Exiv2 - Bug #533

Support multiple APP13 Photoshop 3.0 segments in a JPEG

28 Nov 2007 06:09 - Andreas Huggel

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Andreas Huggel	**% Done:**	100%
Category:	jpeg parser	**Estimated time:**	0.00 hour
Target version:	0.19		

Description

It appears there can be multiple APP13 Photoshop 3.0 segments in a JPEG. The payloads of these should be concatenated before processing and the other way around on write.

Additional information:

The original discussion in the digiKam bugzilla:
http://bugs.kde.org/show_bug.cgi?id=152210
(in particular from comment 13 onwards)

Related issues:

Related to Exiv2 - Bug #542: exiv2 doesn't find exif data in attacched file	**Closed**	
Related to Exiv2 - Bug #663: Failure to write to Jpeg from Canon Ixus - digik...	**Feedback**	**20 Dec 2009**

Associated revisions

Revision 1849 - 29 Jun 2009 08:24 - Andreas Huggel

#533: Applied patch 6_fix_test_conversions_timezone (Michael Ulbrich, Volker Grabsch)

Revision 1856 - 12 Jul 2009 06:22 - Andreas Huggel

#533: Added Volker Grabsch and Michael Ulbrich to authors.

Revision 1861 - 13 Jul 2009 08:02 - Andreas Huggel

#533: Changed two more warnings to debug messages (Volker Grabsch)

History

#1 - 09 Dec 2007 00:03 - Andreas Huggel

r1338: Read IPTC from multiple APP13 Photoshop 3.0 segments

#2 - 08 Oct 2008 03:35 - Andreas Huggel

Another suggestion: Process only the first APP13 segment.
http://uk.groups.yahoo.com/group/exiv2/message/1382

#3 - 10 Oct 2008 14:25 - Michael Ulbrich

Two situations where a JPEG file contains multiple APP13 segments have occured:

a) an IPTC block, which doesn't fit into a single APP13 segment has been split into multiple consecutive APP13 segments.

b) an application, instead of modifying IPTC data contained in an existing APP13 segment, has chosen to prepend a new APP13 segment with an independent IPTC block.

Both cases could be considered "broken" on the metadata level. The basic JPEG structure is intact.

Case a) owes it's size to the presence of IPTC extended dataset 2#202 "ObjectData Preview Data" with a maximum size of 256000 bytes. It is quite uncommon to put an image preview into the IPTC block, since there is image resource block #1036 which usually would contain the (much smaller!) thumbnail image.

Case b) originated from an image communication network where intermediate nodes may add a new IPTC block with credit / copyright information and keep the original IPTC data for reference.

exiv2 currently copes with a) but merges IPTC blocks according to b) into one.

The attempt to uniquely distinguish these cases and handle both correctly seems impossible since there is no known semantic of splitting APP13 segments.

An experimental patch to restrict exiv2's IPTC editing only to the first APP13 segment and leave all possibly following untouched has been provided. [Attachment: fix_jpg_parser.patch -ahu.]

Thanks + Best regards ... Michael

#4 - 19 Jun 2009 14:24 - Michael Ulbrich

- File kde_bug_152210_fixed.jpg added

Some remarks on how correct splitting of APP13 segments in JPEG images might look like.

I recently came across some JPEGs which our Exiv2-based Metadata-Tool was unable to process. On closer inspection these images had multiple APP13 segments due to the presence of multiple large IRBs (IDs 0x07D0 ff.) containing saved path information.

The following refers to above cited original discussion in the digiKam bugzilla: http://bugs.kde.org/show_bug.cgi?id=152210

In comment #14 Andreas attaches a JPEG (http://bugs.kde.org/attachment.cgi?id=22410) where the IPTC IRB sizes were fixed with respect to the original attachment in comment #11 (http://bugs.kde.org/attachment.cgi?id=22180).

The problem with the method of splitting as implemented in these DigiKam examples lies in the fact that it is done on the IPTC (IRB payload) level and not on the IRB level. An application which wishes to detect a split APP13 situation would have to understand IPTC to distinguish a valid split from a broken metadata structure. Actually there are more types of IRBs than just IPTC with differently formatted payloads, which potentially might need splitting among APP13 segments. An application would hence need knowledge about all these payload formats to securely distinguish splitting from corruption.

In example 22410 both APP13 segments decode cleanly on the Image Resource Block level since all the IRB lengths are correct. The decoder has to descend to the IPTC level to find tag 0xca (ObjectPreviewData) with a length pointing beyond the end of the first APP13 segment. The following APP13 segment then starts with the standard preamble "Photoshop 3.0" and again signals an IPTC IRB by "8BIM" 0x0404 followed by the correct length of the second IPTC fragment. The problem here is that a fresh IPTC IRB is opened which soon violates the format by not starting with a valid IPTC tag 0x1c02 but appending the second fragment of tag 0xca instead. BTW example 22410 leads to "Invalid IPTC entry (tag 202, len 68198)" from the first and "Warning = Bad IPTC data tag (marker 0x28)" from the second IPTC fragment, if parsed by exiftool.

Now back to above mentioned JPEGs with path information IRBs split across consecutive APP13 segments. Here the length of the split IRB contains the total - not the fragment - length and points beyond the end of the first APP13 segment. The second APP13 segment starts neither with the preamble nor with a fresh IRB but instead immediately continues with the second fragment of the split IRB. A parser will be able to detect the split solely on the IRB level without descending into payload semantics. Second and higher APP13 markers without preamble / initial 8BIM will give a strong indication of splitting. After concatenating these split APP13 segments parse cleanly on the IRB level. If not, the metadata structure is corrupt.

I have modified example 22410 and split the APP13 markers as described above. The image is attached to this message.

A set of patches is available for reading/writing JPEG APP13 segments according to this method. These patches will be posted together with comments either to the forum or this bug report in the next days.

#5 - 25 Jun 2009 06:11 - Michael Ulbrich

- File exiv2_jpg_patches_version5.tgz added

As already announced here's a set of patches, which implements support for multiple APP13 Photoshop 3.0 segments in JPEG according to previous post.

These are the patches with comments:

1_handle_empty_IRB indep*

Currently empty IRB at end of APP13 is treated as error. This patch changes behaviour
insofar that trailing empty IRBs are treated as normal image resource blocks.

Per IPTC spec no empty IPTC IRBs are allowed but they occur in practice anyway.
size = 12 -> empty IRB
size = 14 -> length 1 + padding '0'

In DEBUG mode empty IRBs will be reported.

2_read_and_modify_only_the_first_XMP_segment indep*

Current implementation modifies last XMP segment, if multiple segments are present.
Change behaviour to only modify first XMP segment and leave the following untouched.

3_skip_writing_redundant_IPTC_IRBs indep*?

Multiple IPTC IRBs inside same APP13 segment (theoretical).

Current implementaion concatenates IPTC IRB contents on read but replaces only first with concatenated IPTC content on write, leaves the following untouched.

```
APP13                           APP13
        IRB-1                           IRB-1
        IRB-2 (IPTC)                    IRB-2 (IPTC)
            IPTC-a                          IPTC-a*
        IRB-3                               IPTC-b*
        IRB-4               ----->          IPTC-c*
        IRB-5 (IPTC)                    IRB-3
            IPTC-b                      IRB-4
        IRB-6 (IPTC)                    IRB-5 (IPTC)
            IPTC-c                          IPTC-b
        IRB-7                           IRB-6 (IPTC)
                                            IPTC-c
                                        IRB-7
```

Now concatenated IPTC IRB replaces first IPTC IRB. Following IPTC IRBs are discarded. Non IPTC IRBs are kept and not modified.

```
APP13                           APP13
        IRB-1                           IRB-1
        IRB-2 (IPTC)                    IRB-2 (IPTC)
            IPTC-a                          IPTC-a*
        IRB-3                               IPTC-b*
        IRB-4               ----->          IPTC-c*
        IRB-5 (IPTC)                    IRB-3
            IPTC-b                      IRB-4
        IRB-6 (IPTC)                    IRB-7
            IPTC-c
        IRB-7
```

4_new_function_Photoshop_valid (needed by 8)

Used to detect extended (split) IRBs. Checks all IRBs in APP13 for correct length. Returns false, if length mismatch was found.

5_recognize_small_corrupt_IRBs indep*

Related to 1. Handle (corrupt) IRBs shorter than 12 byte as error. Additionally needed to correctly locate split boundary.

6_fix_test_conversions_timezone indep*

Set correct timezone for test. 'make test' currently fails, if local time zone not GMT-8

7_more_iptc_test_images (needs all patches to succeed all tests)

Add synthetically generated JPEG images with various IPTC stuctures for testing. Tarball with test images.

8_handle_extended_Photoshop_IRBs

support multiple (split) APP13 Photoshop 3.0 segments in JPEG. On write split APP13 segments on 65533 byte boundary. If split boundary coincides with end of IRB, decrease segment size to force IRB split (detectable!).

Additional remarks:

1. Example JPEG images with split APP13 segments written by Photoshop were found.
In these cases not IPTC IRBs with embedded preview image, but multiple IRBs containing complex clipping paths were responsible for APP13 size exceeding limit.

2. Actually APP13 segments in above example JPEGs were split on 32000 byte boundary and not on the expected 65533. On write Exiftool uses the latter as well.

3. Patches 1, 2, 3, 5, 6 are independent (indep*). Patch 4 and test images are needed by patch 8. Patches apply cleanly against exiv2-0.18.2. All tests succeed.

4. If patches are accepted for inclusion, please add Volker Grabsch (vogATnotjusthostingDOTcom) as contributor (in fact he did all the coding ...).

Thanks to all for exiv2!

Best regards ... Michael

#6 - 28 Jun 2009 09:42 - Andreas Huggel

Thanks for this very comprehensive analysis and patchset!

I've checked-in all patches except for 6_* with r1842 - r1848 (but forgot the # in front of the issue number in the commit message), without much testing. I'll have a closer look at 6_* as soon as I can. The output of "cd test; make" now contains several warnings, is that intended? (see below).

As usual, the copyright question: Can I add you to the list of authors of the modified files but retain the copyright for the complete files myself?

Andreas

```
Running addmoddel.sh ...
All testcases passed.
Running bugfixes-test.sh ...
Files ./tmp/bugfixes-test.out-stripped and ./data/bugfixes-test.out differ
265,266d264
< Warning: Invalid Photoshop IRB data size 144775 or extended Photoshop IRB
< Warning: Invalid Photoshop IRB data size 144775 or extended Photoshop IRB
Running exifdata-test.sh ...
All testcases passed.
Running exiv2-test.sh ...
All testcases passed.
Running imagetest.sh ...

Erase all tests.......Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
.........
Copy all tests.........
Copy iptc tests.........
------------------------------------------------------------
All test cases passed
Running iotest.sh ...

Io tests...
------------------------------------------------------------
All test cases passed
Running iptctest.sh ...

Read tests....Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
............
Remove tests....Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
............
Add/Mod tests....Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
............
Extended tests....Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 33034 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 33034 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
.Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 25 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
Warning: Invalid Photoshop IRB data size 3978 or extended Photoshop IRB
............
------------------------------------------------------------
All test cases passed
```

```
Running modify-test.sh ...
All testcases passed.
Running path-test.sh ...
Running stringto-test.sh ...
All testcases passed.
Running tiff-test.sh ...
All testcases passed.
Running write-test.sh ...
All testcases passed.
Running write2-test.sh ...
All testcases passed.
Running xmpparser-test.sh ...
All testcases passed.
Running conversions.sh ...
All testcases passed.
```

#7 - 29 Jun 2009 19:10 - Andreas Huggel

> 6_fix_test_conversions_timezone indep*
>
> Set correct timezone for test. 'make test' currently fails, if local time zone not GMT-8

Checked-in with r1849 (this time with #). I like the "elegant hack" used to fix this, thanks!
PS: Local timezone here is GMT+8, not -8

Andreas

#8 - 30 Jun 2009 03:56 - Michael Ulbrich

> The output of "cd test; make" now contains several warnings, is that intended?

For the moment, yes. Without patch 8 applied, any mismatch in claimed versus actual IRB size leads to an "Error: Invalid Photoshop IRB data size XXXX". We've changed this to "Warning: Invalid Photoshop IRB data size XXXX or extended Photoshop IRB" since a mismatch in IRB size may now resolve later in case of extended/split APP13 segments.

We see two possibilities to get rid of the warnings:

1. Completely eliminate the warnings and only report an error if a split APP13 situation cannot be resolved.

2. Change condition from "#ifndef SUPPRESS_WARNINGS" to "#ifdef DEBUG".

> Can I add you to the list of authors of the modified files but retain the copyright for the complete files myself?

The answer is "yes" for Volker as well as for me.

... Michael

#9 - 30 Jun 2009 04:22 - Andreas Huggel

Michael Ulbrich wrote:

> We see two possibilities to get rid of the warnings:
>
> 1. Completely eliminate the warnings and only report an error if a split APP13 situation cannot be resolved.

Yes, I think that is the correct behavior.

> > Can I add you to the list of authors of the modified files but retain the copyright for the complete files myself?
>
> The answer is "yes" for Volker as well as for me.

Thank you. I'll update the source files.

Andreas

#10 - 30 Jun 2009 04:29 - Andreas Huggel

Michael, from your experience, what applications can deal with this kind of split APP13 segments (other than exiftool)?

#11 - 01 Jul 2009 01:52 - Michael Ulbrich

Andreas Huggel wrote:

> Michael, from your experience, what applications can deal with this kind of split APP13 segments (other than exiftool)?

Hmm, we are mostly modifying IPTC/XMP metadata of existing images, which come from a broad range of sources. Before we switched to exiv2 we had a tool based on 'iptcutil', which can still be found in the libtiff distribution.

From that perspective we do not handle a lot of metadata on the application level, but within our own software environment. OTOH people who work with our modified images, mostly use the tools from A***e. So compatibility with these apps is always a must.

Exiftool is able to process split APP13 segments, but does AFAIR not distinguish between a true split and multiple APP13 segments that have been accidentally written.

ImageMagick seems unable to handle any form of split APP13, neither the "old" method with split on the IPTC payload level nor the "new" with split on the IRB level. IM IPTC metadata handling still seems to be based on modified iptcutil code.

Please let me know, if there are any other relevant - open source - apps, which are aware of image metadata. I may then check how they behave in these special cases.

#12 - 02 Jul 2009 04:32 - Michael Ulbrich

Andreas Huggel wrote:

> Michael Ulbrich wrote:
>
> > We see two possibilities to get rid of the warnings:
> >
> > 1. Completely eliminate the warnings and only report an error if a split APP13 situation cannot be resolved.
>
> Yes, I think that is the correct behavior.

From our POV it would be better to be able to reactivate the warnings by defining the DEBUG symbol and recompile. There is already other DEBUG output from Photoshop::locateIrb() in jpgimage.cpp. In case of troubleshooting some strange or broken input format (which I expect to show up), this would be helpful.

#13 - 02 Jul 2009 20:27 - Andreas Huggel

Reminder for myself: Review the test which seems to filter large IPTC previews:

http://bugs.kde.org/show_bug.cgi?id=152210#c21

#14 - 12 Jul 2009 06:40 - Andreas Huggel

Michael Ulbrich wrote:

> From our POV it would be better to be able to reactivate the warnings by defining the DEBUG symbol and recompile. There is already other DEBUG output from Photoshop::locateIrb() in jpgimage.cpp. In case of troubleshooting some strange or broken input format (which I expect to show up), this would be helpful.

r1858

#15 - 12 Jul 2009 06:44 - Andreas Huggel

Andreas Huggel wrote:

> Reminder for myself: Review the test which seems to filter large IPTC previews:
>
> http://bugs.kde.org/show_bug.cgi?id=152210#c21

The test is in order. It doesn't filter just large IPTC previews.

#16 - 12 Jul 2009 06:45 - Andreas Huggel

- Status changed from Assigned to Resolved

- Target version set to 0.19

- % Done changed from 0 to 100

#17 - 13 Jul 2009 04:53 - Volker Grabsch

- File change_Photoshop_IRB_warning_to_debug.patch added

Thanks for r1858.

However, the two other warnings "Invalid or extended Photoshop IRB" should be converted to DEBUG messages, too, since they may also appear in a valid split APP13.

Attached is a patch which solves the problem.

Greets,
Volker

#18 - 13 Jul 2009 08:03 - Andreas Huggel

> However, the two other warnings "Invalid or extended Photoshop IRB" should be converted to DEBUG messages, too, since they may also appear in a valid split APP13.
>
> Attached is a patch which solves the problem.

Patch checked-in. Thanks.

#19 - 30 Dec 2009 07:50 - Andreas Huggel

- Status changed from Resolved to Closed

Files

fix_jpg_parser.patch	9.78 KB	10 Oct 2008	Redmine Admin
kde_bug_152210_fixed.jpg	890 KB	19 Jun 2009	Michael Ulbrich
exiv2_jpg_patches_version5.tgz	145 KB	25 Jun 2009	Michael Ulbrich
change_Photoshop_IRB_warning_to_debug.patch	758 Bytes	13 Jul 2009	Volker Grabsch