Exiv2 - Bug #1380

Iptc.Application2.DateCreated and Iptc.Application2.TimeCreated date formats are different in read/write

14 Nov 2020 23:59 - Jehan Pagès

Status:	New	**Start date:**	14 Nov 2020
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			

Description

Version of Exiv2: 0.27.3 (and tested HEAD of 0.27-maintenance branch too)

1. Description of problem

A date and time can be written with Exiv2 in IPTC (tags Iptc.Application2.DateCreated and Iptc.Application2.TimeCreated respectively) using a specific format "2020-11-15" and "HH:MM:SS-HH:MM" but when read back (still with Exiv2), it expects another format ("20201115" and "HHMMSS:HHMM" which corresponds to what you wrote here: https://exiv2.org/iptc.html) resulting in following errors being returned: "Unsupported date format" and "Unsupported time format".

Note that if I try to set instead with "20201115" and "HHMMSS:HHMM" format instead, then I get the same errors but this time when setting the IPTC tag. It really seems like the API is not consistent on write and read.

1. More info

Looking quickly into exiv2 code, for instance in `src/value.cpp`, I can see that DateValue::read(const byte* buf, long len, ByteOrder /*byteOrder*/) definitely expects a string of 8-byte string with format "%4d%2d%2d" (similarly for TimeValue::read()). This is the code we hit when we previously set date/time in another format (also with exiv2, I remind) hence get kerUnsupportedDateFormat and kerUnsupportedTimeFormat errors.

But now when I set a new tag, Iptcdatum::setValue(const std::string& value) calls DateValue::read(const std::string& buf) which has the format "%4d-%d-%d".

So basically you have 2 versions of read() for DateValue and TimeValue, and they do something different. And as it turns out, we hit one of the version when writing a new tag, the other when reading tags from existing files. Therefore Exiv2 fails to read some tags it created itself.

src/pngimage.cpp

History

#1 - 15 Nov 2020 01:08 - Jehan Pagès

So actually I thought I might as well just fix this myself. My solution is to have more robust code by accepting both formats (for date and time). Maybe some variants may not be technically right according to specs, but since they are so easily distinguishable, it's probably better to accept them all and simply parse them accordingly.

Also this will allow existing code using the current implementation to not suddenly break because format changed in one or another function.

So anyway, attached is the patch.

#2 - 15 Nov 2020 01:09 - Jehan Pagès

- File 0001-More-robust-date-and-time-format-read.patch added

Oups, previous comment failed to attach the patch somehow. Retrying.

Files

0001-More-robust-date-and-time-format-read.patch	4.94 KB	15 Nov 2020	Jehan Pagès