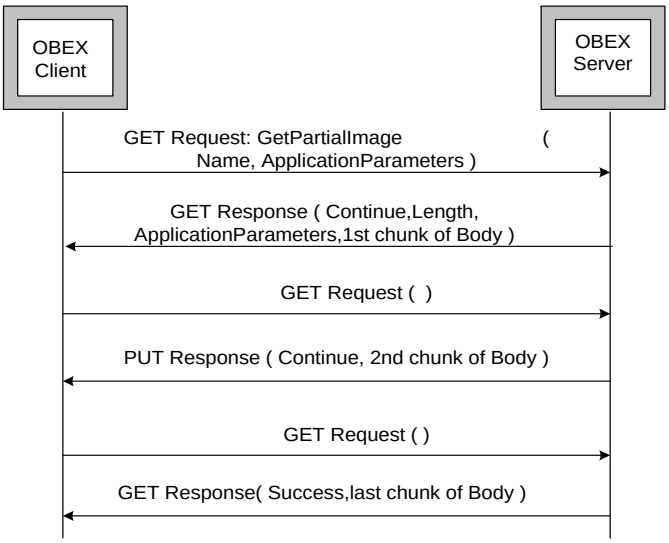


Figure 5.2: Example OBEX Packet Exchange Sequence for GetPartialImage Operation

Figure 5.2 illustrates a multi-packet GetImage operation. The first response packet shall contain the headers other than Body/EndOfBody (and possibly a Body header depending on the remaining number of bytes available in the frame).

5.3 OBEX Error Codes

Imaging Responders are required to support only two OBEX response codes:

- Bad Request: Indicates that the request could not be correctly interpreted or handled.
- Not Implemented: Indicates that the requested function is not supported.

Table 5.4 lists all of the OBEX response codes defined for the Basic Imaging Profile.

Error Code	Client (Interprets the Error Codes)	Server (Informs of Errors)	Meaning in Basic Imaging Profile
Bad Request	M*	M	Function not recognized or ill-formatted.
Not Implemented	M*	M	Function recognized but not supported.
Forbidden	M*	O	Function recognized and correctly formatted but temporarily barred.
Unauthorized	M*	O	In operations with actual exchange of an object in the body header (either in the request or the response), indicates that the function was recognized and well-formatted, but that the object to be handled is protected and access is not authorized (either temporarily or permanently).
Precondition Failed	M*	O	The function was recognized and well-formatted but there is a problem with one of the request's parameter values.
Not Found	M*	O	The function was recognized and well-formatted

Error Code	Client (Interprets the Error Codes)	Server (Informs of Errors)	Meaning in Basic Imaging Profile
			and all the parameters are proper, but the image handle or file name (depending on the function) could not be found.
Not Acceptable	M*	O	In case of a Push operation, the function is recognized and well-formatted, but the size of the body (indicated in the length header) is too big for the local buffers. In the case of a Get operation with an Img-Description, the function is recognized and well-formatted, but the XML descriptor requests a format that cannot be provided.
Service Unavailable	M*	O	The function was recognized and well-formatted and is normally executable, but a system condition temporarily prevents it from being performed.

Table 5.4: OBEX Response Codes

M*: indicates that the Imaging Initiator shall recognize this response code as an error code.

On the Imaging Initiator side, all the response codes listed in Table 5-4 must be recognized as error codes; how to handle these error codes is left to the implementer's discretion.

Support for response codes other than Bad Request and Not Implemented is optional; it is recommended, however, that as many of the others as possible be supported because they are more informative and give the client a better indication of the nature of an error; this permits better error reporting. The "x complements y" relationship between response codes is illustrated in Figure 5.3.

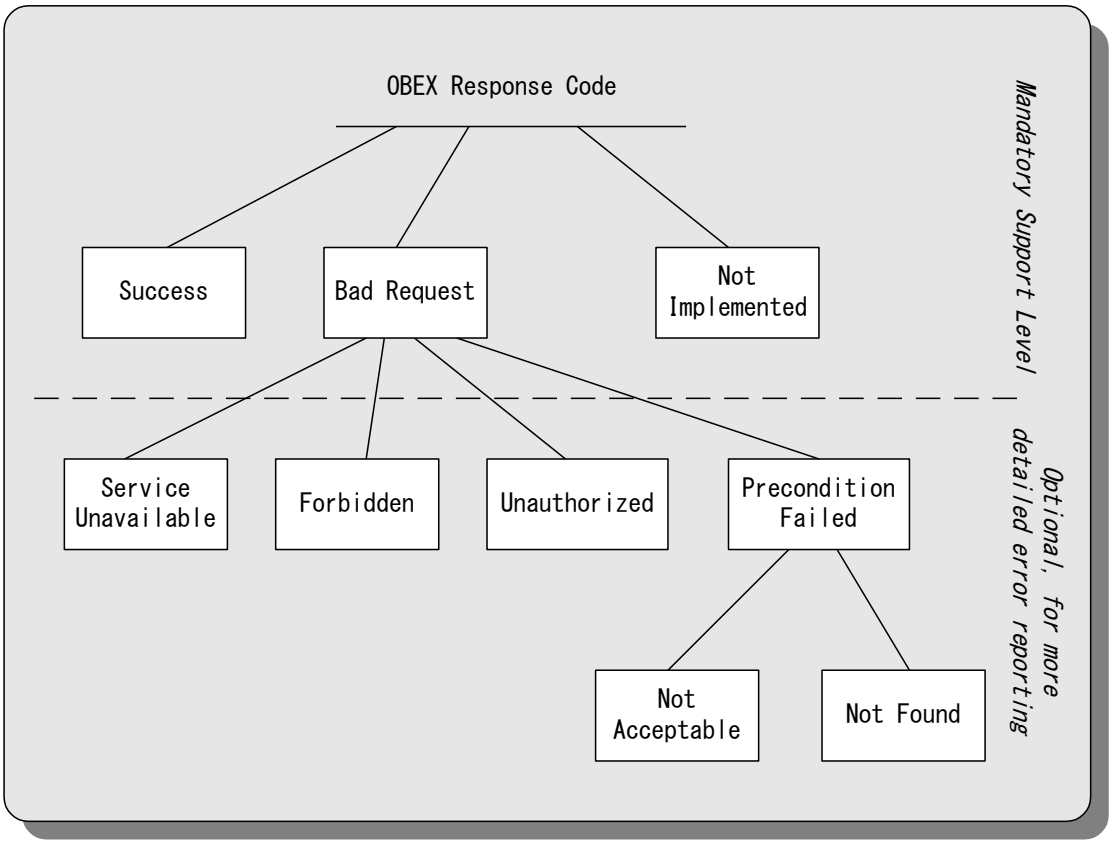


Figure 5.3: Logical Relationship between Basic Imaging OBEX Response Codes

When multi-packet responses are used, response codes must be returned as early as possible, preferably in the first response packet. In some cases – for example, Service Unavailable – it is possible that an error condition won't arise until the operation is underway, in which case it is acceptable to return a response code in a packet other than the first one.

5.4 Initializing OBEX

The initialization procedure is defined in [6].

Support for OBEX authentication is mandatory, including support for OBEX user IDs as described in [4]. Whether or not it is actually used is left to the implementer's discretion.

Note that if a device initiates OBEX authentication, interoperability cannot be guaranteed with devices that lack a user interface. Therefore it is recommended that OBEX authentication be turned off.

5.5 Establishing an OBEX Connection

See Section 5.4.1 in [7] for a description of OBEX connection establishment without authentication.

The use of the Target header is mandatory in the Basic Imaging Profile.

5.5.1 Primary and Secondary Connections

Some Basic Imaging Profile features require bi-directional communication between the Imaging Initiator and the Imaging Responder. For this reason two related OBEX connections are established.

The establishment of a secondary connection does not imply switching of the Imaging Initiator and Imaging Responder roles. The Imaging Initiator is always an OBEX client in the primary connection and an OBEX server in the secondary connection; the Imaging Responder is always an OBEX server in the primary connection and an OBEX client in the secondary connection.

5.5.2 Primary Session Establishment

Primary connections are OBEX connections established using a Target header with the Bluetooth Basic Imaging UUID value corresponding to the feature that is initiated. All other OBEX connections are referred to as secondary connections.

Basic Imaging primary connections	UUID
Basic Imaging Image Push	E33D9545-8374-4AD7-9EC5-C16BE31EDE8E
Basic Imaging Image Pull	8EE9B3D0-4608-11D5-841A-0002A5325B4E
Basic Imaging Advanced Image Printing	92353350-4608-11D5-841A-0002A5325B4E
Basic Imaging Automatic Archive	940126C0-4608-11D5-841A-0002A5325B4E
Basic Imaging Remote Camera	947E7420-4608-11D5-841A-0002A5325B4E
Basic Imaging Remote Display	94C7CD20-4608-11D5-841A-0002A5325B4E

Table 5.5: Bluetooth Basic primary connection UUIDs

5.5.3 Secondary Session Establishment

Secondary connections are OBEX connections established using a Target header with either of the following values:

Basic Imaging secondary connection	UUID
Basic Imaging Referenced Objects	8E61F95D-1A79-11D4-8EA4-00805F9B9834
Basic Imaging Archived Objects	8E61F95E-1A79-11D4-8EA4-00805F9B9834

Table 5.6: Bluetooth Basic Secondary Session UUIDs

Secondary connection can only be initiated and maintained while there is a primary connection in place. A secondary connection is initiated by the server of the primary connection. The client of the primary connection can optionally monitor the secondary connection by issuing GetStatus requests (see Section 4.5.15). Note that the secondary connection can be terminated by the client of the primary connection at any time; for that purpose, the client of the primary connection can either issue an Abort (if the GetStatus function is used) or issue a Disconnect for the primary connection.

5.6 Disconnecting

See Section 5.12 in [7].

If for some reason the primary OBEX connection is aborted or disconnected, the secondary connection must be aborted and disconnected as well. A secondary connection cannot exist outside the primary session to which it is related.

5.7 Single Response Mode

When transferring “large” objects and OBEX over L2CAP Single response mode (SRM) should be used. The definition of “large” is implementation-dependent. The Server should use the OBEX Length header to deduce the size of the object being transferred.

5.8 Reliable Sessions

OBEX Reliable Sessions shall not be used within this profile.

6 Service Discovery

6.1 Service Discovery Service Records

6.1.1 Imaging Responder Service

The SDP service records for the Imaging service are defined in Table 6.1, Table 6.2, and Table 6.3.

If a Basic Imaging Profile implementation can offer several outcomes as result of an Image Push operation, there shall be as many service discovery records for that implementation as there are possible outcomes, each service discovery record having bit 0 of the Supported Features attribute set to 0 and one and only one of bits 1, 2, or 3 set. This allows an Imaging Initiator to choose the outcome of an Image Push operation by opening the L2CAP protocol service multiplexer (PSM) or RFCOMM channel indicated in the relevant service discovery record.

If a Basic Imaging Profile implementation does not need or does not wish to specify the outcome of an Image Push operation, it provides one and only one instance of the service discovery record, in which bit 0 of the Supported Features attribute is set to 1 and bits 1, 2, and 3 are set to 0. For example, an implementation on which the outcome of the Image Push operations can be controlled from the user interface on an operation per operation basis would probably choose not to indicate any specific outcome in its service discovery record.

The Service Name attribute is used to further refine the nature of the service available by providing a human readable name. It is especially useful in case of multiple instances of service discovery record.

As an example of combined usage of the Supported Features and Service Name attributes for differentiating multiple instances of the service discovery record, let’s consider a hypothetical printer that also has image storage capability. This printer would expose two imaging service records: the first record would set the Service Name attribute to “storage” and the ImagePush-Store bit in the Supported Features attribute to 1, while the second would set the Service Name attribute to “print” and the ImagePush-Print bit in the Supported Features attribute to 1. By connecting to the L2CAP PSM or RFCOMM channel number listed in the first record, a client device would specify that images it sends to the printer should be stored. By connecting to the L2CAP PSM or RFCOMM channel number listed in the second record, the client device would indicate that images it pushed to the printer should be printed.

The Supported Functions attribute is a straightforward declaration of the functions that are supported by the implementation, among all the functions that are defined in Section 4.5.

The Total imaging data capacity attribute is used to indicate the maximum amount of imaging data in bytes that the implementation is capable of storing.

Item	Definition:	Type/ Size:	Value:	AttrID	Status	Default
Service class ID list				See [9]	M	
Service class #0		UUID	Imaging Responder		M	
Protocol descriptor list				See [9]	M	
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	Channel number	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	
Service name	Displayable text name	String	Service-provider defined	See [9]	O	“Imaging”
Bluetooth profile Descriptor list				See [9]	M	
Profile ID #0	Supported profile	UUID	Imaging			Imaging
Param #0	Profile version	Unit16	0x0102	See [9]		0x0102
Supported capabilities	Imaging capabilities flags	Uint8	Bit 0 = Generic imaging Bit 1 = Capturing Bit 2 = Printing Bit 3 = Displaying Bit 4..7 = *Reserved*	See [9]	M	0x00
Supported features	Imaging features flags	Uint16	Bit 0 = ImagePush Bit 1 = ImagePush-Store Bit 2 = ImagePush-Print Bit 3 = ImagePush-Display Bit 4 = ImagePull Bit 5 = AdvancedImagePrinting Bit 6 = AutomaticArchive Bit 7 = RemoteCamera Bit 8 = RemoteDisplay Bit 9..15 = *Reserved*	See [9]	M	

Item	Definition:	Type/ Size:	Value:	AttrID	Status	Default
Supported functions	Imaging functions flags	Uint32	Bit 0 = GetCapabilities Bit 1 = PutImage Bit 2 = PutLinkedAttachment Bit 3 = PutLinkedThumbnail Bit 4 = RemoteDisplay Bit 5 = GetImagesList Bit 6 = GetImageProperties Bit 7 = GetImage Bit 8 = GetLinkedThumbnail Bit 9 = GetLinkedAttachment Bit 10 = DeleteImage Bit 11 = StartPrint Bit 12 = *Reserved* Bit 13 = StartArchive Bit 14= GetMonitoringImage Bit 16 = GetStatus Bit 15, 17...31 = *Reserved*	See [9]	M	
Total imaging data capacity	Maximum memory available for image storage	Uint64	Memory in bytes	See [9]	M	
GoepL2capPsm	L2CAP PSM	Uint16	Varies	See [9]	M	

Table 6.1: Imaging Service Record

6.1.2 Referenced Objects Service

Item	Definition:	Type/ Size:	Value:	AttrID	Status	Default
Service class ID list				See [9]	M	
Service class #0		UUID	Imaging Referenced Objects		M	
Protocol descriptor list				See [9]	M	
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	Channel number	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	
Service name	Displayable text name	String	Service-provider defined	See [9]	O	“Imaging Referenced Objects”

Item	Definition:	Type/ Size:	Value:	AttrID	Status	Default
ServiceID*		UUID	Varies	See [9]	M	
Bluetooth profile Descriptor list				See [9]	M	
Profile ID #0	Supported profile	UUID	Imaging			Imaging
Param #0	Profile version	Uint16	0x0102	See [9]		0x0102
Supported Functions	Imaging functions flags	Uint32	Bit 0 = GetCapabilities Bit 1..11 = *Reserved* Bit 12 = GetPartialImage Bit 13..31 = *Reserved*	See [9]	M	
GoepL2CapPsm	L2CAP PSM	Uint16	Varies	See [9]	M	

Table 6.2: Referenced Objects Service Record

*The ServiceID attribute is defined in [1].

6.1.3 Archived Objects Service

Item	Definition:	Type/ Size:	Value:	AttrID	Status	Default
Service class ID list				See [9]	M	
Service class #0		UUID	Imaging Automatic Archive		M	
Protocol descriptor list				See [9]	M	
Protocol ID #0		UUID	L2CAP		M	
Protocol ID #1		UUID	RFCOMM		M	
Param #0	Channel number	Uint8	Varies		M	
Protocol ID #2		UUID	OBEX		M	
Service name	Displayable text name	String	Service-provider defined	See [9]	O	“Imaging Automatic Archive”
ServiceID*		UUID	Varies	See [9]	M	
Bluetooth profile Descriptor list				See [9]	M	
Profile ID #0	Supported profile	UUID	Imaging			Imaging
Param #0	Profile version	Unit16	0x0102	See [9]		0x0102
Supported functions	Imaging functions flags	Uint32	Bit 0 = GetCapabilities Bit 1..4 = Reserved Bit 5 = GetImagesList Bit 6 = GetImageProperty Bit 7 = GetImage Bit 8 = GetLinkedThumbnail Bit 9 = GetLinkedAttachment Bit 10 = DeleteImage Bit 11..31 = Reserved	See [9]	M	
GoepL2capPsm	L2CAP PSM	Uint16	Varies	See [9]	M	

Table 6.3: Automatic Archive Service Record

*The ServiceID attribute is defined in [1].

6.2 Service Discovery Procedure

The GOEP selection procedure defined in GOEP v2 [7] shall be used to determine whether to connect using OBEX over L2CAP or OBEX over RFCOMM.

The Imaging Initiator is required to perform the service discovery procedure as specified in GOEP [6].

All Imaging Initiators should check the Imaging Responder's SDP database to learn whether the features required by the Initiator are supported. The Imaging Initiator may use a Basic Imaging Profile feature if and only if the SDP database of the Imaging Responder advertises support for that feature.

The Imaging Initiator should also learn which optional functions are supported by the Imaging Responder. Only functions which are advertised as supported by the Imaging Responder should be invoked by the Imaging Initiator.

Note that the Imaging Initiator should be able to handle the case where there are multiple Imaging service records in one Imaging Responder SDP database. Imaging devices are encouraged to create multiple service records if they offer more than one service as an object exchange server. The different services can be identified by the service name and class of service fields in the SDP service record.

If an Imaging Initiator wants to connect to a specific OBEX service, it should learn where this service is located by selecting the appropriate SDP service record before establishing the connection. The Imaging Initiator should use the OBEX session UUID to establish an OBEX connection with the Imaging Responder on top of the established Bluetooth connection.

7 GOEP Interoperability Requirements

This section defines the requirements to interoperate with different versions of GOEP.

Client devices implementing this profile shall implement GOEP v2.0 or later, and follow the GOEP SDP Interoperability Requirements to determine whether the Imaging Responder device supports GOEP over L2CAP or only GOEP over RFCOMM.

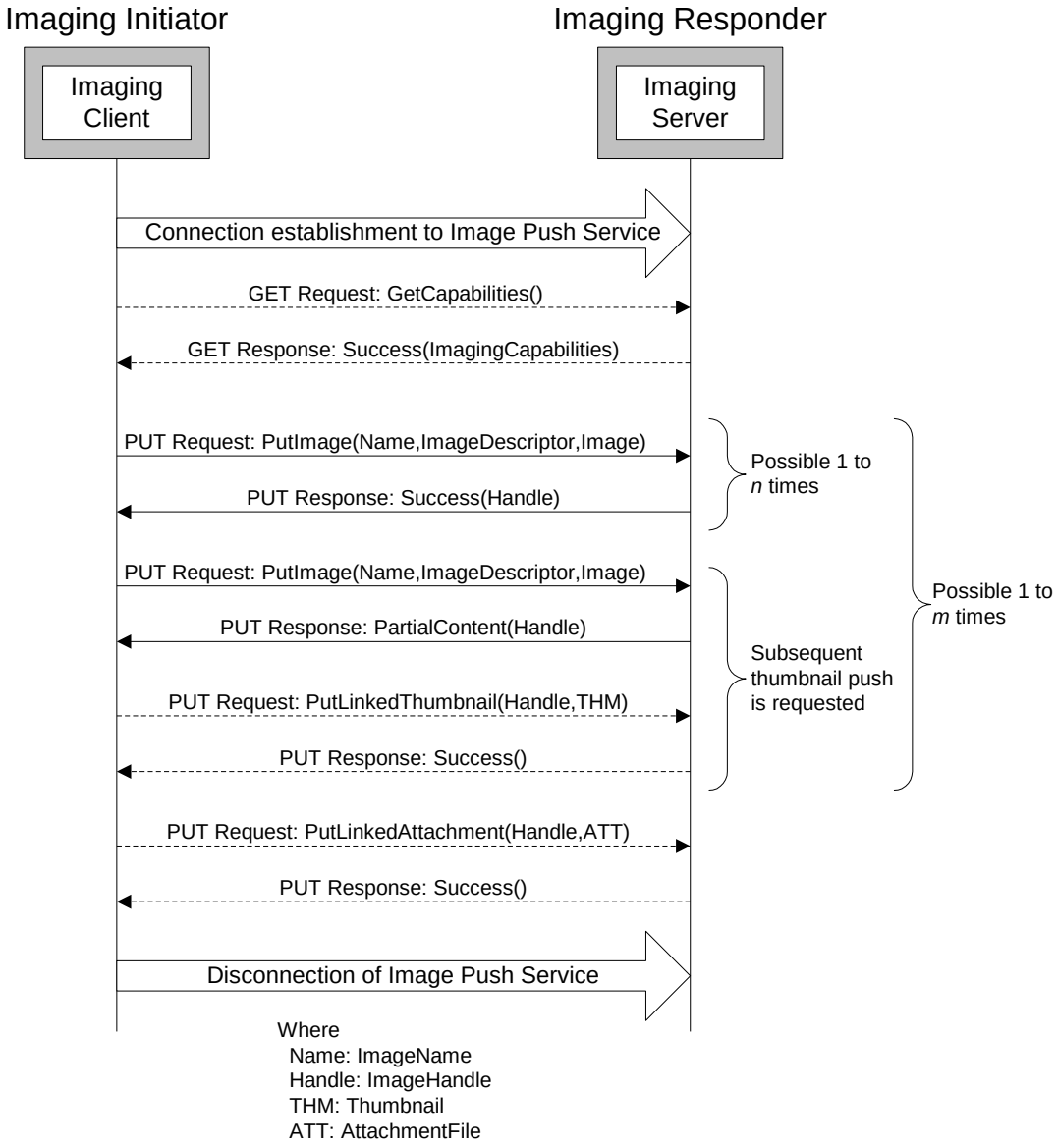
The following table shows which IrOBEX version will be used between devices implementing different versions of this profile:

Responder	v1.1 or later	Earlier than v1.1
Initiator		
v1.1 or later	IrOBEX v1.5	IrOBEX v1.2
Earlier than v1.1	IrOBEX v1.2	IrOBEX v1.2

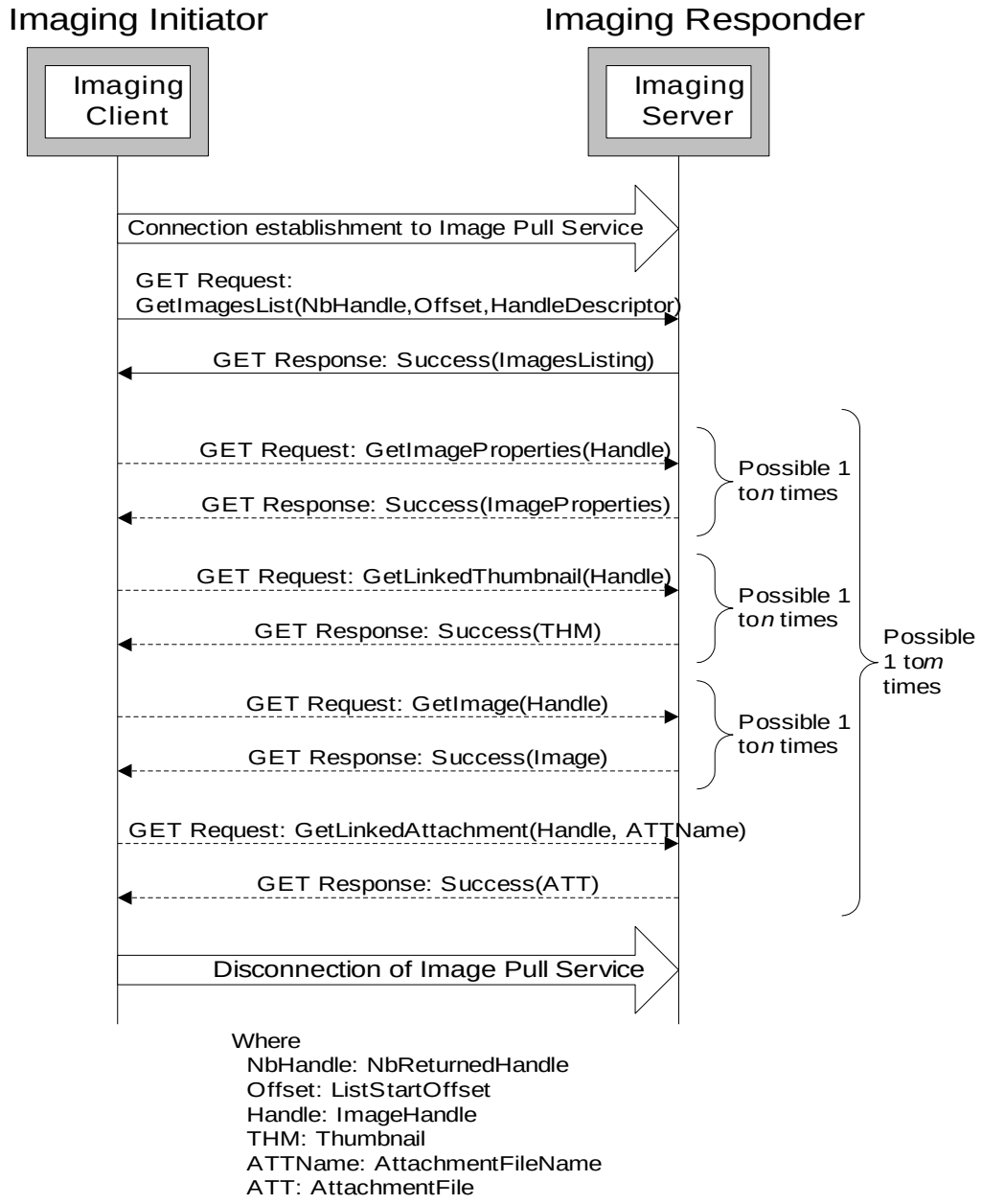
Features from IrOBEX later than v1.2 shall not be used if the OBEX connection is established over RFCOMM.

8 Annex A: Typical Message Sequence Charts

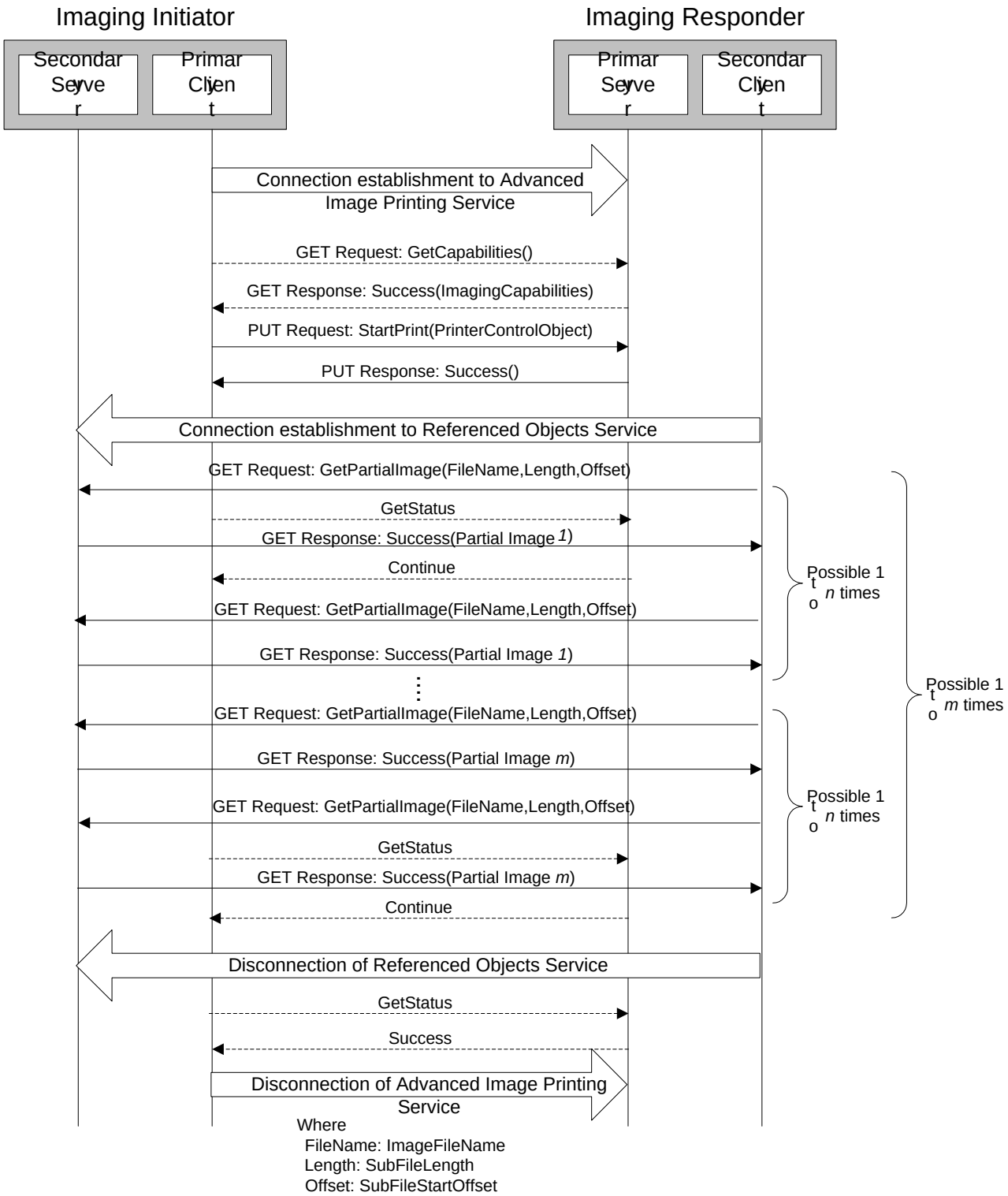
8.1 Image Push Feature



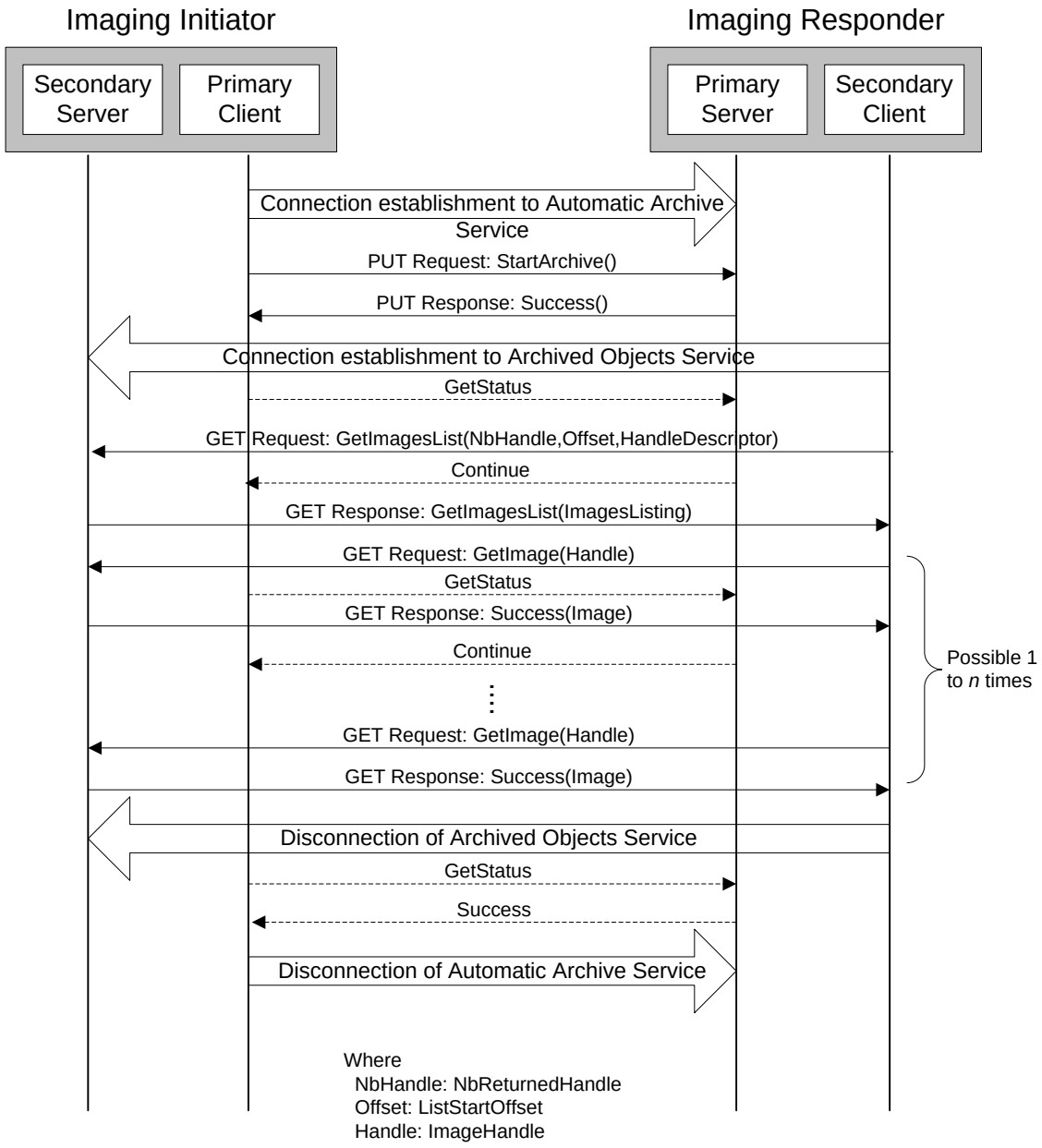
8.2 Image Pull Feature



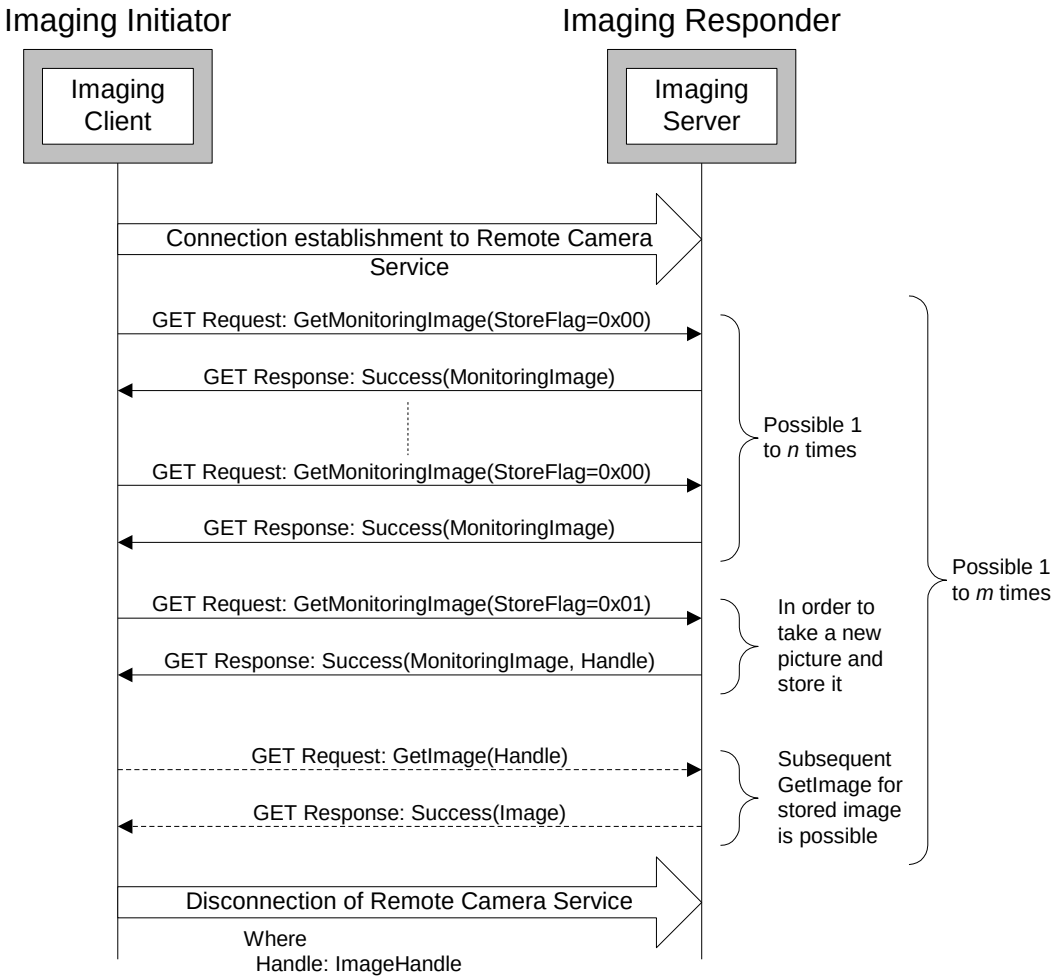
8.3 Advanced Image Printing Feature



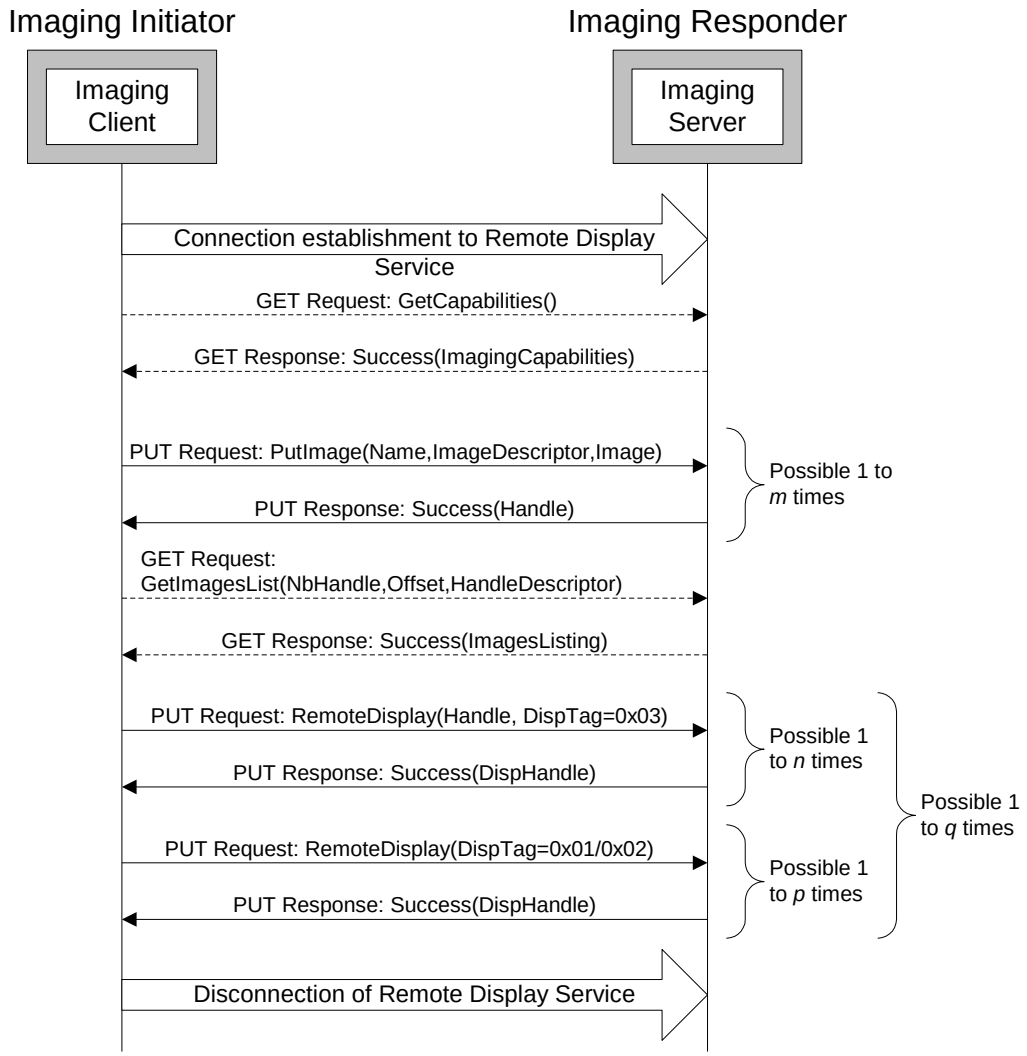
8.4 Automatic Archive Feature



8.5 Remote Camera Feature



8.6 Remote Display



Where
NbHandle: NbReturnedHandle
Offset: ListStartOffset
Handle: ImageHandle
DispTag: RemoteDisplay tag in Application Parameter header
DispHandle: DisplayedImageHandle

9 Annex B: Implementation Guidelines for DCF Devices

- It is recommended that DCF devices use the handle creation policy outlined in this annex.
- The file storage structure on DCF devices is illustrated in Figure B-1.

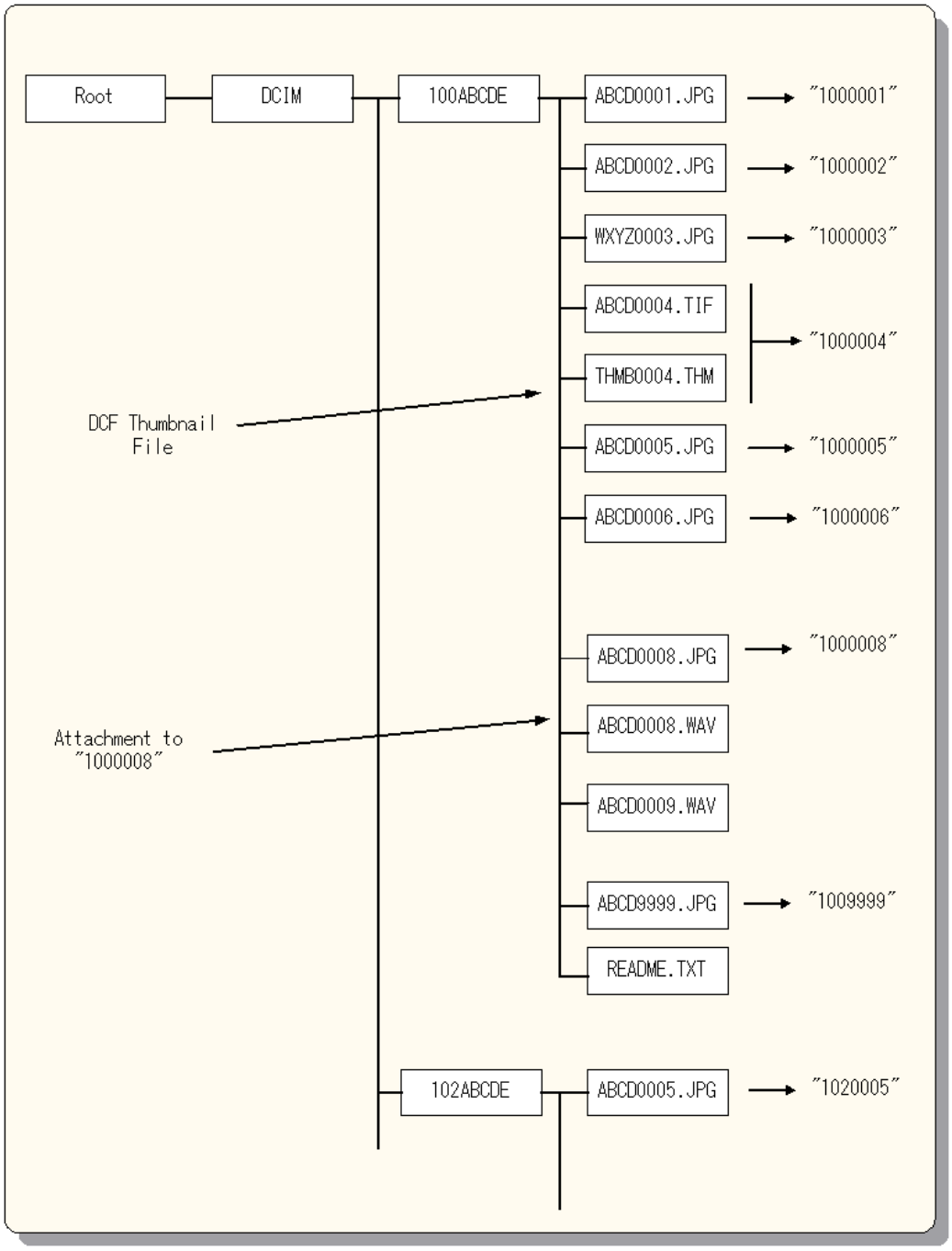


Figure 9.1: File Storage Structure on DCF Devices

- Folder names under the DCIM DCF main folder are formed from three digits plus five characters. The five characters can be chosen arbitrarily, but it is imperative that

each folder has three unique digits. Similarly, file names within DCF folders are composed of four arbitrarily chosen characters plus four digits that must be unique to an image – all files related to that image must have the four digits in their names.

- It is recommended that handles on such devices be produced by combining the three digits from the folder name with the four digits from the file name in this format: FFFffff, where FFF represents the three digits characteristic of the folder and ffff represents the four digits specific to an image.

10 Annex C: Synopsis of the OBEX Frame Structures in the Basic Imaging Profile, Phase 1

REQUESTS	Field		Header	
	Opcode	Packed length	Connection ID	Type
GetCapabilities	GET			x-bt/img-capabilities
GetImageList	GET			x-bt/img-listing
GetImageProperties	GET			x-bt/img-properties
GetImage	GET			x-bt/img-img
GetLinkedThumbnail	GET			x-bt/img-thm
GetLinkedAttachment	GET			x-bt/img-attachment
GetPartialImage	GET			x-bt/img-partial
GetMonitoringImage	GET			x-bt/img-monitoring
GetStatus	GET			x-bt/img-status
PutImage	PUT			x-bt/img-img
PutLinkedThumbnail	PUT			x-bt/img-thm
PutLinkedAttachment	PUT			x-bt/img-attachment
RemoteDisplay	PUT			x-bt/img-display
DeleteImage	PUT			x-bt/img-img
StartPrint	PUT			x-bt/img-print
StartArchive	PUT			x-bt/img-archive

RESPONSES	Field		Header	
	Opcode	Packed length	Connection ID	Type
GetCapabilities	GET-Response			
GetImageList	GET-Response			
GetImageProperties	GET-Response			
GetImage	GET-Response			
GetLinkedThumbnail	GET-Response			
GetLinkedAttachment	GET-Response			
GetPartialImage	GET-Response			
GetMonitoringImage	GET-Response			
GetStatus	GET-Response			
PutImage	PUT-Response			
PutLinkedThumbnail	PUT-Response			
PutLinkedAttachment	PUT-Response			
RemoteDisplay	PUT-Response			
DeleteImage	PUT-Response			
StartPrint	PUT-Response			
StartArchive	PUT-Response			

REQUESTS	Name	Img-Description	Img-Handle
GetCapabilities			
GetImageList		XML—HandlesDescriptor	
GetImageProperties			ImageHandle
GetImage		XML—ImageDescriptor	ImageHandle
GetLinkedThumbnail			ImageHandle
GetLinkedAttachment	“Name”		ImageHandle
GetPartialImage	“Filename”		
GetMonitoringImage			
GetStatus			
PutImage	“Filename”	XML—Image Descriptor	
PutLinkedThumbnail			ImageHandle
PutLinkedAttachment	“Name”	XML—AttachmentDescriptor	ImageHandle
RemoteDisplay			ImageHandle
DeleteImage			ImageHandle
StartPrint			
StartArchive			

RESPONSES	Name	Img-Description	Img-Handle
GetCapabilities			
GetImageList		XML—HandlesDescriptor	
GetImageProperties			
GetImage			
GetLinkedThumbnail			
GetLinkedAttachment			
GetPartialImage			
GetMonitoringImage			ImageHandle
GetStatus			
PutImage			ImageHandle
PutLinkedThumbnail			
PutLinkedAttachment			
RemoteDisplay			ImageHandle
DeleteImage			
StartPrint			
StartArchive			

REQUESTS	Header Application Parameters							
	Remote Display	Nb Returned Handles	ListStart Offset	Latest Captured Image	Store Flag	Sub File Length	Sub File Start Offset	Total File Size
GetCapabilities								
GetImageList		X	X	X				
GetImageProperties								
GetImage								
GetLinkedThumbnail								
GetLinkedAttachment								
GetPartialImage						X	X	
GetMonitoringImage					X			
GetStatus								
PutImage								
PutLinkedThumbnail								
PutLinkedAttachment								
RemoteDisplay	X							
DeleteImage								
StartPrint								
StartArchive								

RESPONSES	Header Application Parameters							
	Remote Display	Nb Returned Handles	ListStart Offset	Latest Captured Image	Store Flag	Sub File Length	Sub File Start Offset	Total File Size
GetCapabilities								
GetImageList		X						
GetImageProperties								
GetImage								
GetLinkedThumbnail								
GetLinkedAttachment								
GetPartialImage								X
GetMonitoringImage								
GetStatus								
PutImage								
PutLinkedThumbnail								
PutLinkedAttachment								
RemoteDisplay								
DeleteImage								
StartPrint								
StartArchive								

11 Annex D: References

11.1 Normative References

[1] Bluetooth Core Specification v3.0 or later
[2] Specification of the Bluetooth System, RFCOMM with TS 07.10 Specification
[3] TS 101 369 (GSM 07.10) version 6.3.0: Digital cellular telecommunications system (Phase 2+) (GSM);Terminal Equipment to Mobile Station (TE-MS) multiplexer protocol
[4] IrOBEX Specification, Version 1.5, Infrared Data Association
[5] Specification of the Bluetooth System, Generic Object Exchange Profile
[6] Specification of the Bluetooth System, Generic Access Profile Specification
[7] Bluetooth Special Interest Group, Generic Object Exchange Profile v2.0
[8] IrDA Interoperability v2.0
[9] Bluetooth SIG Assigned Numbers – Service Discovery
[10] Digital Print Order Format, Version 1.10 http://panasonic.jp/dc/dpof_110/white_e.htm
[11] Extensible Markup Language (XML) 1.0 (Fifth Edition) W3C Recommendation 26 November 2008 (http://www.w3c.org/TR/REC-xml)
[12] ISO/IEC 10918-1 (JPEG) International Standard – Information Technology – Digital Compression and Coding of Continuous-tone Still Images: Requirements and Guidelines
[13] JEIDA Standard Design Rule for Camera File System, Version 1.0
[14] JEIDA Standard Digital Still Camera Image File Format Standard, Version 2.1
[15] Graphics Interchange Format, Version 89a (http://www.w3.org/Graphics/GIF/spec-gif89a.txt)
[16] WAP-190-WAESpec (Wireless Application Protocol, Wireless Application Environment Specification, Version 1.3)
[17] Portable Network Graphics (PNG) Specification (Second Edition) W3C Recommendation 10--November-2003 (http://www.w3c.org/TR/REC-png.html)
[18] BMP (Windows Bitmap) specification http://en.wikipedia.org/wiki/BMP_file_format
[19] The JPEG2000 specifications are currently awaiting publication as the ISO/IEC 15444 family of specifications. The latest drafts are available here.

12 Annex E: BIP over AMP (Informative)

This annex provides guidance to implementers when using this version of BIP in devices conforming to Bluetooth v3.0 + HS or later.

12.1 BIP using OBEX over L2CAP

When BIP is using OBEX over L2CAP devices should use an AMP when transferring “large” images. SRM should be used to fully utilize HS capabilities of the devices.

The implementer of a BIP device should consider the use of an AMP when transferring images; the Initiator can initiate connection over AMP and both the Initiator and the Responder can initiate L2CAP Move to AMP procedure based on the size of the image file, or the Image Properties, or information obtained using the GetImage function.

12.2 BIP using OBEX over RFCOMM

When BIP is using OBEX over RFCOMM, the RFCOMM entity may also be transporting traffic from other profiles. These means the L2CAP Move procedure should only happen if all profiles running over RFCOMM are capable of running over AMP and accept a move onto another logical link. It is not recommended to perform the L2CAP Move procedure when using GOEP v1.1. Devices should use GOEP v2.0 with OBEX over L2CAP to obtain the benefit of the AMP when using this profile.

13 Annex F: Acronyms and Abbreviations

Acronym or Abbreviation	Meaning
AMP	Alternate MAC/PHY
BB	Baseband
CoD	Class of Device
DCF	Design rules for Camera File system
GOEP	Generic Object Exchange Profile
HCI	Host Controller Interface
HS	High Speed
L2CAP	Logical Link and Control Adaptation Protocol
LC	Link Controller
LM	Link Manager
LMP	Link Manager Protocol
MSC	Message Sequence Chart
OBEX	Object Exchange Protocol
PIM	Personal Information Management
PLMN	Public Land Mobile Network
PSM	Protocol Service Multiplexer
QoS	Quality of Service
RFCOMM	Serial Cable Emulation Protocol
SD	Service Discovery
SDP	Service Discovery Protocol
SDPDB	Service Discovery Protocol Database
UI	User Interface
EXIF	Exchangeable image file for digital still cameras
XML	Extensible Markup Language
DTD	Document Type Definition
MIME	Multipurpose Internet Mail Extension
UUID	Universal Unique Identifier
UTC	Coordinated Universal Time