The Metadata in JPEG files

This article shares Tuan's (GSoC13 Student) investigations about the metadata structure in JPEG files. It also introduces briefly the algorithm used in exiv2 to read and write Exif, IPTC, XMP data and image comments on the JPEG files.

1. Background

JPEG ISO standard is a commonly used method of lossy compression for digital photography. The name "JPEG" stands for Joint Photographic Experts Group, the name of the committee.

JPEG refers only to a class of compression algorithms, not to a specific file format. In order to produce files with embedded JPEG streams, a number of file format standards have been adapted or devised. Some of them are JPEG /JFIF, JPEG /SPIFF (Still Picture Interchange File Format), JPEG /CIFF, JPEG/Exif (Exchangeable image file format).

Among them, the most common types are JPEG/Exif and JPEG/JFIF.

- JPEG/Exif is the most common image format used by digital cameras and other photographic image capture devices.
- JPEG/JFIF is the most common format for storing and transmitting photographic images on the World Wide Web.

2. The metadata structure in JPEG

A JPEG file contains several segments; each segment contains different kinds of data, delimited by two-byte codes called markers. The markers are hexadecimal; they begin with 0xFF and end with a code (1 byte) indicating the kind of marker.

Some markers consist of just those two bytes; others are followed by two bytes indicating the length of marker-specific payload data that follows. The length includes the two bytes for the length, but not the two bytes for the marker.

Short name	Bytes	Payload	Name and Comments
SOI	0xFF, 0xD8	None	Start Of Image
SOF0	0xFF, 0xC0	Variable size	Start Of Frame (Baseline DCT) Indicates that this is a baseline DCT-based JPEG, and specifies the width, height, number of components, and component subsampling
SOF2	0xFF, 0xC2	Variable size	Start Of Frame (Progressive DCT) Indicates that this is a progressive DCT-based JPEG, and specifies the width, height, number of components, and component subsampling
DHT	0xFF, 0xC4	Variable size	Define Huffman Table(s)
DQT	0xFF, 0xDB	Variable size	Define Quantization Table(s)
DRI	0xFF, 0xDD	2 bytes	Define Restart Interval Specifies the interval between RSTn markers, in macroblocks. This marker is followed by two bytes indicating the fixed size so it can be treated like any other variable size segment.
SOS	0xFF, 0xDA	Variable size	Start Of Scan Begins a top-to-bottom scan of the image. In baseline DCT JPEG images, there is generally a single scan. Progressive DCT JPEG images usually contain multiple scans. This marker specifies which slice of data it will contain, and is immediately followed by entropy-coded data.
RSTn	0xFF, 0xDn n(n=0..7)	None	Restart Inserted every r macroblocks, where r is the restart interval set by a DRI marker. Not used if there was no DRI marker. The low 3 bits of the marker code cycle in value from 0 to 7.
APPn	0xFF, 0xEn	Variable size	Application-specific For example, an Exif JPEG file uses an APP1 marker to store metadata, laid out in a structure based closely on TIFF.
COM	0xFF, 0xFE	Variable size	Comment
EOI	0xFF, 0xD9	None	End Of Image

Fig.1. The common JPEG markers. From Wikipedia, https://en.wikipedia.org/wiki/JPEG

The metadata in JPEG file is stored in APPn (0xFF, 0xEn) segment and the comment is stored in COM segment (0xFF, 0xFE). Several vendors might use the same APPn marker type to include their information, so these markers often begin with a vendor name (e.g., "Exif" or "Adobe") or some other identifying string.

Exiv2 provides fast and easy read write access to the Exif, IPTC and XMP. Hence, this article only focuses on the position of Exif, IPTC and XMP data in JPEG files.

2.1 Exif

Exif JPEG file uses an APP1 segment to store the information (and multiples APP2 segments for flashPix data). Exif APP1 segment stores a great amount of information on photographic parameters for digital cameras and it is the preferred way to store thumbnail images nowadays. It can also host an additional section with GPS data. All details about Exif are available at [[http://www.exif.org/Exif2-2.PDF]]

In theory, Exif APP1 is recorded immediately after the SOI marker (the marker indicating the beginning of the file). However, this leads to the incompatibility between the Exif and JFIF standards because both of them specify that their particular application segment (APP0 for JFIF, APP1 for Exif) must be the first in the image file. In practice, most JPEG files contain a JFIF marker segment (APP0) that precedes the Exif APP1. This allows older readers to correctly handle the format JFIF segment, while newer readers also decode the following Exif segment, being less strict about requiring it to appear first. This way will not affect the image decoding for most decoders, but poorly designed JFIF or Exif parsers may not recognize the file properly.

Exif APP1 segment consists of the APP1 marker (0xFFE1), Exif identifier string (“Exif\0\0”), and the attribute information itself. The identifier string "Exif\0\0” is used to avoid a conflict with other applications using APP1 (e.g XMP).

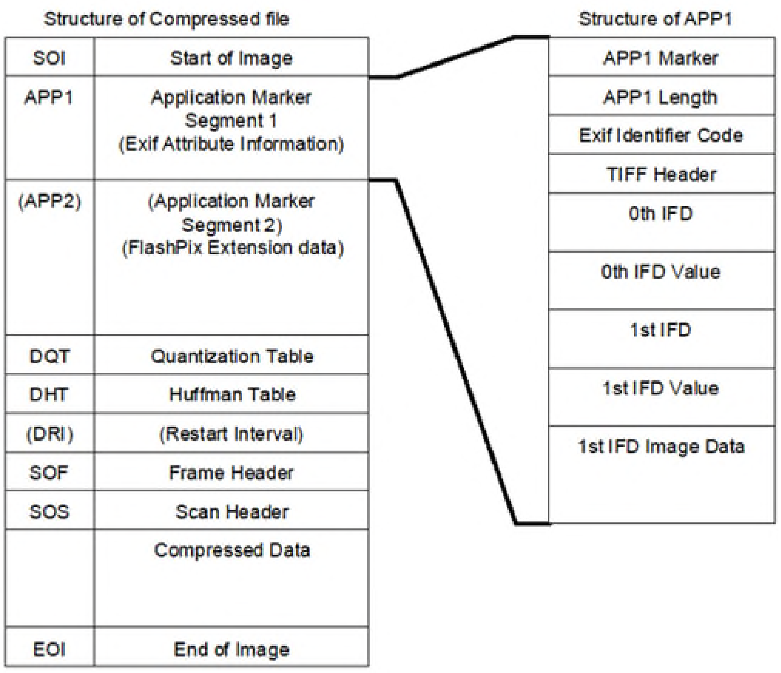

Fig.2. Basic Structure of JPEG Files. From Exif.org, *http://www.exif.org/Exif2-2.PDF*

Exif does not use APPn segments other than APP1, APP2 and COM segments. However, some unknown APPn may still exist on the file structure and Exif readers should be designed to skip over them.

2.2 XMP

In a typical edited JPEG file, XMP (eXtensible Metadata Platform) information is typically included alongside Exif and IPTC (Information Interchange Model data). XMP uses an APP1 segment in order to store metadata information; the storage format is RDF (Resource Description Framework) implemented as an application of XML.

XMP APP1 segment consists of the APP1 marker (0xFFE1), XMP identifier string (“ http://ns.adobe.com/xap/1.0/\x00”), and Unicode XMP packet (the encoding is usually UTF-8, but it can also be UTF-16 or UTF-32). The packet cannot be split in multiple segments, so there is a maximum size of approximately 64KB (2^16-1 bytes).

The structure of the packet content can be found at http://www.w3.org/TR/REC-rdf-syntax/.
The reference document for XMP 3.2 can be downloaded from Adobe Systems Incorporated http://xml.coverpages.org/xmp.html

When Adobe first introduced XMP, it was intended to totally contain the XMP block in a single segment of a JPEG. So, the XMP/xml could not be longer that 65k bytes. In 2010, the standard was upgraded to contain a mechanism by which multiple segments could be used.

The application exiv2(.exe) in version 0.25 supports an option -pX to extract the RAW XMP/xml packet from an image. The -pX option supports both the single segment (65k) and multi-segment standard. However libexiv2 generally does not support the multi-segment standard. A future version after v0.25 will fully support multi-segment XMP.

2.3 IPTC

Adobe Photoshop uses the APP13 segment for storing non-graphic information, such as layers, paths, IPTC data and more. The content of an APP13 segment is formed by APP1 marker (0xFFE1), an identifier string (usually "Photoshop 3.0\000", but also 'Adobe_Photoshop2.5:', used by earlier versions) followed by a sequence of resource data blocks. In general, a resource block contains only a few bytes, but there is the important IPTC block can be quite large. The IPTC block may not fit into one APP13 segment, so it can be split into multiple APP13 segments.

The reference document for the Photoshop file format is available at http://www.adobe.com/devnet-apps/photoshop/fileformatashtml/

2.4 ICC

The ICC profile is stored in one or more APP2 chunks. Although many ICC profiles are small, they can also be more that than a single chunk. Chunks are limited to 65535 bytes by the two-byte chunk length. To address this, the ICC APP2 data has a 16 byte header as follows:

bytes	data
0..11	ICC_PROFILE\0
12	icc chunk-count
13	icc total chunks

The maximum size of an ICC profile is limited to 256 * (63535 - 16) = 16,261,888 bytes. This is quite a large profile!

The first four bytes on an ICC Profile are the length of the file and bigEndian encoded.

3. Exiv2 JPEG read/write metadata algorithms

This section introduces briefly about the algorithm used in Exiv2. For more details, please download the source code of Exiv2 and read the code in src/jpgimage.cpp file.

3.1 Read Algorithm

From the above investigation, the read algorithm is quite simple. We just need to go through the markers, find and read the content of Exif APP1, XMP APP1, IPTC APP13 segments. Those segments all locate before the SOS segment (which is immediately followed by entropy-coded data) and often locate right after the SOI segment (not guaranteed). Hence, it’s not necessary to read the whole JPEG file to check whether the metadata exists or not; the process can be stopped whenever the marker SOS is found.

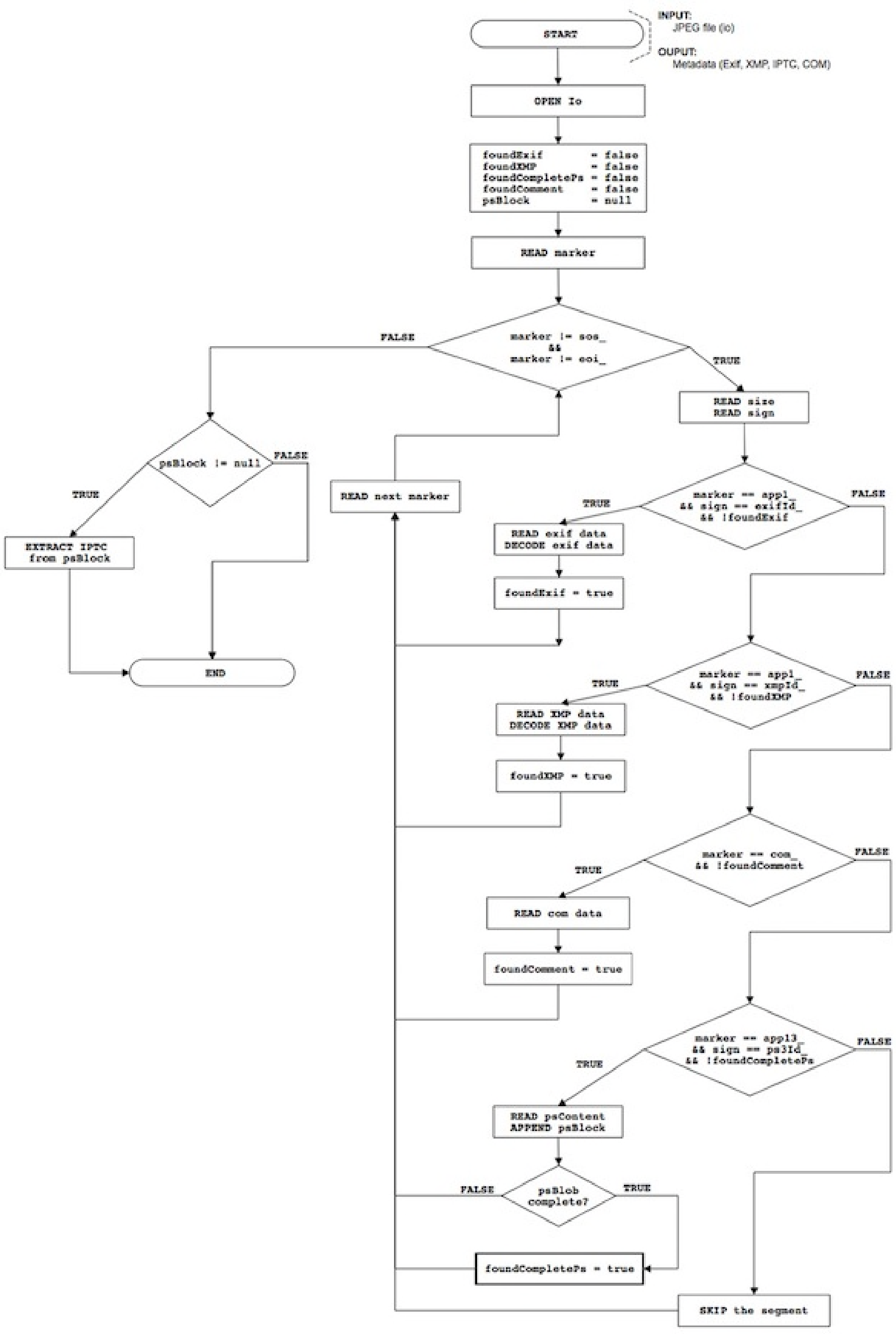

Fig.3. Flowchart of readMetadata

Notes:

- The standard JPEG files should have at most one Exif APP1 segment and one XMP APP1 segment. Hence, if there’s more than one Exif APP1 segment or one XMP APP1 segment, exiv2 only reads the first segment.
- JPEG files can have multiple comments, but for now exiv2 only reads the first one (most of the jpeg files only have one anyway).

3.2 Write Algorithm

The general idea for the write algorithm is to find Exif APP1, XMP APP1, IPTC APP13 segments, remove them from the JPEG file and replace them with the new metadata. To simplify this a bit, we agree the following rules (this is standard in most jpegs anyway).

- The order of the Exif, XMP, IPTC segments in the output file (regardless of their positions in the input file)
  SOI | (APP0) | (Exif App1) | (XMP App1) | (IPTC App13) | … | EOI
- The COM segment is located just before SOFn in the output file (regardless of its position in the input file).

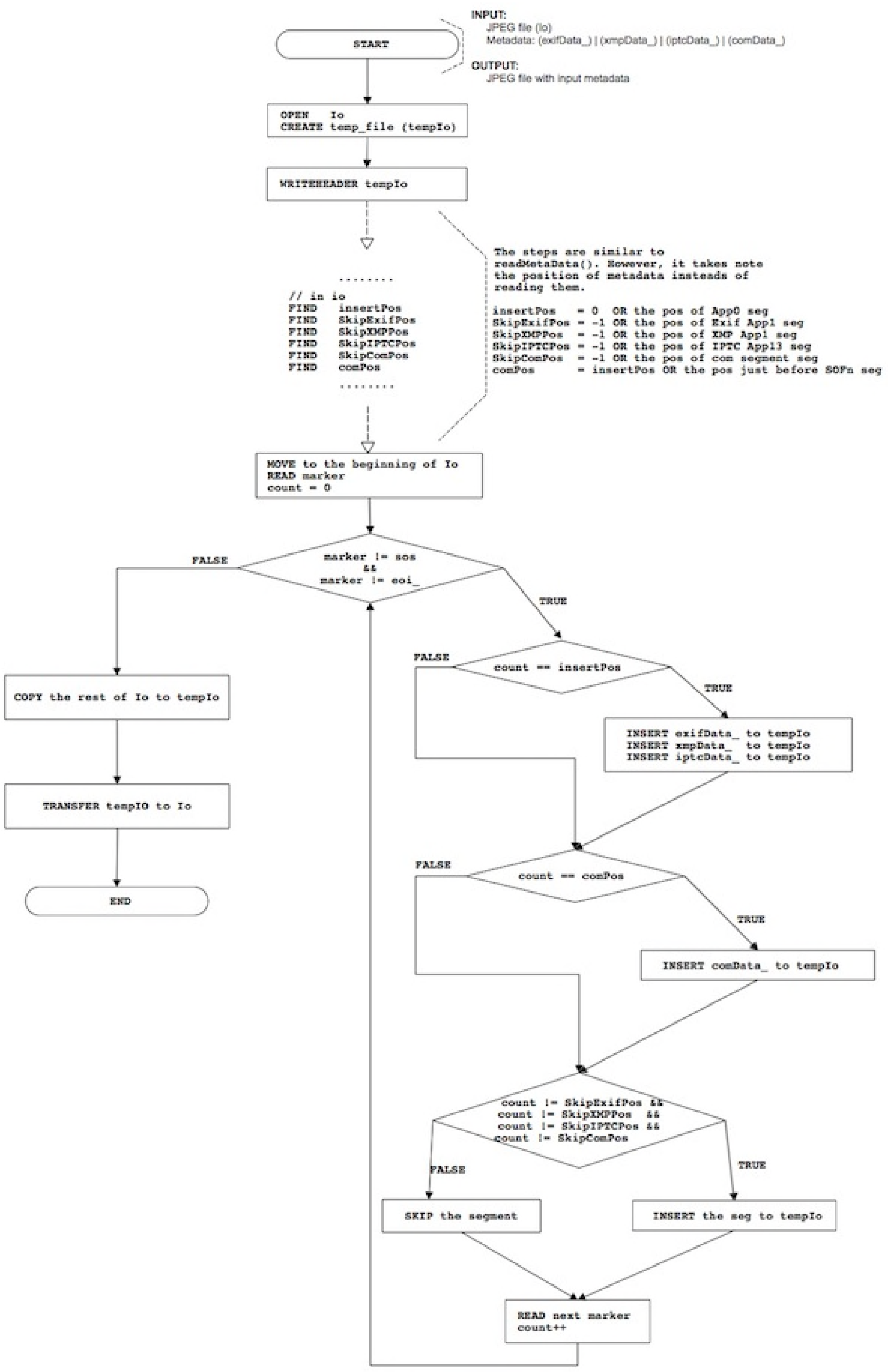
INPUT:
JPEG file (Io)
Metadata: (exifData_) | (xmpData_) | (iptcData_) | (comData_)
OUTPUT:
JPEG file with input metadata
START
OPEN Io
CREATE temp_file (tempIo)
WRITEHEADER tempIo
The steps are similar to readMetaData(). However, it takes note the position of metadata insteads of reading them.
insertPos = 0 OR the pos of App0 seg
SkipExifPos = -1 OR the pos of Exif App1 seg
SkipXMPPos = -1 OR the pos of XMP App1 seg
SkipIPTCPos = -1 OR the pos of IPTC App13 seg
SkipComPos = -1 OR the pos of com segment seg
comPos = insertPos OR the pos just before SOFn seg
....
// in io
FIND insertPos
FIND SkipExifPos
FIND SkipXMPPos
FIND SkipIPTCPos
FIND SkipComPos
FIND comPos
....
MOVE to the beginning of Io
READ marker
count = 0
marker != sos && marker != eoi_
FALSE
TRUE
COPY the rest of Io to tempIo
TRANSFER tempIO to Io
END
count == insertPos
FALSE
TRUE
INSERT exifData_ to tempIo
INSERT xmpData_ to tempIo
INSERT iptcData_ to tempIo
count == comPos
FALSE
TRUE
INSERT comData_ to tempIo
count != SkipExifPos && count != SkipXMPPos && count != SkipIPTCPos && count != SkipComPos
FALSE
TRUE
SKIP the segment
INSERT the seg to tempIo
READ next marker
count++

Fig.4. Flowchart of writeMetadata

Example 1:
The following is an example to show the changes in the structure of JPEG files after the writeMetadata. In this example, after adding the Exif.Photo.UserComment, not only the Exif APP1 segment is added, but the IPTC APP13 is also moved to right after Exif APP1.

```
$ ./exiv2 -pa test.jpg
Iptc.Application2.RecordVersion              Short       1  2
Iptc.Application2.Keywords                   String      4  2010
Iptc.Application2.Keywords                   String     10  California
Iptc.Application2.Keywords                   String      9  Christmas
Iptc.Application2.Keywords                   String      9  canderson
Iptc.Application2.Keywords                   String      6  nature
Iptc.Application2.Keywords                   String      6  winter
Iptc.Application2.Copyright                  String     12  (C)CANDERSON
$ ./exifprint test.jpg -struc
STRUTURE OF FILE:
marker | size | signature
0xd8
0xe0        16  JFIFHH
0xe2      3160  ICC_PROFILE
                           HLinomntrRGB
0xed     18462  Photoshop 3.08BIM
0xe1     34447  XMP://ns.adobe.com/xmp/extension/
0xdb        67
0xdb        67
0xc0        17
```

```
0xc4       31
0xc4       70
0xc4       29
0xc4       68
0xda       12
0xd9
-----------------
$ ./exiv2 -M"add Exif.Photo.UserComment Hello" test.jpg
$ ./exifprint test.jpg -struc
STRUTURE OF FILE:
marker | size | signature
0xd8
0xe0       16  JFIFHH
0xe1       66  ExifIIi
0xed    18462  Photoshop 3.08BIM
0xe2     3160  ICC_PROFILE
                          HLinomntrRGB
0xe1    34447  XMP://ns.adobe.com/xmp/extension/
0xdb       67
0xdb       67
0xc0       17
0xc4       31
0xc4       70
0xc4       29
0xc4       68
0xda       12
0xd9
-----------------
Exif.Image.ExifTag                           0x8769 Long        1  26
Exif.Photo.UserComment                       0x9286 Undefined  13  Hello
```

Example 2:
This example shows the output using exiv2(.exe) for v0.25 which has an option -pS to print the structure of the image file. You can clearly see the position of the APP1 Exif block, the APP1 XMP block and the APP13 PhotoShop block which hosts the IPTC data. exiv2(.exe) also has an option -pX to extract the raw XMP for external processing. Both the Exif and Iptc data blocks are stored using the TIFF container specification. The example in the TIFF document illustrates how to extract and print the structure of the Exif data written in TIFF format. http://dev.exiv2.org/projects/exiv2/wiki/The_Metadata_in_TIFF_files

```
$ exiv2 -pa ~/test.jpg
Exif.Image.Make                              Ascii      18  NIKON CORPORATION
....
Exif.Thumbnail.YCbCrPositioning              Short       1  Centered
Iptc.Envelope.ModelVersion                   Short       1  4
Iptc.Envelope.CharacterSet                   String      3  %G
Iptc.Application2.RecordVersion              Short       1  4
Iptc.Application2.Caption                    String     27  Mrs Johnson's Austin Office
Xmp.xmp.Rating                               XmpText     1  0
Xmp.xmp.ModifyDate                           XmpText    25  2015-02-13T20:46:51-06:00
Xmp.dc.description                           LangAlt     1  lang="x-default" Mrs Johnson's Austin Office
$ exiv2 -pS ~/test.jpg
STRUCTURE OF JPEG FILE: /Users/rmills/test.jpg
 address | marker     | length  | data
       2 | 0xd8 SOI
       4 | 0xe1 APP1  |   14862 | Exif..II*......................
   14868 | 0xe1 APP1  |     699 | http://ns.adobe.com/xap/1.0/.<
   15569 | 0xe2 APP2  |    4094 | MPF.II*...............0100....
   19665 | 0xed APP13 |     110 | Photoshop 3.0.8BIM.......6....
   19777 | 0xdb DQT   |     132
   19911 | 0xc0 SOF0  |      17
   19930 | 0xc4 DHT   |     418
   20350 | 0xda SOS   |      12
837 rmills@rmillsmbp:~/gnu/exiv2/trunk/website $ exiv2 -pX ~/test.jpg | xmllint -pretty 1 -
<?xml version="1.0"?>
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="XMP Core 5.1.2">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description xmlns:xmp="http://ns.adobe.com/xap/1.0/" xmlns:dc="http://purl.org/dc/elements/1.1/" rdf:about=""
      <dc:description>
        <rdf:Alt>
          <rdf:li xml:lang="x-default">Mrs Johnson's Austin Office</rdf:li>
        </rdf:Alt>
      </dc:description>
    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
$
```

Note:

- XMP://ns.adobe.com/xmp/extension/ is not the valid XMP identifier string in Exiv2. The valid string should be http://ns.adobe.com/xap/1.0/\x00. Hence, XMP doesn't appear in the output.
- The test.jpg is attached at the bottom of this wiki. You can download and reproduce the above example easily.

References

1. Wikipedia. JPEG.	http://en.wikipedia.org/wiki/JPEG.
2. Wikipedia. Exif.	http://en.wikipedia.org/wiki/Exif.
3. Exif Spec Version 2.2. (Apr, 2002)	http://www.exif.org/Exif2-2.PDF

4. C-Cube Microsystems. JPEG File Interchange Format Version 1.02. (Sep, 1992).	http://www.w3.org/Graphics/JPEG/jfif3.pdf
5. Adobe. Adobe Photoshop File Formats Specification. (Jun, 2013).	http://www.adobe.com/devnet-apps/photoshop/fileformatashtml/
6. Coverpages.org. Extensible Metadata Platform (XMP).	http://xml.coverpages.org/xmp.html
7. W3C. RDF/XML Syntax Specification. (Feb, 2004).	http://www.w3.org/TR/REC-rdf-syntax/
8. CPAN. The structure of JPEG files.	http://search.cpan.org/dist/Image-MetaData-JPEG/lib/Image/MetaData/JPEG/Structures.pod
9. JPEG File Interchange Format File Format Summary.	http://www.fileformat.info/format/jpeg/egff.htm
10. Ehow.com. The Structure of a JPEG.	http://www.ehow.com/info_12202329_structure-jpeg.html

test.jpg - from http://www.flickr.com/photos/canderson/5470671552/ (284,572 KB) Tuan Nhu, 23 Jul 2013 09:28

fig2.jpg - Basic Structure of JPEG Files, From Exif.org (154,944 KB) Tuan Nhu, 24 Jul 2013 00:34

fig3.jpg - Flowchart of readMetadata (342,674 KB) Tuan Nhu, 24 Jul 2013 00:37

fig4.jpg - Flowchart of writeMetadata (442,927 KB) Tuan Nhu, 24 Jul 2013 00:38

fig2d.jpg (154,944 KB) Robin Mills, 17 Aug 2013 13:33

fig3d.jpg (70,383 KB) Robin Mills, 17 Aug 2013 13:33

fig4d.jpg (81,015 KB) Robin Mills, 17 Aug 2013 13:33