Exiv2 - Bug #1221

Export DNG to JPEG issues - missing metadata

05 Sep 2016 18:41 - Wil Cowb

Status:	Closed	**Start date:**	05 Sep 2016
Priority:	Normal	**Due date:**	
Assignee:	Robin Mills	**% Done:**	100%
Category:	not-a-bug	**Estimated time:**	2.00 hours
Target version:	0.26		

Description

JPEGs created from DNGs have no metadata from the original file. Camera, date taken, shutter speed, aperture, exposure, ISO, etc. everything is gone.

Sample files can be downloaded here:
Raw - https://drive.google.com/file/d/0B5_iknSPeSNBMVBIT0wxNXowRE0/view?usp=sharing
JPEG with the problem - https://drive.google.com/file/d/0B5_iknSPeSNBUkJ0VmNadmhMT1k/view?usp=sharing
XMP - https://drive.google.com/file/d/0B5_iknSPeSNBTkcteUlsV2ZHM2M/view?usp=sharing

Tested with two versions of Exiv2:

1:
dpkg -l | grep -i exiv2
ii exiv2 0.25-xenial~ppa2 amd64 EXIF/IPTC metadata manipulation tool
ii libexiv2-14:amd64 0.25-xenial~ppa2 amd64 EXIF/IPTC metadata manipulation library
ii libgexiv2-2:amd64 0.10.3-2 amd64 GObject-based wrapper around the Exiv2 library
ii python-pyexiv2 0.3.2-5ubuntu4build2 amd64 Python binding to Exiv2
ii python-pyexiv2-doc 0.3.2-5ubuntu4build2 all Documentation for Python binding to Exiv2

2:
dpkg -l | grep -i exiv2
ii exiv2 0.25-2.1pmjdebruijn1~xenial amd64 EXIF/IPTC metadata manipulation tool
ii libexiv2-14:amd64 0.25-2.1pmjdebruijn1~xenial amd64 EXIF/IPTC metadata manipulation library
ii libgexiv2-2:amd64 0.10.3-2 amd64 GObject-based wrapper around the Exiv2 library
ii python-pyexiv2 0.3.2-5ubuntu4build2 amd64 Python binding to Exiv2
ii python-pyexiv2-doc 0.3.2-5ubuntu4build2 all Documentation for Python binding to Exiv2

Tested with two types of DNG files:
1. CR2/PEF converted to DNG via Adobe DNG converter
2. DNG created by Pentax camera itself

Software package used - darktable 2.0.5

Similar thread on the darktable bug report platform:
https://redmine.darktable.org/issues/11130

No errors can be seen in the terminal on opening darktable or exporting to JPEG

History

#1 - 05 Sep 2016 20:40 - Robin Mills

- Category set to not-a-bug

- Status changed from New to Assigned

- Assignee set to Robin Mills

- Target version set to 0.26

- % Done changed from 0 to 100

- Estimated time set to 2.00 h

Andrey

Thank you for report this. I like the dog in the image.

I don't believe this is an Exiv2 issue. Exiv2 does not have code to convert DNG to JPEG. And you've said that you are using ***darktable*** (which uses libexiv2 to do many things). So my speculation is that you have used ***darktable*** to successfully create a JPEG. The resulting JPG contains an image with no metadata.

The exiv2 command-line tool can copy metadata from a DNG to a JPEG. However when I run the command to extract the metadata, there is an error message:

```
1320 rmills@rmillsmbp:~/Downloads $ exiv2 -ea --verbose --force IMGP0385.dng
File 1/1: IMGP0385.dng
Writing Exif data from IMGP0385.dng to ./IMGP0385.exv
Writing XMP data from IMGP0385.dng to ./IMGP0385.exv
Warning: Exif tag Exif.Image.DNGPrivateData not encoded
Warning: Exif tag Exif.Image.ProfileLookTableData not encoded
./IMGP0385.exv: Could not write metadata to file: Size of Exif JPEG segment is larger than 65535 bytes
```

The reason for the message is because there is too much Exif data to be stored in the JPEG. No JPEG can ever store more than 64k of Exif metadata because of the design of the JPEG container. All Exif data must reside within a single APP1 segment which is limited to 64k bytes.

Your DNG seems to contain too much Exif metadata to be stored in a JPEG. DNGs are Tiff files and have a limit of 1g of Exif data.

How can this be solved? Some application (such as DNG Creator or ***darktable***) has to filter the metadata to a limited subset in the JPEG. ***darktable*** can use the APIs in libexiv2 to delete uninteresting metadata to obtain a subset that fits into a 64k segment.

There is something that can be quickly and easily done on the command-line, or by ***darktable***. The DNG does contain XMP metadata. The XMP can be extracted from the DNG using the command ***$ exiv2 -pX foo.dng > foo.xmp*** The XMP can be inserted into a JPEG use the exiv2 v0.26 command ***$ exiv2 -iXX foo.jpg***

Putting the whole thing together with your files:

```
1344 rmills@rmillsmbp:~/Downloads $ exiv2 -pa IMGP0385_01.jpg
1345 rmills@rmillsmbp:~/Downloads $                                    # empty
1345 rmills@rmillsmbp:~/Downloads $                                    # no metadata
1345 rmills@rmillsmbp:~/Downloads $ exiv2 -pX IMGP0385.dng > IMGP0385_01.xmp  # extract the XMP from the DNG
1346 rmills@rmillsmbp:~/Downloads $ exiv2 -iXX IMGP0385_01.jpg          # put the XMP into the JPG
1347 rmills@rmillsmbp:~/Downloads $ exiv2 -pa IMGP0385_01.jpg           # a little happiness
Xmp.xmp.CreatorTool                         XmpText    33  Adobe DNG Converter 9.5 (Windows)
Xmp.xmp.ModifyDate                          XmpText    25  2016-03-26T10:20:43-06:00
Xmp.xmp.CreateDate                          XmpText    19  2016-02-13T05:03:45
Xmp.xmp.MetadataDate                        XmpText    25  2016-03-26T10:20:43-06:00
Xmp.aux.LensInfo                            XmpText    25  180/10 550/10 35/10 56/10
Xmp.aux.Lens                                XmpText    35  smc PENTAX-DA L 18-55mm F3.5-5.6 AL
Xmp.aux.LensID                              XmpText     5  7 222
Xmp.aux.ApproximateFocusDistance            XmpText     7  200/100
Xmp.photoshop.DateCreated                   XmpText    19  2016-02-13T05:03:45
Xmp.xmpMM.DocumentID                        XmpText    44  xmp.did:a00face5-8d33-ea44-ad2e-565ea608fd60
Xmp.xmpMM.OriginalDocumentID                XmpText    32  3332781E8A87190381750F32CF4AD870
Xmp.xmpMM.InstanceID                        XmpText    44  xmp.iid:a00face5-8d33-ea44-ad2e-565ea608fd60
Xmp.xmpMM.History                           XmpText     0  type="Seq"
Xmp.xmpMM.History[1]                        XmpText     0  type="Struct"
Xmp.xmpMM.History[1]/stEvt:action           XmpText     7  derived
Xmp.xmpMM.History[1]/stEvt:parameters       XmpText    69  converted from image/x-pentax-raw to image/dng, sa
ved to new location
Xmp.xmpMM.History[2]                        XmpText     0  type="Struct"
Xmp.xmpMM.History[2]/stEvt:action           XmpText     5  saved
Xmp.xmpMM.History[2]/stEvt:instanceID       XmpText    44  xmp.iid:a00face5-8d33-ea44-ad2e-565ea608fd60
Xmp.xmpMM.History[2]/stEvt:when             XmpText    25  2016-03-26T10:20:43-06:00
Xmp.xmpMM.History[2]/stEvt:softwareAgent    XmpText    33  Adobe DNG Converter 9.5 (Windows)
Xmp.xmpMM.History[2]/stEvt:changed          XmpText     1  /
Xmp.xmpMM.DerivedFrom                       XmpText     0  type="Struct"
Xmp.xmpMM.DerivedFrom/stRef:documentID      XmpText    32  3332781E8A87190381750F32CF4AD870
Xmp.xmpMM.DerivedFrom/stRef:originalDocumentID XmpText    32  3332781E8A87190381750F32CF4AD870
Xmp.dc.format                               XmpText     9  image/dng
1348 rmills@rmillsmbp:~/Downloads $
```

Regretfully, I do not believe Team Exiv2 can be of further help to you in this matter. I won't close this issue as you may wish to discuss it further.

When you discuss this matter with ***darktable***, can I bring to your attention that XMP is also limited to 64k bytes. I see that ***darktable*** is storing the file revision history in the XMP. That could quickly challenge the 64k XMP limit. Adobe have added a mechanism called "ExtendedXMP" to enable XMP to exceed that limit. Support for "ExtendedXMP" is limited in Exiv2 v0.26. It is also limited in many other applications. Editing an image in many editors will ignore and delete "ExtendedXMP".

Robin

#2 - 05 Sep 2016 20:58 - Roman Lebedev

Robin Mills wrote:
Not commenting on the content right now, but *please* don't call darktable as DarkTable, dark table, Dark Table, etc.
It's darktable.

#3 - 05 Sep 2016 22:23 - Robin Mills

I've edited the post to consistently say ***darktable***.

#4 - 06 Sep 2016 16:38 - Matthieu Volat

After a quick discussion with Roman where this issue was mentionned: I believe the issue is the file is already heavily loaded with exif data, and with the one darktable try to add, size grows too large(66706 bytes using git version) and darktable throw away the exif data when exporting.

#5 - 06 Sep 2016 17:43 - Robin Mills

- Status changed from Assigned to Closed

#6 - 11 Sep 2016 10:07 - Roman Lebedev

Thanks to Matthieu Volat, darktable now filters a bit more tags:
https://github.com/darktable-org/darktable/pull/1256
https://github.com/darktable-org/darktable/pull/1259

For now it did fix this particular issue, but i bet there are more cases.

#7 - 11 Sep 2016 10:49 - Robin Mills

Roman

Yes. Thanks for giving me an update on this. Well done ***Matthieu*** for filtering out the junk from the DNG that can't fit into the 64k Exif/JPEG restriction. I suspect there will be more of these.

I think libexiv2 image->writeMetadata() throws an exception when the 64k limit is exceeded. ***darktable*** could catch that and do one of a couple of things:

1. Warn the user that the metadata has been lost (not useless, however not very helpful)
2. Remove more and more metadata until what remains fits

To implement this second concept, ***darktable*** requires a list of *deplorable* keys and keeps removing these *deplorables* until writeMetadata() succeeds.

This result is better than the current behaviour of removing all Exif metadata. An unpleasant side effect is that some deplorables_ will be sometimes written and sometimes not - it depends on the company they keep!

#8 - 26 Sep 2016 19:35 - Robin Mills

- File XmpPart3-2016-page13.png added

Folks:

I've made a discovery about this!

Adobe published a new Edition of the XMPsdk on 30 July 2016. The latest version of this document XMP-Toolkit-SDK-CC201607/docs/XMPSpecificationPart3.pdf contains this statement on page 13. This statement was not in the 2010 Edition of that document.

> The JPEG standard does not prescribe ordering among APPn segments, but some related standards do. For example, Exif requires that Exif APP1 segment be immediately after the SOI. Also, some applications improperly assume that the segments are in a particular order. For compatibility, it is best to put the Exif APP1 first, the XMP APP1 next, the PSIR APP13 next, followed by all other marker segments.
>
> ***NOTE*** If the size of the Exif APP1 marker or the PSIR APP13 marker exceeds 64KB, the marker is split into multiple blocks of 64KB size each.

XmpPart3-2016-page13.png

If this is true, we can indeed store huge blobs of Exif metadata. If the metadata exceeds 64k, we can simply divide it into multiple segments. However, I would want to see a reliable example of such a file. By "reliable", I mean it has a blue-chip pedigree (such as PhotoShop). And I'd like to see a specification which is focused on this matter. The statement in this spec doesn't feel definitive and doesn't address APP1 continuation block signatures.

A typical JPEG structure is as follow:

```
696 rmills@rmillsmbp:~/gnu/exiv2/trunk $ exiv2 -pS test/data/Reagan.jpg
STRUCTURE OF JPEG FILE: test/data/Reagan.jpg
 address | marker       |  length | data
       0 | 0xffd8 SOI
```

```
      2 | 0xffe1 APP1  |    5718 | Exif..MM.*.....................     <---- This is the Exif data (Tiff Enc
oded)
    5722 | 0xffed APP13 |    3038 | Photoshop 3.0.8BIM.........Z...
    8762 | 0xffe1 APP1  |    5329 | http://ns.adobe.com/xap/1.0/.<?x
   14093 | 0xffe2 APP2  |     576 | ICC_PROFILE......0ADBE....mntrRG chunk 1/1
   14671 | 0xffee APP14 |      14 | Adobe.d@......
   14687 | 0xffdb DQT   |     132
   14821 | 0xffc0 SOF0  |      17
   14840 | 0xffdd DRI   |       4
   14846 | 0xffc4 DHT   |     418
   15266 | 0xffda SOS
697 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

I think Adobe are saying that if you follow an APP1 segment with one or more APP1 segments of ***exactly*** 64k bytes you should consider it to be a "segmented" continuation of its predecessor. That seems reasonable to me. However it's quite different from the "chunk" mechanism used to allow ICC profiles to exceed a single APP2 segment. Additionally, APP1 segments begin with a ***signature*** such as "Exif\0\0" to identify them to readers.

I've hunted through Phil (of ExifTool fames) repository of 7000 JPG files. Not one has Exif data encoded in this way.

AGFA have files with multiple segments of length 65535. However they are not APP1 segment. Here's the evidence:

```
702 rmills@rmillsmbp:~/gnu/exiv2/trunk $ exiv2 -pS /mmHD/Users/rmills/Jenkins/testfiles/Phils/Agfa/AgfaOPTIMA1
338mT.jpg
STRUCTURE OF JPEG FILE: /mmHD/Users/rmills/Jenkins/testfiles/Phils/Agfa/AgfaOPTIMA1338mT.jpg
 address | marker       |  length | data
       0 | 0xffd8 SOI
       2 | 0xffe1 APP1  |   46459 | Exif..II*.....................
   46463 | 0xffe3 APP3  |   65535 | ................................   <--- what is this?
  112000 | 0xffe4 APP4  |   65535 | ..Hc..w .8<...z..M.77.h...{.....   <--- or this?
  177537 | 0xffe5 APP5  |    7243 | .U......K..u=).pl.W.F...B.$.3...
  184782 | 0xffdb DQT   |     132
  184916 | 0xffc0 SOF0  |      17
  184935 | 0xffc4 DHT   |      75
  185012 | 0xffda SOS
703 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

I'm willing to implement "Exif data exceeds 64k" feature in the future ***provided*** we have a more substantial specification and good pedigree test files.

I've opened a new Exiv2 v1.0 Feature Request for this: #1232.

Files

XmpPart3-2016-page13.png	142 KB	26 Sep 2016	Robin Mills