Picture Translation – A Challenge

DIaLOGIKa/makz/divo/wk 23 June 2008

Contents

Introduction

This document explains how the binary data structures of inline and floating pictures are to be interpreted to be able to convert them to OpenXML.

The following descriptions and algorithms are based on *Microsoft Word 2007 Binary File Format Documentation*, *Microsoft Office 97-2007 Office Drawing Binary File Format* and our own findings.

Unfortunately, the above-mentioned specifications exhibit a number of shortcomings as far as inline and floating pictures are concerned.

How Inline Pictures are Stored in a .doc File

A picture is represented in the document text stream as a special character, an ASCII 1 whose CHP has the fSpec bit set to 1. The file location of the picture is stored in chp.fcPic which is a byte offset into the data stream. Beginning at this position, a header data structure, the PIC, is stored.

This header data structure is described as follows:

Picture Descriptor (on File) (PICF)

b_{10}	b_{16}	Field	Type	Size	Bitfield	Comments
0	0	lcb	long			Number of bytes in the PIC structure plus size of following picture data which may be a Window's metafile, a bitmap, or the filename of a TIFF file. In the case of a Macintosh PICT picture, this includes the size of the PIC, the standard "x" metafile, and the Macintosh PICT data. See Appendix B for more information.
4	4	cbHeader	unsigned			Number of bytes in the PIC (to allow for future expansion).
6	6	mfp.mm	short			
…	…	…	…	…	…	…
46	2E	brcTop	BRC			Specification for border above picture
54	36	brcLeft	BRC			Specification for border to the left of picture
62	3E	brcBottom	BRC			Specification for border below picture
70	46	brcRight	BRC			Specification for border to the right of picture
78	4E	dxaOrigin	short			Horizontal offset of hand annotation origin
80	50	dyaOrigin	short			Vertical offset of hand annotation origin
82	52	cProps	short			Unused
84	54	rgb				Variable array of bytes containing Window's metafile, bitmap or TIFF file filename

This does not match to the PICF that we find in our test documents (the sample below describes a JPEG image inserted in a binary Word document):

■ = Fix part of the PIC structure

■ = variable part of the PIC structure, the rgb array …

■ = JPEG Image

```
(0000) E3 8B 00 00 44 00 64 00 00 00 00 00 00 00 08 00
(0010) 00 00 00 00 00 00 00 00 00 00 00 00 70 17 70 17
(0020) 97 02 8A 02 00 00 00 00 00 00 00 00 00 00 00 00
(0030) 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
(0040) 00 00 00 00 0F 00 04 F0 A8 00 00 00 B2 04 0A F0
(0050) 08 00 00 00 01 04 00 00 00 0A 00 00 73 00 0B F0
(0060) 76 00 00 00 00 01 1A 40 00 00 02 01 79 3C 00 00
(0070) 04 41 01 00 00 00 05 C1 0E 00 00 00 06 01 02 00
(0080) 00 00 FF 01 00 00 08 00 81 C3 3E 00 00 00 30 00
(0090) 34 00 30 00 38 00 32 00 38 00 00 00 44 00 61 00
(00A0) 73 00 20 00 69 00 73 00 74 00 20 00 6D 00 65 00
(00B0) 69 00 6E 00 20 00 61 00 6C 00 74 00 65 00 72 00
(00C0) 6E 00 61 00 74 00 69 00 76 00 65 00 72 00 20 00
(00D0) 54 00 65 00 78 00 74 00 00 00 13 00 22 F1 06 00
(00E0) 00 00 BF 01 00 00 60 00 00 00 10 F0 04 00 00 00
(00F0) 00 00 00 80 52 00 07 F0 E7 8A 00 00 05 05 74 E0
(0100) 55 D1 2E 89 8A 02 D6 43 91 69 25 FB 94 CC FF 00
(0110) C3 8A 00 00 01 00 00 00 44 00 00 00 00 00 7A 04
(0120) A0 46 1D F0 BB 8A 00 00 74 E0 55 D1 2E 89 8A 02
(0130) D6 43 91 69 25 FB 94 CC FF FF D8 FF E0 00 10 4A
(0140) 46 49 46 00 01 02 00 00 64 00 64 00 00 FF EC 00
(0150) 11 44 75 63 6B 79 00 01 00 04 00 00 00 26 00 00
```

Inconsistencies in the PICF Structure

The total length in PICF.lcb is correct as well as the most of the other fields in the struct. However, the following points are not clearly described:

- The value in PICF.cbHeader does not seem to make sense. We always find a value of 0x44 bytes for the length of the PICF header, but the size of the PICF struct is already at least 0x54 bytes according to the spec.
- The format of the PICF.rgb field is not specified. The filename of the picture seems only one part of the information stored here. How should this field be interpreted?
- How to find the beginning of the picture data, which starts at a variable offset?

Corrected PICF Data Structure

After some guessing we found out that the BRC values for the borderaround the picture are not 8 bytes each as stated in the specification but only 4 bytes. Consequently, the total length of the header strucure is 44 bytes (as indicated in PICF.cbHeader) and the subsequent data structure starts at by 0x0044 (see below).

The corrected PICF data structure looks as follows:

…	…	…	…	…	…	…
46	2E	brcTop	BRC			Specification for border above picture

50	**32**	brcLeft	BRC	Specification for border to the left of picture
54	**36**	brcBottom	BRC	Specification for border below picture
58	**3A**	brcRight	BRC	Specification for border to the right of picture
62	**3E**	dxaOrigin	short	Horizontal offset of hand annotation origin
64	**40**	dyaOrigin	short	Vertical offset of hand annotation origin
66	**42**	cProps	short	Unused
68	**44**	`rgb`		Variable array of bytes containing Window's metafile, bitmap or TIFF file filename

Uncover PICF.rgb

We assume that the PICF.rgb structure consists of a number of drawing group objects. Coming back to our example:

```
(0040) 00 00 00 00 0F 00 04 F0 A8 00 00 00 B2 04 0A F0
(0050) 08 00 00 00 01 04 00 00 00 0A 00 00 73 00 0B F0
(0060) 76 00 00 00 00 01 1A 40 00 00 02 01 79 3C 00 00
(0070) 04 41 01 00 00 00 05 C1 0E 00 00 00 06 01 02 00
(0080) 00 00 FF 01 00 00 08 00 81 C3 3E 00 00 00 30 00
(0090) 34 00 30 00 38 00 32 00 38 00 00 00 44 00 61 00
(00A0) 73 00 20 00 69 00 73 00 74 00 20 00 6D 00 65 00
(00B0) 69 00 6E 00 20 00 61 00 6C 00 74 00 65 00 72 00
(00C0) 6E 00 61 00 74 00 69 00 76 00 65 00 72 00 20 00
(00D0) 54 00 65 00 78 00 74 00 00 00 13 00 22 F1 06 00
(00E0) 00 00 BF 01 00 00 60 00 00 00 10 F0 04 00 00 00
(00F0) 00 00 00 80 52 00 07 F0 E7 8A 00 00 05 05 74 E0
(0100) 55 D1 2E 89 8A 02 D6 43 91 69 25 FB 94 CC FF 00
(0110) C3 8A 00 00 01 00 00 00 44 00 00 00 00 00 7A 04
(0120) A0 46 1D F0 BB 8A 00 00 74 E0 55 D1 2E 89 8A 02
(0130) D6 43 91 69 25 FB 94 CC FF FF D8 FF E0 00 10 4A
```

The first drawing object has an fbt value of 0xF004, consequently it is an msofbtSpContainer. Its length is 0xA0. The second object is an msofbtBSE (0xF007, a BLIP store entry record). Its length includes the image. This object is immediately follows by an msofbtBlip object (0xF01D, BLIP object). The difference between this ftb value and the starting ftb value of all BLIP objects (0xF018) defines the type of image: 0xF01A-0xF018 = 5, i.e. a JPEG image. The length of this object also includes the image data. The length of the header structure of the JPEG BLIP object is 17 bytes, i.e. the actual image data starts with 0xFF 0xD8 0xFF …

Translating Floating Pictures

The data structures used to store floating pictures in a binary Word document are even more complex than those for inline pictures.

Floating pictures are stored as a drawing record or shape. All drawings stored in a document are referenced via the dggInfo pointer found in the FIB at position 0x22A. The pointer points to a drawing group container in the table stream and is followed – as usual – by a long value specifying the length of the data structure in the table stream.

Example:

```
(0210)  00 00 79 13 00 00 00 00 00 00 A5 13 00 00 00 00
(0220)  00 00 00 00 00 00 00 00 00 00 A5 13 00 00 D9 01
(0230)  00 00 A7 16 00 00 16 00 00 00 B0 15 00 00 00 00
```

Swinging between the Streams

The dggInfo structure is shown below. It starts with an msofbtDggContainer record (0xF000) which has a length of 0x6C bytes, followed by an msofbtDgg record (0xF002). One of the records inside the msofbtDggContainer is the msofbtBSE (0xF007) which points back into the WordDocument stream (0x0E34) where the actual picture is stored (FBSE.foDelay).

```
(1390)  00 07 08 00 00 05 00 00 00 00 08 00 00 07 08 00
(13A0)  00 06 00 00 00 0F 00 00 F0 6C 00 00 00 00 00 06
(13B0)  F0 18 00 00 00 02 08 00 00 02 00 00 00 02 00 00
(13C0)  00 01 00 00 00 01 00 00 00 03 00 00 00 1F 00 01
(13D0)  F0 2C 00 00 00 52 00 07 F0 24 00 00 00 05 05 46
(13E0)  0C AD 97 2C 04 3F 4A 11 DE E3 FB FC B3 27 66 FF
(13F0)  00 66 8B 00 00 01 00 00 00 34 0E 00 00 00 00 00
(1400)  00 40 00 1E F1 10 00 00 00 FF FF 00 00 00 00 FF
(1410)  00 80 80 80 00 F7 00 00 10 00 0F 00 02 F0 5C 01
(1420)  00 00 10 00 08 F0 08 00 00 00 02 00 00 00 02 04
(1430)  00 00 0F 00 03 F0 FA 00 00 00 0F 00 04 F0 28 00
(1440)  00 00 01 00 09 F0 10 00 00 00 00 00 00 00 00 00
(1450)  00 00 00 00 00 00 00 00 00 00 02 00 0A F0 08 00
(1460)  00 00 00 04 00 00 05 00 00 00 0F 00 04 F0 C2 00
(1470)  00 00 B2 04 0A F0 08 00 00 00 02 04 00 00 00 0A
(1480)  00 00 93 00 0B F0 7E 00 00 00 BF 00 04 00 04 00
(1490)  04 41 01 00 00 00 05 C1 16 00 00 00 3F 01 00 00
(14A0)  06 00 BF 01 00 00 10 00 FF 01 00 00 08 00 80 C3
(14B0)  14 00 00 00 81 C3 1E 00 00 00 BF 03 00 00 22 00
(14C0)  49 00 4D 00 47 00 5F 00 31 00 34 00 33 00 36 00
(14D0)  5F 00 63 00 00 00 50 00 69 00 63 00 74 00 75 00
(14E0)  72 00 65 00 20 00 30 00 00 00 49 00 4D 00 47 00
(14F0)  5F 00 31 00 34 00 33 00 36 00 5F 00 63 00 2E 00
(1500)  4A 00 50 00 47 00 00 00 23 00 22 F1 0C 00 00 00
(1510)  BF 01 00 00 60 00 3F 05 00 00 01 00 00 00 10 F0
(1520)  04 00 00 00 00 00 00 00 00 00 11 F0 04 00 00 00
(1530)  01 00 00 00 0F 00 04 F0 42 00 00 00 12 00 0A F0
(1540)  08 00 00 00 01 04 00 00 00 0E 00 00 53 00 0B F0
(1550)  1E 00 00 00 BF 01 00 00 10 00 CB 01 00 00 00 00
(1560)  FF 01 00 00 08 00 04 03 09 00 00 00 3F 03 01 00
```

```
(1570)  01 00 00 00 11 F0 04 00 00 00 01 00 00 00 00 00
(1580)  00 00 07 00 00 00 02 04 00 00 00 00 00 00 00 00
```

The FBSE.foDeleay pointer refers to the following data structure in the WordDocument stream. As you can see, this data structure is similar to the above described data structure for inline pictures (actually, the picture size is slightly different due to different images used in the example files).

```
(0E20)  24 90 A0 05 25 B0 00 00 17 B0 C4 02 18 B0 C4 02
(0E30)  0C 90 C4 02 A0 46 1D F0 5E 8B 00 00 46 0C AD 97
(0E40)  2C 04 3F 4A 11 DE E3 FB FC B3 27 66 FF FF D8 FF
(0E50)  E1 0F 28 45 78 69 66 00 00 49 49 2A 00 08 00 00
(0E60)  00 0A 00 0F 01 02 00 06 00 00 00 86 00 00 00 10
(0E70)  01 02 00 17 00 00 00 8C 00 00 00 12 01 03 00 01
```

Undocumented Padding Byte

After the msofbtDggContainer object we encounter an msofbtDgContainer (0xF002) with a number of properties stored for the drawing objects.

However, between the msofbtDggContainer and the msofbtDgContainer record we realized that Word always (?) stores an undocumented padding zero byte, cf. position 0x1419.