MCPC TR-001 Ver.1.0
BASIC IMAGING PROFILE
テクニカル リファレンス
Ver.1.0
2003年9月26日

Mobile Computing Promotion Consortium

(MCPC)

変更履歴

版	日付	記載箇所	詳細
1.0	2003/09/26		Ver1.0 作成

目 次

概要

本文書は、Bluetooth® Special Interest Group(SIG)が規格する Basic Imaging Profile (BIP)仕様の内容を補足するものであり、BIP を実装する機器間の相互接続性の向上を目的とする。本文書では、Q&A 形式により推奨する実装方法およびその具体例を示している。

注意

本文書は、その内容を強制するものではなく、Bluetooth®の権利を侵害するものではない。またその利用に関しては、利用者の責任において実施されるものとする。MCPC は本文書に関する法的侵害、市場性、またあるいは特定の目的に対する整合性を含むいかなる明示的、または暗示的保証をしない。

参考規格／仕様

本文書にて参照している参考規格、関連仕様書を以下に示す。なお、断りが有る場合を除き、参照するバージョンは最新版とする。

参考文献
[1] BASIC IMAGING PROFILE Interoperability Specification
[2] IMPLEMENTATION GUIDELINES BASIC IMAGING PROFILE, Revision 1.0
[3] JEITA Standard Exchangeable image file format for digital still camera, Version 2.21
[4] JEITA Standard Design rule for Camera File system, Version 2.0
[5] ISO/IEC 10918-1 / ITU-T Recommendation T.81 information technology - Digital compression and coding of continuous-tone still images - Requirements and guide-lines
[6] IrOBEX Specification, Version 1.2, Infrared Data Association
[7] Specification of the Bluetooth System, Assigned Numbers Specification

略語･記号の定義

本文書で使用する略語･記号について以下に示す。

略語・記号	定義
CoD	Class of Device
DCF	Design rules for Camera File system
OBEX	Object Exchange Protocol
SDP	Service Discovery Protocol
Exif	Exchangeable image file for digital still cameras
XML	Extensible Markup Language
DTD	Document Type Definition

Q1. Capabilities オブジェクトにおける preferred-format の書式について

A.

preferred-format 要素は、Responder が受信可能な画像フォーマットのうちから最適な画像フォーマットを Initiator に伝えるために利用される。但し、オリジナル画像が指定された画像フォーマットに合致するか、Initiator が送信画像を指定された画像フォーマットに変換可能である場合にのみ有効である。この preferred-format で指定される画像フォーマットは、Q2 で後述の image-formats で記述された範囲内のものでなくてはならない。また、オリジナル画像を画像フォーマット変換されることなく受信したい Responder は preferred-format 要素を指定すべきでない。

Imaging-capabilities object の DTD では、preferred-format 要素の出現回数が"?"で指定されていることから、複数個の記述は不可能であり、必要のない場合は省略することが可能である。また、preferred-format 要素の ATTLIST 宣言によると、属性値として" encoding "、" pixel "、" transformation "、" maxsize " の４つが定義されている。このうち " encoding " は必須の属性であり、preferred-format 要素を記述する場合必ず記述しなければならない。一方、" pixel " は省略可能であるが、相互接続性を高めるためにはできるだけ記述することを推奨する。その際、" pixel " については固定値による記述（例："640*480"）または範囲指定による記述（例："1*1-1600*1200"）ができるが、固定値による記述を推奨する。以下は、Imaging-capabilities object の DTD より preferred-format 要素に関する ELEMENT 宣言と ATTLIST 宣言を抜粋したものである。

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

次に、preferred-format 要素の記述例を示す。

- preferred-format 要素の記述例１

  Responder はいかなるピクセルサイズの JPEG 画像および BMP 画像を受信可能であるが、ピクセルサイズが 320*240 の JPEG 画像を優先して受信したい場合

```
<imaging-capabilities version="1.0">
<preferred-format encoding="JPEG" pixel="320*240"/>
<image-formats encoding="JPEG" pixel="1*1-65535*65535"/>
<image-formats encoding="BMP" pixel=" 1*1-65535*65535"/>
</imaging-capabilities>
```

- preferred-format 要素の記述例２

  Responder はピクセルサイズが 160*120 と 640*480 の JPEG 画像を受信可能であるが、ピクセル

サイズが 640*480 **の** JPEG **画像を優先して受信したい場合**

<imaging-capabilities version=”1.0”>

<preferred-format encoding=”JPEG” pixel=”640*480”/>

<image-formats encoding=”JPEG” pixel=”160*120”/>

<image-formats encoding=”JPEG” pixel=”640*480”/>

</imaging-capabilities>

- preferred-format **要素の記述例３**

  Responder **はいかなるピクセルサイズの** JPEG **画像も受信可能であるが、ピクセルサイズが** 640*480 (VGA)**から** 1024*768 (XGA)**までの** JPEG **画像を優先して受信したい場合**

  <imaging-capabilities version=”1.0”>

  <preferred-format encoding=”JPEG” pixel=”640*480-1024*768”/>

  <image-formats encoding=”JPEG” pixel=”1*1-65535*65535”/>

  </imaging-capabilities>

- preferred-format **要素の記述例４**

  Responder **はいかなるピクセルサイズの** JPEG **画像も受信可能であるが、データサイズ** 50000 **バイト以下の** JPEG **画像を優先して受信したい場合**

  <imaging-capabilities version=”1.0”>

  <preferred-format encoding=”JPEG” maxsize=”50000”/>

  <image-formats encoding=”JPEG” pixel=”1*1-65535*65535”/>

  </imaging-capabilities>

- preferred-format **要素の記述例５**

  Responder **はいかなるピクセルサイズの** JPEG **画像および** BMP **画像を受信可能であるが、**JPEG **画像を優先して受信したい場合**

  <imaging-capabilities version=”1.0”>

  <preferred-format encoding=”JPEG”/>

  <image-formats encoding=”JPEG” pixel=” 1*1-65535*65535”/>

  <image-formats encoding=”BMP” pixel=” 1*1-65535*65535”/>

  </imaging-capabilities>

上記例に示したように、preferred-format **要素を記載する場合、“** pixel”**については固定値による記述（例：**”640*480）**または範囲指定による記述（例：**”1*1-1600*1200”）**ができる。しかし、**Initiator **が送信する画像のピクセルサイズを決定する処理負担を下げるために固定値による記述を推奨する。**

Q2. Capabilities オブジェクトにおける image-formats の書式について

A.

image-formats 要素は、Responder が受信可能な画像フォーマットを Initiator に伝えるために利用される。

Imaging-capabilities object の DTD では、image-formats 要素の出現回数が"*"で指定されていることから、複数個の記述が可能である。(ImagePush の Responder として動作しない場合などは省略可能である。)また、image-formats 要素の ATTLIST 宣言によると、属性値として" encoding "、" pixel "、" maxsize " の 3 つが定義されている。このうち " encoding " は必須の属性であり、必ず記述しなければならない。一方、" pixel "は省略可能であるが、相互接続性を高めるためにはできるだけ記述することを推奨する。" pixel " を省略した場合は、受信ピクセルサイズに制限がないものと扱われるものとする。以下は Imaging-capabilities object の DTD より image-formats 要素に関する ELEMENT 宣言と ATTLIST 宣言を抜粋したものである。

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

次に、image-formats 要素の記述例を示す。

- image-formats 要素の記述例１
  Responder がいかなるピクセルサイズの JPEG 画像も受信可能である場合
  <imaging-capabilities version="1.0">
  <image-formats encoding="JPEG" pixel="1*1-65535*65535"/>
  </imaging-capabilities>

- image-formats 要素の記述例２
  Responder が 1*1 から 1600*1200 までのピクセルサイズの JPEG 画像を受信可能である場合
  <imaging-capabilities version="1.0">
  <image-formats encoding="JPEG" pixel="1*1-1600*1200"/>
  </imaging-capabilities>

- image-formats 要素の記述例３
  Responder が 160*120、320*240、640*480、1280*960 のピクセルサイズの JPEG 画像のみ受信可能である場合
  <imaging-capabilities version="1.0">
  <image-formats encoding="JPEG" pixel="160*120"/>

<image-formats encoding=”JPEG” pixel=”320*240”/>
<image-formats encoding=”JPEG” pixel=”640*480”/>
<image-formats encoding=”JPEG” pixel=”1280*960”/>
</imaging-capabilities>

・ image-formats **要素の記述例４**
Responder **が** 160*120 **から** 320*240**、**640*480**、**1280*960 **のピクセルサイズの** JPEG **画像のみ受信可能である場合**
<imaging-capabilities version=”1.0”>
<image-formats encoding=”JPEG” pixel=”160*120-320*240”/>
<image-formats encoding=”JPEG” pixel=”640*480”/>
<image-formats encoding=”JPEG” pixel=”1280*960”/>
</imaging-capabilities>

上記例に示したように、Image-formats **要素を記載する場合、**Responder **の受信能力により受信可能ピクセルサイズを範囲指定（例：**”1*1-1600*1200”**）または固定値（例：**”640*480”**）で記述することができる。しかし、ピクセルサイズを固定値で記述した場合、**Initiator **がほぼ近い値にリサイズできるが数ピクセル一致しないため送信できないという状況が発生する可能性がある。そのため、相互接続性を高めるためには** Responder **の受信能力に応じて可能な限り受信可能ピクセルサイズを範囲指定で記述することを推奨する。**

Q3. Capabilities オブジェクトの解析手順例、アルゴリズムについて

A.

ImagePush の Initiator として動作する場合の Image Capabilities Object 解析手順の一例を以下に示す。

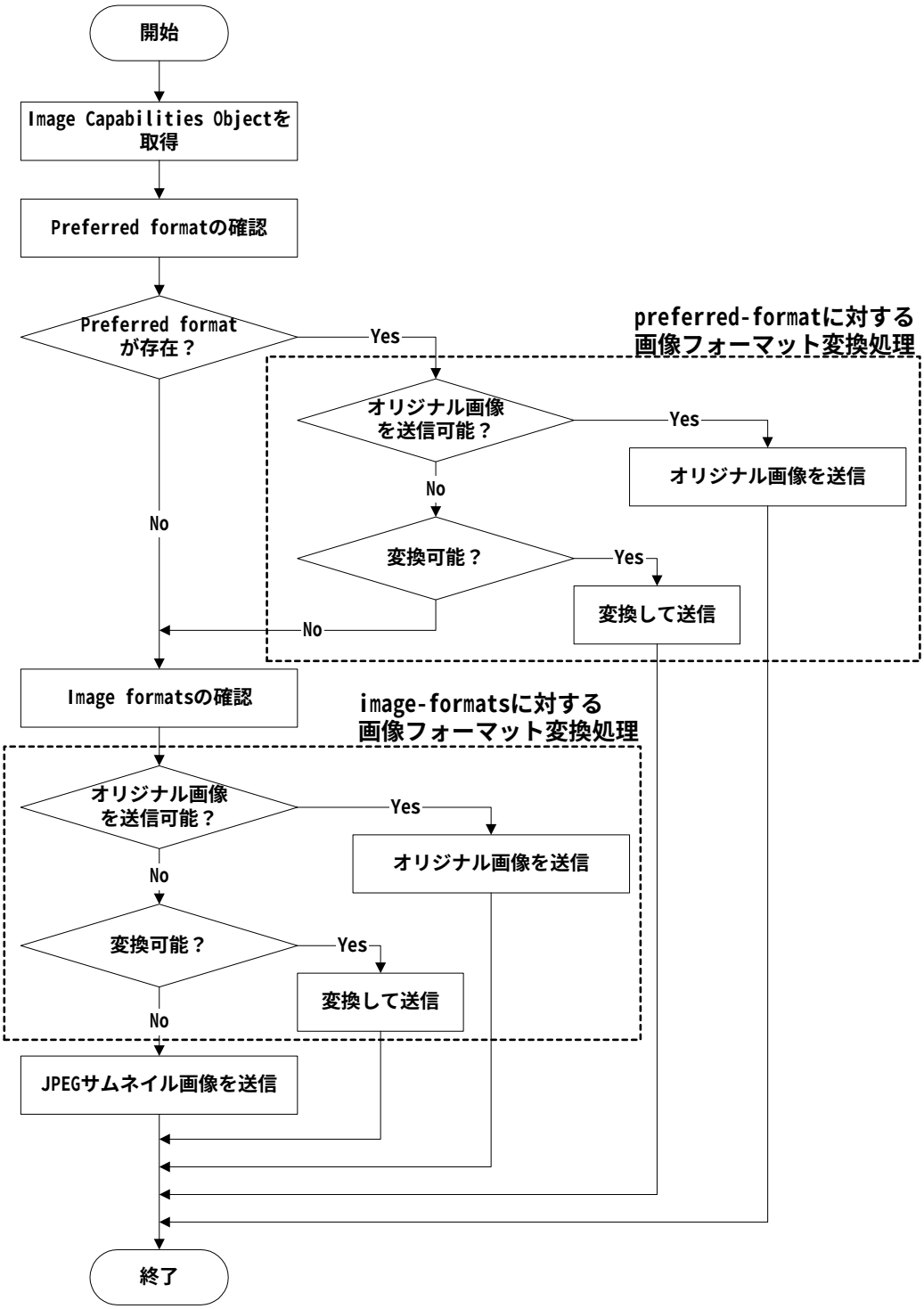


図１　推奨する Image Capabilities Object **解析手順**

preferred-format に対する画像フォーマット変換処理について

preferred-format に記載されている内容を解析し、オリジナル画像を送信可能であれば、変換することなく送信する。preferred-format の内容解析の結果、オリジナル画像の送信が不可であれば、変換処理を行い変換した画像を送信する。なお、変換処理が不可の場合は、preferred-format に対する画像フォーマット変換処理を終了する。

preferred-format でピクセルサイズが固定値で記述されている場合、リサイズ可能であれば必ず Responder が指定したピクセルサイズにリサイズすることを推奨する。preferred-format でピクセルサイズが範囲指定により記述されている場合で、オリジナル画像がその範囲指定の上限より大きい場合は上限値として指定されているピクセルサイズにあわせることを推奨する。preferred-format でピクセルサイズが範囲指定により記述されている場合で、オリジナル画像がその範囲指定の下限より小さい場合は下限値として指定されているピクセルサイズにあわせることを推奨する。また、Responder が指定するピクセルサイズに縦横比を維持したまま縮小リサイズする際、小数点処理などの影響により正確に指定ピクセルサイズに合わせられない場合は、必要に応じて fill、crop、stretch 等の変換処理を利用してあわせることを推奨する。一方、preferred-format でピクセルサイズが記述されていない場合、Responder が受信可能なピクセルサイズは image-formats に記述されていることから（Q1,Q2 参照）、リサイズするピクセルサイズについては image-formats の値を参照して決定する。

image-formats に対する画像フォーマット変換処理について

image-formats に記載されている内容を解析し、オリジナル画像を送信可能であれば、変換することなく送信する。image-formats の内容解析の結果、オリジナル画像の送信が不可であれば、変換処理を行い変換した画像を送信する。なお、変換処理が不可の場合は、image-formats に対する画像フォーマット変換処理を終了する。

リサイズする場合は、オリジナル画像のピクセルサイズより大きなピクセルサイズにリサイズすることはできるかぎり避けるべきである。オリジナル画像のピクセルサイズより小さいピクセルサイズで、かつ Responder より指定される image-formats の範囲指定ピクセルサイズ内で可能な限り大きなピクセルサイズとすることを推奨する。また、Responder が指定するピクセルサイズに縦横比を維持したまま縮小リサイズする際、小数点処理などの影響により正確に指定ピクセルサイズに合わせられない場合は、必要に応じて fill、crop、stretch 等の変換処理を利用してあわせることを推奨する。

Q4. ImagePush 時の Image Descriptor の image 要素の記述について

A.

ImagePush で画像を送信する際、Image Descriptor の image 要素に送信する画像のプロパティを記述することが必須である。Image Descriptor は以下のとおり定義されている。

```
<!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
]>
```

image 要素の ATTLIST 宣言によると、属性値として “ encoding ”、“ pixel ”、“ size ”、“ maxsize ”、“ transformation ” の５つが定義されている。このうち “ encoding ”、“ pixel ” は必須の属性であり、必ず記述しなければならない。一方、“ size ” は省略可能であるが、相互接続性を高めるためにはできるだけ記述することを推奨する。

Initiator が Image Descriptor の size を用いて、Responder に画像サイズを通知することで、Responder は画像受信開始時に画像サイズを確認することが可能となる。これにより、例えば、Responder が受信バッファのサイズと比較して画像のサイズが大きい場合には直ぐにエラーを示すレスポンスコードを返し処理を終了することが可能となる。

Q5. Image Push の Responder から Initiator に対するデータ送信の抑制について

A.

Image Push を使用した場合、通信シーケンスの途中で Responder において受信データを保存するための十分な記憶領域が無くなったとき等に、Responder が Initiator からのデータの送信を一時的に抑制させたい状況が発生する。

具体例として、Initiator からプリンタに画像ファイルを送信するときの通信シーケンスを考える。このときに、Initiator から続けて複数枚の画像ファイルを送信する場合や、送信の途中でプリンタにおいて用紙切れやインク切れが発生した場合において、このような状況が発生する。このときの通信シーケンス例を以下に示す。

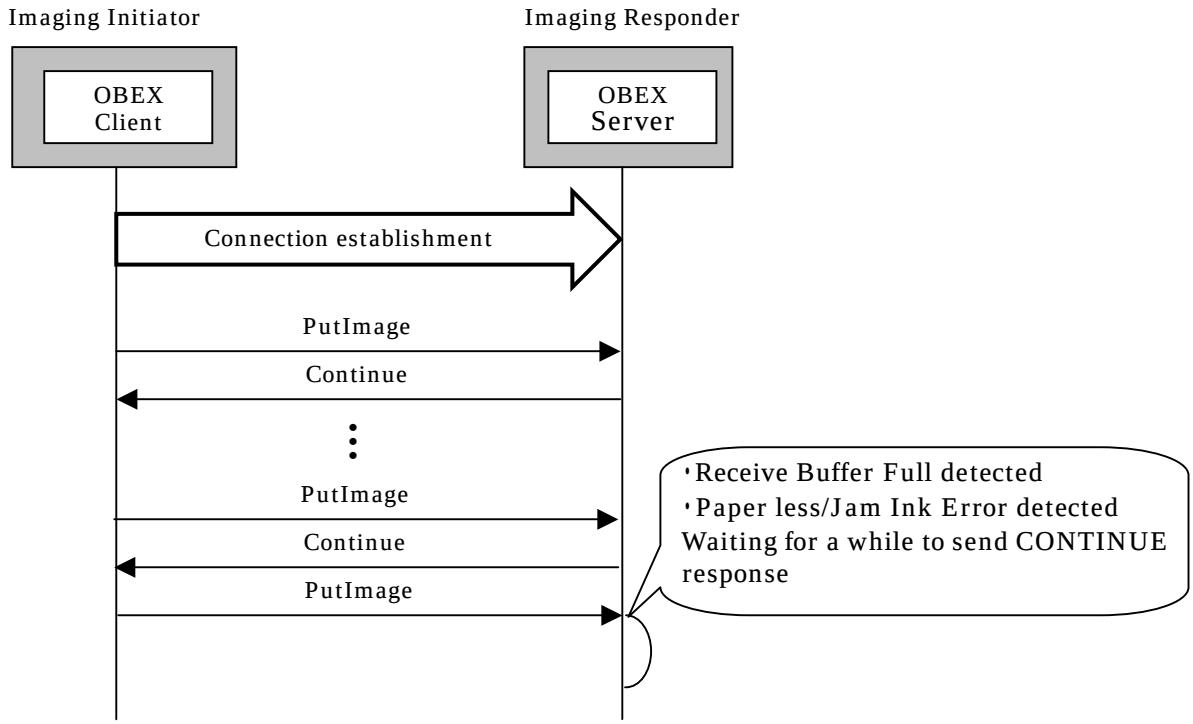


図２　ImagePush において Responder がレスポンスを保留する場合の通信シーケンス例

図２の通信シーケンス例における問題点を以下に示す。

【Responder（プリンタなど）における問題】

- Initiator がレスポンスの受信に対してタイマをかけている場合にタイムアップで切断されてしまい印刷が行えないことがある。
- Responder がエラーを示すレスポンスコード(ex. Not Acceptable/Service Unavailable)を返す実装を行うと Initiator が切断する可能性がある。

【Initiator への影響】

- Initiator が例えば画像送信中を表すプログレスバーを使用するアプリケーションであった場合、レスポンスがないことからプログレスバーの動き止まりアプリケーションがフリーズしたように見える。
- レスポンスがないことで Responder に修復不可能な問題が発生しているのか警告（修復可能な問題）

が発生しているのか区別ができない。

このようなケースでは以下の実装を推奨する。

- Responder は OBEX 層において Put に対するレスポンスを保留する実装を行う。
- Initiator はタイムアウトの時間を十分とるように実装を行う。
- Initiator はタイムアップ時にコネクションを強制切断するものとする。

図３に Initiator におけるタイムアップ発生時の通信シーケンスを示す。

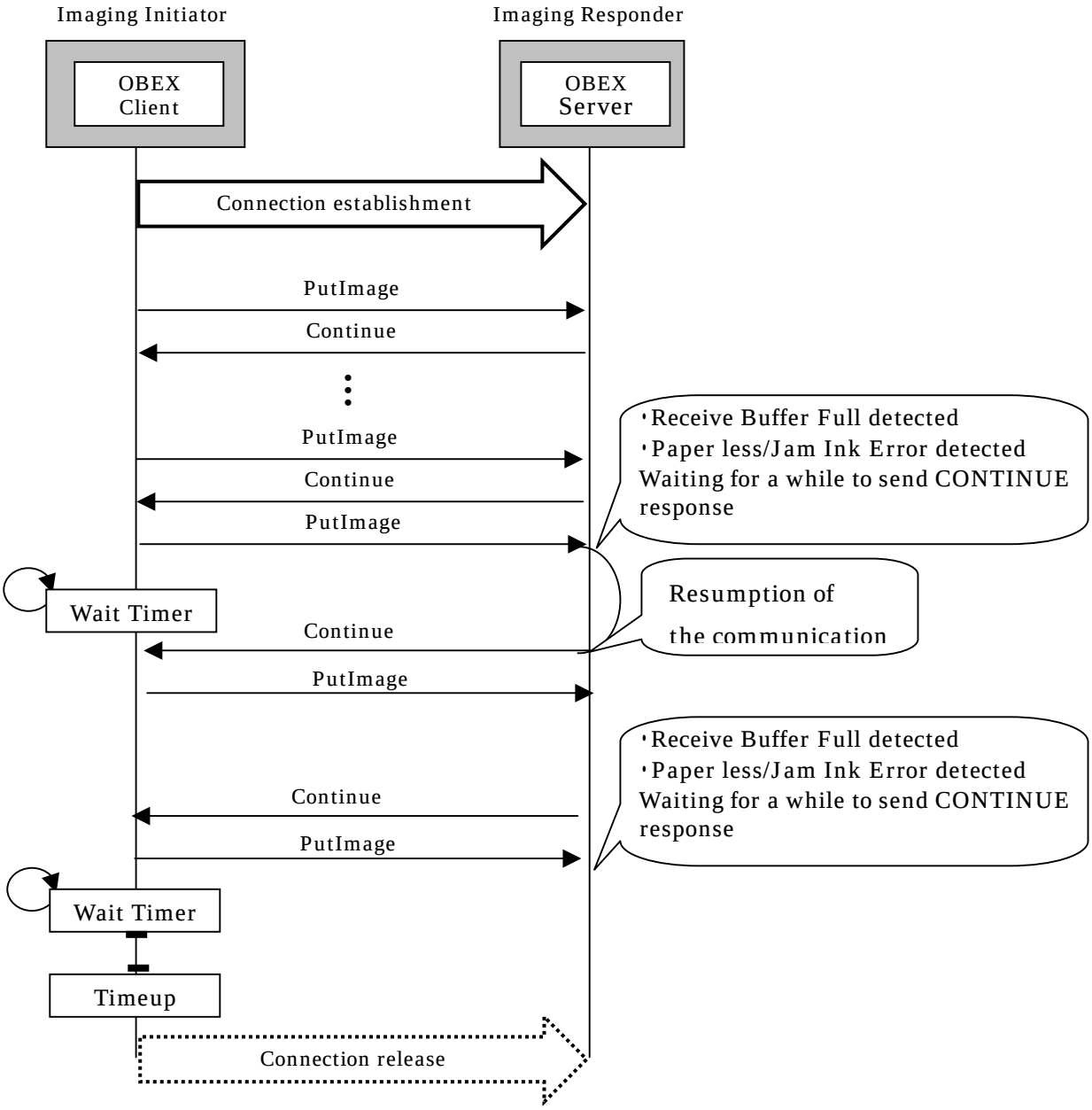


図３　Image Push における推奨実装を行った場合の通信シーケンス例

タイムアップ時間に関してはInitiatorの実装依存である。参考までに、プリンタに対し画像ファイルを送信する機能を備えるInitiatorにおけるタイムアップ時間の設定方針例を以下に示す。

例１） プリンタの用紙切れやインク切れが発生したときを考慮してタイムアップ時間を設定する場合

実際の利用状況に依存するが、ユーザがプリンタを復帰させるために十分な時間をタイムアップ時間として設定する必要がある。

例２） インクジェットプリンタに対して同一コネクションにて連続して印刷要求を行うことを考慮してタイムアップ時間を設定する場合

インクジェットプリンタが一枚の画像を印刷するために要する時間を考慮してタイムアップ時間として設定する必要がある。連続印刷を考慮した実装を行っているInitiatorは、プリンタでの用紙一枚の印刷に要する時間よりタイムアップ時間が短いときには、連続印刷中にタイムアップにより必ず送信失敗が発生する可能性（※）があることに注意しなければならない。連続印刷を期待するようなInitiatorでは、印刷時間を考慮してタイムアップを設定する必要がある。

例３） Initiator自体の電力消費量を最優先に考慮してタイムアップ時間を設定する場合

Initiator側のバッテリー容量を考慮してタイムアップ時間を設定する。

※：Q4やQ12に記載されているような実装をしているInitiatorおよびResponderの組み合わせで通信した場合は、このような問題が発生する可能性は低い。

Q6. GetStatus Request に対するレスポンスコードについて

A.

BIP[1]の 4.5.15 では GetStatus Request に対するレスポンスコードについて下記のように記述されている。

- Responder は、一度 secondary connection が確立された後は、Initiator からの GetStatus request に対して、secondary connection 接続中は“ Continue ”をレスポンスコードとして返し、切断後には“ OK, Success ”をレスポンスコードとして返す。
- Responder のエラーにより secondary connection に影響がある場合には、Initiator からの GetStatus request に対して、Responder はエラーを示すレスポンスコードを返す。

上記に加えて、相互接続性を高めるために、Responder は以下のように追加実装することを推奨する。

- Responder は、secondary connection が確立する前の Initiator からの GetStatus request に対して、“ Continue ”をレスポンスコードとして返す。

ここで推奨された追加実装をした場合の通信シーケンス例を図４に示す。

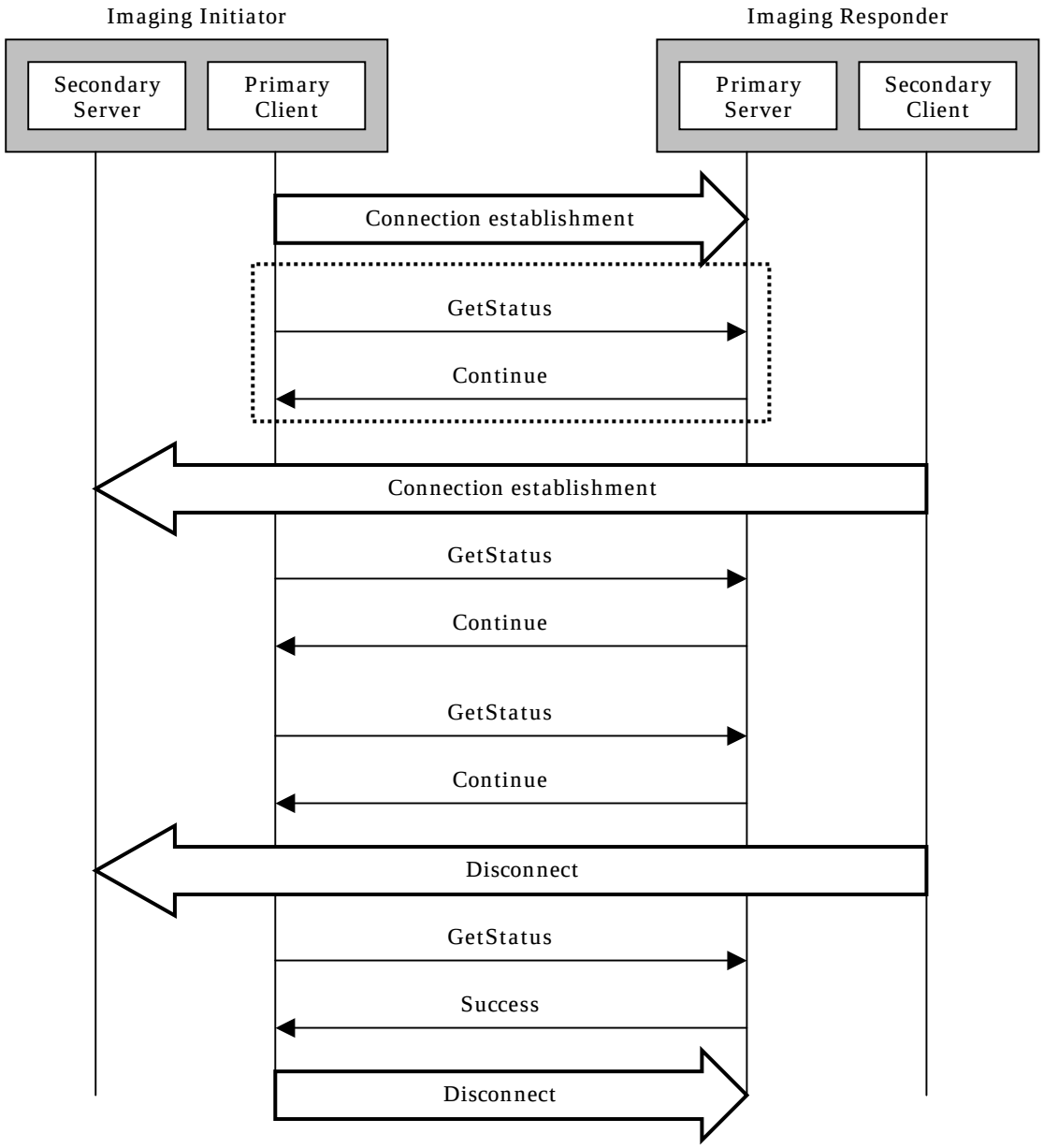


図４　Advanced Image Print **における推奨実装を行った場合の通信シーケンス例**

以下、解説を行う。

図４の点線で囲まれた部分に関して Responder **は、**Initiator **の** Primary Client **が** GetStatus request **を送信する場合と送信しない場合があることを想定しなければならない。（**GetStatus request **の送信条件について、**BIP[1]**では** StartArchive **に関しては明記されているが、**StartPrint **に関しては記述されていない。）例えば、**Responder **において「**secondary connection **が確立していないときにレスポンスコードとしてエラーまたは“** OK, Success **”を返す」という実装をすると、**Responder **が正常動作している場合には** secondary connection **の確立処理の直前あるいはその最中に** Initiator **が** primary connection **を切断してしまうことが懸念される。（**BIP[1]**では、“** OK, Success **”を受信すると** Initiator **は** primary

connection を切断するものと明記されている。）このときの通信シーケンス例を図５に示す。

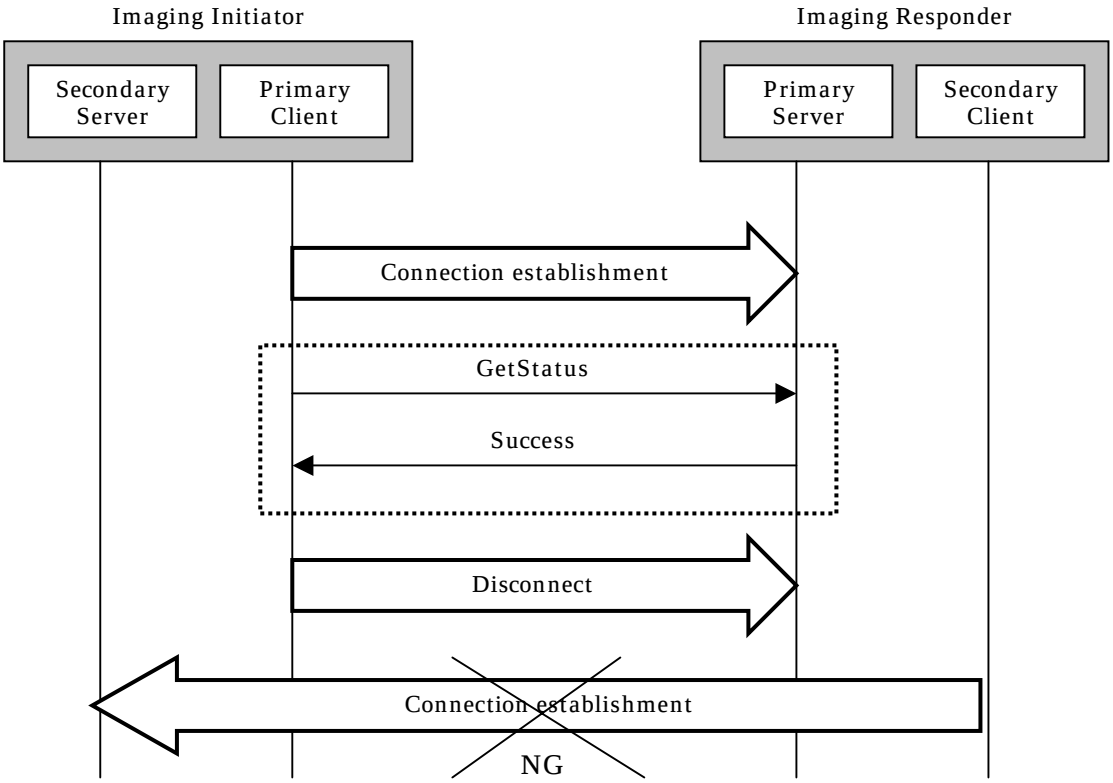


図５　推奨実装を行わなかった場合に懸念される問題

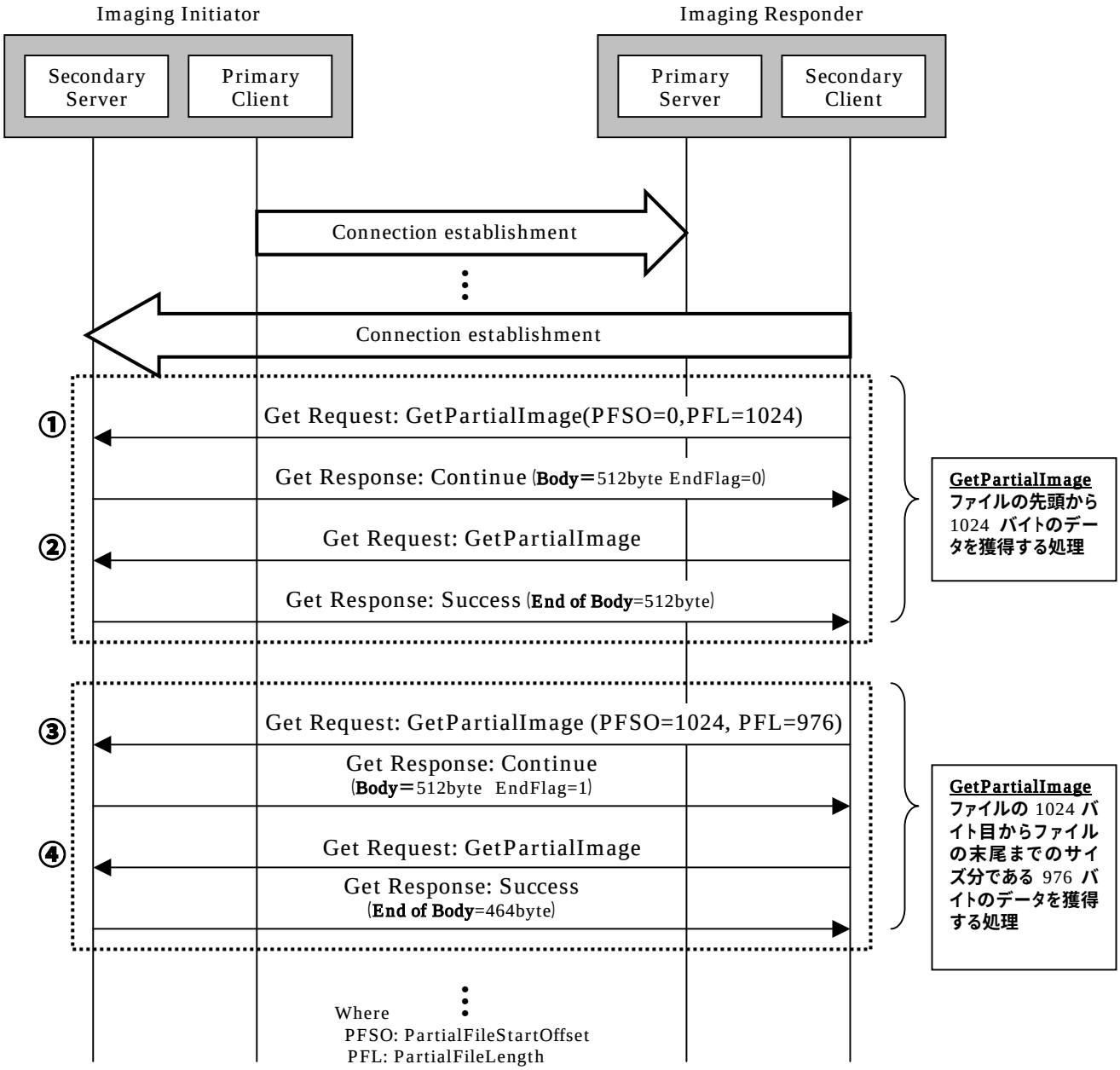
よって、図５のシーケンスのような問題を回避するために、冒頭にあげた追加実装を推奨する。

Q7. GetPartialImage の Body ヘッダと End of Body ヘッダの利用方法について

A.

Initiator は、GetPartialImage で要求された partial image ファイルを送信する際に、partial image ファイルの 最後の部分である場合には End of Body ヘッダを利用し、それ以外の場合は、Body ヘッダを利用することを推奨する。また、レスポンスが image ファイルの最後の部分である場合には EndFlag=1 を設定する。

以下に、ファイルサイズが 2000 バイトの JPEG 画像(img5.jpg)を GetPartialImage で獲得する場合の Body ヘッダと End of Body ヘッダの利用について例を示す。



Secondary Client Request:	**Bytes**	**Meaning**
①Opcode	0x83	**GET**, Final bit set
	0xnnnn	Length of packet
	0xCB	HI for **Connection Id** header
	0xnnnnnnnn	Connld = nnnnnnnn
	0x42	HI for **Type** header
	0x0014	Length of **Type** header
	x-bt/img-partial	MIME Media-Type of object, null terminated
	0x01	HI for **Name** header
	0x0015	Length of **Name** header
	0069 006D 0067	**Name** header content “img5.jpg”
	0035 002E 006A	(UTF-16 encoded), null terminated
	0070 0067 0000	
	0x4C	HI for **Application Parameter** header
	0x000F	Length of **Application Parameter** header
	04 04 00000400	PartialFileLength (= 1024 Bytes)
	05 04 00000000	PartialFileStartOffset (= 0)
Secondary Server response:		
①Response code	0x90	**Continue,** Final bit set
	0xnnnn	Length of response packet
	0xC3	HI for **Length** header
	0x00000400	Length of partial file
	0x4C	HI for **Application Parameter** header
	0x000C	Length of **Application Parameter** header
	06 04 000007D0	TotalFileSize (= 2000 Bytes)
	07 01 00	EndFlag (= FALSE)
	0x48	HI for **Body** header
	0x0200	Length of **Body** header
	0x……	
Secondary Client Request:		
②Opcode	0x83	**GET**, Final bit set
	0xnnnn	Length of packet
	0xCB	HI for **Connection Id** header
	0xnnnnnnnn	Connld = nnnnnnnn
Secondary		

Server response:		
②Response code	0xA0	**Success,** Final bit set
	0xnnnn	Length of response packet
	0x49	HI for **End of Body** header
	0x0200	Length of **End of Body** header
	0x……	

Secondary Client Request:	**Bytes**	**Meaning**
③Opcode	0x83	**GET**, Final bit set
	0xnnnn	Length of packet
	0xCB	HI for **Connection Id** header
	0xnnnnnnnn	Connld = nnnnnnnn
	0x42	HI for **Type** header
	0x0014	Length of **Type** header
	x-bt/img-partial	MIME Media-Type of object, null terminated
	0x01	HI for **Name** header
	0x0015	Length of **Name** header
	0069 006D 0067	**Name** header content “img5.jpg”
	0035 002E 006A	(UTF-16 encoded), null terminated
	0070 0067 0000	
	0x4C	HI for **Application Parameter** header
	0x000F	Length of **Application Parameter** header
	04 04 000003D0	PartialFileLength (= 976 Bytes)
	05 04 00000400	PartialFileStartOffset (= 1024)
Secondary Server response:		
③Response code	0x90	**Continue,** Final bit set
	0xnnnn	Length of response packet
	0xC3	HI for **Length** header
	0x000003D0	Length of partial file
	0x4C	HI for **Application Parameter** header
	0x000C	Length of **Application Parameter** header
	06 04 000007D0	TotalFileSize (= 2000 Bytes)
	07 01 01	EndFlag (= TRUE)
	0x48	HI for **Body** header
	0x0200	Length of **Body** header
	0x……	

Secondary Client Request:		
④Opcode	0x83 0xnnnn 0xCB 0xnnnnnnnn	**GET**, Final bit set Length of packet HI for **Connection Id** header Connld = nnnnnnnn
Secondary Server response:		
④Response code	0xA0 0xnnnn 0x49 0x01D0 0x……	**Success,** Final bit set Length of response packet HI for **End of Body** header Length of **End of Body** header

図6 Advanced Image Print **における** Body **ヘッダと** End of Body **ヘッダの利用例**

Q8. デジタルスチルカメラ等に送信する画像フォーマットについて

A.

JPEG 画像フォーマットといっても世の中にはいろいろなタイプの JPEG 画像フォーマットがあるため、すべての画像フォーマットに対応するのは難しく、JPEG 画像の送信に成功しても Responder が表示できるとは限らない。ここでは、デジタルスチルカメラ等に対して JPEG 画像を送信する場合の注意事項について以下に示す。

デジタルスチルカメラ等では Exif[3]と DCF[4]に準拠した画像フォーマットが一般的に使用されている。これ以外の JPEG 画像をデジタルスチルカメラ等に送信した場合、主画像やサムネイル画像が表示されない場合があるため、以下のように実装をすることを推奨する。

１)デジタルスチルカメラに JPEG 画像を送信する場合、Initiator は Exif[3]と DCF[4]に準拠可能ならば、これに準拠した画像フォーマット(サムネイル付)の JPEG 画像を送信することを推奨する。 (デジタルスチルカメラかどうかは CoD で判断することができる。)

２）デジタルスチルカメラに JPEG 画像を送信する場合、Initiator は Exif[3]と DCF[4]に準拠不可能であれば、以下の条件の JPEG 画像を送ることを推奨する。

- 標準ハフマンテーブル[5]を使用のこと
- ベースライン DCT 方式[5]であること

なお、デジタルスチルカメラは Responder として動作する場合、Exif[3]と DCF[4]に準拠しない画像フォーマットが送信される可能性があることを考慮しておくべきである。

Q9. ファイル名が機器のファイルシステムで処理不可能な場合について

A.

Nameヘッダに記述されている文字がResponderのファイルシステムで利用不可能な場合がある。例えば、Responderが日本語を取り扱えない時に、InitiatorがPutImageのNameヘッダに日本語ファイル名を利用して画像を送信する場合がこれに該当する。

このようなケースは、BIP[1]のホワイトペーパー[2]のQ14においても同様の項目が示されており、実装依存の処理であるが相互接続を高めるためにも以下の処理を推奨する。

- 送信されたファイル名が Responder のファイルシステムで利用できない場合でも、送信されたファイル名を Responder 内で取り扱えるファイル名に変換して処理を続行する。

Q10. Maximum OBEX Packet Size について

A.

OBEX[6]では接続処理時にお互いの Maximum OBEX Packet Size を比較し小さい方にあわせて、以後の通信パケットサイズを決める。（OBEX[6]の仕様上の可能な Packet Size サイズは 255bytes～64Kbytes-1）

このサイズが小さいと送信レートが極端に遅くなる。例えば PUT オペレーションの場合、OBEX Client がデータを送ると OBEX Server がレスポンスを返す。OBEX Client は送信したデータのレスポンスを受け取るまで次のデータを送信しない。このレスポンス待ちが大きなオーバーヘッドになっている。Packet Size が大きいと一度に送るデータ量が増えるため、レスポンス待ちの回数が減りスループットが向上する。

BIP[1]では、画像を扱うためデータサイズが大きい。スループット向上はエンドユーザーにとって重要であるため、Packet Size は出来るだけ大きくすることを推奨する。Packet Size として、サムネイル画像を一度に送信できる程度のサイズ（例えば、5Kbytes）を推奨する。

Q11. ImagePush サポート時のサービスレコードの Supported features 属性について

A.

BIP[1]の 6.1.1 を参照すること。

Q12. サービスレコードの Total image data capacity の取扱いについて

A.

Total image data capacity には Responder で画像を保存可能なメモリサイズの参考値を記述する。例えば、画像の受信に応じて保存可能なメモリサイズは減少するが、Total image data capacity 値を動的に変更する必要はない。

Initiator は Total image data capacity 値を参照し、送信画像のサイズと比較することで、明らかに送信画像のサイズが大きい場合は、画像送信を行う以前に速やかに処理を中止することができる。

Total image data capacity 値はあくまで参考値なので、Total image data capacity 値が送信画像のサイズより大きいからといって、画像を送信した際に必ずしもメモリフルが起きないことを保証するものではない。

Q13. Class of Device/Service (CoD)の設定について

A.

CoD は３octet の値であり、

b_{23} b_{22} b_{21} b_{20} b_{19} b_{18} b_{17} b_{16} b_{15} b_{14} b_{13} b_{12} b_{11} b_{10} b_9 b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0

と表記することができる。ここで、“ b_{23} b_{22} b_{21} b_{20} b_{19} b_{18} b_{17} b_{16} b_{15} b_{14} b_{13} ”は Major Service Classes、“ b_{12} b_{11} b_{10} b_9 b_8 ”は Major Device Classes、“ b_7 b_6 b_5 b_4 b_3 b_2 ”は Minor Device Classes である。また、“ b_1 b_0 ”は Format Type field であり、値は“ 0 0 ”である。それぞれのビットの意味については、Assigned Numbers Specification [7]に記載されているとおりとする。

実装する機能（サービス）によって Major Service Classes の値は変わるものとするが、BIP[1]対応機器は Object Transfer 属性のビット(b_{20})を１にすることを推奨する。CoD の例を以下に示す。ここで記述表現は次のとおりとする。

機器の種類

Major Service Class: b_{23} b_{22} b_{21} b_{20} b_{19} b_{18} b_{17} b_{16} b_{15} b_{14} b_{13}

Major Device Class: b_{12} b_{11} b_{10} b_9 b_8

Minor Device Class: b_7 b_6 b_5 b_4 b_3 b_2

Digital Still Camera

Major Service Class: 0 0 0 **1** 1 0 0 0 0 0 0

Major Device Class: 0 0 1 1 0

Minor Device Class: 0 0 1 0 0 0

Printer

Major Service Class: 0 0 0 **1** 0 1 0 0 0 0 0

Major Device Class: 0 0 1 1 0

Minor Device Class: 1 0 0 0 0 0

Cellular Phone

Major Service Class: 0 1 0 **1** 0 0 0 0 0 0 0

Major Device Class: 0 0 0 1 0

Minor Device Class: 0 0 0 0 0 1

Laptop PC

Major Service Class: 0 0 0 **1** 0 0 0 0 0 0 0

Major Device Class: 0 0 0 0 1

Minor Device Class: 0 0 0 0 1 1

Handheld PC/PDA

Major Service Class: 0 0 0 **1** 0 0 0 0 0 0 0

Major Device Class: 0 0 0 0 1

Minor Device Class: 0 0 0 1 0 0

Palm sized PC

Major Service Class: 0 0 0 **1** 0 0 0 0 0 0 0

Major Device Class: 0 0 0 0 1

Minor Device Class: 0 0 0 1 0 1

参画メンバー

清水一夫		オリンパス光学株式会社/ITX株式会社
玉津保治		オープンインタフェース株式会社
田中英明		オープンインタフェース株式会社
高橋康将		オープンインタフェース株式会社
斎藤創一		株式会社デンソー
稲葉清高		株式会社リコー
内山裕章		株式会社リコー
川村卓也	リーダー	株式会社東芝
青木孝泰		株式会社東芝　デジタルメディアネットワーク社
藤井賢一	サブリーダー	キヤノン株式会社
倉田賢一		セイコーエプソン株式会社
小川智広		ソニー株式会社
安田光義		ソニー・エリクソン・モバイルコミュニケーションズ株式会社
加藤博史		太陽誘電株式会社
村上克己		富士通株式会社/富士通デバイス株式会社
渡辺賀央		富士通株式会社/富士通デバイス株式会社
稲垣達也		松下電器産業株式会社

（敬称略・会社名50音順）

MCPC TR-001 Ver.1.0
Basic Imaging Profile
Technical Refarence Ver.1.0

平成15年9月26日

発行元：モバイルコンピューティング推進コンソーシアム（ＭＣＰＣ）
〒105-0011 東京都港区芝公園３－５－１２ 芝公園真田ビル