Exiv2 - Bug #662

Incorrect Unicode encoding of Exif UserComment tag

18 Dec 2009 09:46 - Leo Sutic

Status:	Closed	**Start date:**	18 Dec 2009
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:	metadata	**Estimated time:**	0.00 hour
Target version:	0.20		

Description

In the Exif UserComment tag, the characters may be encoded as Ascii, Unicode, JIS or "Undefined".

This bug concerns the encoding when choosing Unicode encoding.

Exiv2 uses UTF-8. Exiftool, Windows and Microsoft Photo Info use UCS-2 and decode UserComments written by exiv2 as a jumble of glyphs. Exiv2, OTOH, can't decode the tags written by the aforementioned programs. There is a problem here.

So, which one is right?

The Exif 2.1 spec references the "Unicode Standard, The Unicode Consortium, 1991, Addison-Wesley". This is the 1.0.0 version of the Unicode spec, and back then UCS-2 was the default encoding. The Exif 2.2 spec makes no reference to any specific version of the Unicode standard, but we can assume that it is the intention of the Exif standards body to make Exif 2.2 backwards compatible with Exif 2.1.

I would therefore say that it is Exiv2 that is in error here.

If the Exiv2 team finds this argument valid, we need a short and a long term solution. The short term solution is intended to bridge the gap between now and until all images whose UserComment fields have been written as UTF-8 have been converted to UCS-2. The long term solution is to completely switch to UCS-2.

My suggested short-term solution is this:

1. This only applies to UserComment tags marked "Unicode".
2. Make Exiv write UCS-2 tags, always. Optionally, allow the user to apecify a "UTF-8"-encoding, in case someone really needs it.
3. On read, decode the 8-byte charset specifier. If it is Unicode:
    1. Look at the tag size. If odd, it can't be UCS-2 (as it is a fixed-size, 2-byte ecoding). Decode as UTF-8.
    2. Look for a zero byte in the text. If one is found, the text can't be UTF-8, as that encoding has no zero bytes, and the UserComment field isn't null-terminated. Decode as UCS-2.
    3. If no zero bytes, and even tag length - check for any bytes > 0x7f or <= 0x8. If none, decode as UTF-8, as it is likely to be an Ascii comment written as UTF-8. (The limits have been chosen to allow everything from tab to the end of Ascii.)
    4. Decode as UCS-2.

There will always be corner cases, but right now exiv2 UserComment tags can't be read by any other Exif viewer, so the fact that this hasn't been changed yet tells me that people don't use exiv2 to write UserComment tags all that much. We are therefore trading failure on corner cases of a feature that isn't used all that much against having that feature work **at all** with other programs. I think the trade-off it worth it.

Additionally:

1. A conversion tool that reads UTF-8 and writes UCS-2 could be created.
2. Perhaps some parameter could be passed to the parser to force parsing of UserComment as UTF-8 or UCS-2, for the corner cases.

I'd send a patch, but I'd like to run this past you first to see if there is any interest in accepting such a patch, should I write it, before I go through the work of doing it.

Related issues:

Related to Exiv2 - Bug #708: On Windows, convertStringCharset() should use re...	**Closed**	**28 May 2010**
Is duplicate of Exiv2 - Bug #562: Exif.Photo.UserComment unicode comment does...	**Closed**	

Associated revisions

Revision 1998 - 07 Jan 2010 08:55 - Andreas Huggel

Published convertStringCharset() in the API (for #662).

Revision 2000 - 12 Jan 2010 06:06 - Andreas Huggel

#662: Patch exiv2-exifcomment-unicode.patch from Leo Sutic (unmodified, without exiv2-bug662.jpg).

Revision 2001 - 12 Jan 2010 08:29 - Andreas Huggel

#662: Mostly formatting changes and a few tweaks. Move exifcomment tests to bugfixes-test.sh

Revision 2002 - 12 Jan 2010 09:07 - Andreas Huggel

#662: Updated expected test results.

Revision 2003 - 13 Jan 2010 18:26 - Andreas Huggel

#662: Fixes by Leo Sutic. Added carriage return to the special characters.

Revision 2005 - 14 Jan 2010 08:54 - Andreas Huggel

#662: Charset conversion on read and write (and if needed on copy).

Revision 2006 - 15 Jan 2010 03:29 - Andreas Huggel

#662: Added CommentValue::detectCharset and an optional parameter for the encoding to CommentValue::comment().

Revision 2011 - 19 Jan 2010 06:04 - Andreas Huggel

#662: Code tweak and updated expected test results.

Revision 2013 - 20 Jan 2010 04:07 - Andreas Huggel

#662: Detect and interpret a BOM.

Revision 2027 - 12 Feb 2010 07:37 - Andreas Huggel

#662: Added new option -n and action fixcom to exiv2 utility.

History

#1 - 18 Dec 2009 17:46 - Andreas Huggel

Leo,

Thanks for your thoughts.

> In the Exif UserComment tag, the characters may be encoded as Ascii, Unicode, JIS or "Undefined".
>
> This bug concerns the encoding when choosing Unicode encoding.

Technically, this is a duplicate of #562. I'll comment here but we should eventually close this bug as a duplicate.

> Exiv2 uses UTF-8.

Exiv2 actually doesn't do any conversion at all. It just writes the string that the applications/user provides to the field. So it's up to them to do the right thing for now.

> I would therefore say that it is Exiv2 that is in error here.

Yes. Exif UserComment tags with charset set to UNICODE should be encoded in UCS-2.

> If the Exiv2 team finds this argument valid, we need a short and a long term solution. The short term solution is intended to bridge the gap between now and until all images whose UserComment fields have been written as UTF-8 have been converted to UCS-2. The long term solution is to completely switch to UCS-2.

Since we don't have control over what happens to images which have already been modified, we can aim for the long term solution straight away. The concern is that applications which use Exiv2 should continue to function if possible, in particular if they already convert the comments correctly.

1. This only applies to UserComment tags marked "Unicode".
2. Make Exiv write UCS-2 tags, always.

Ok. Note that there is some existing code to convert between UCS-2 and UTF-8, used for certain Windows XP tags. There is also code used to convert between XMP and IPTC datasets, which detects a charset and converts to UTF-8 (r1908). Aim to reuse existing code and generalize where needed.

> Optionally, allow the user to apecify a "UTF-8"-encoding, in case someone really needs it.

Not desirable. Instead provide backward compatibility similar to what was done for #571 with r1908.

1. On read, decode the 8-byte charset specifier. If it is Unicode:
    1. Look at the tag size. If odd, it can't be UCS-2 (as it is a fixed-size, 2-byte ecoding). Decode as UTF-8.
    2. Look for a zero byte in the text. If one is found, the text can't be UTF-8, as that encoding has no zero bytes, and the UserComment field isn't null-terminated. Decode as UCS-2.
    3. If no zero bytes, and even tag length - check for any bytes > 0x7f or <= 0x8. If none, decode as UTF-8, as it is likely to be an Ascii comment written as UTF-8. (The limits have been chosen to allow everything from tab to the end of Ascii.)
    4. Decode as UCS-2.

Consider something similar to what Exiftool does. It writes the UserComment tag with an Exif character code
"ASCII" if the text consists of only 7-bit characters, else it uses the Exif
character code "UNICODE" and encodes the text in UTF-16.
It encodes the UTF-16 string using the same byte order as the rest of the
Exif/TIFF structure and without a BOM.
On read it expects a UTF-16 encoded text, has some intelligence to guess the
byte order, and interprets a BOM if there is one. It doesn't seems to have any
provision for UTF-8 encoded UserComment text.

The proposed logic based on size and existence of 0-bytes in the comment will fail in many cases because some cameras write 0-byte (sometimes other characters) as fillers to this field, presumably so that it can be modified later without having to re-write the complete TIFF structure.

> I'd send a patch, [...]

Please do :) I'll be happy to discuss further and comment as the work progresses.

Andreas

#2 - 18 Dec 2009 18:21 - Andreas Huggel

Related info

- General remarks about the lack of Exiv2 characterset conversions
  http://dev.exiv2.org/boards/3/topics/show/220#message-221
  http://dev.exiv2.org/boards/3/topics/show/62#message-66
- The Metadata Working Group has published a technical specification with useful information on the topic.
- Bugs #562, #592: Exif.Photo.UserComment unicode comment doesn't work (Debian bug #486884). Exiv2 should decode to and from UTF-16 / UCS-2.
- Exiv2 vs Exiftool handling of Exif UNICODE user comments
- Bug #571: Convert character set when writing XMP sidecar. Adds UTF-8 charset detection logic and charset conversion to Exiv2
- Using UTF-8 (and other character sets) in IPTC datasets

#3 - 18 Dec 2009 19:09 - Leo Sutic

Hi Andreas,

having had a brief look at the code, it seems like CommentValue::read(const std::string& comment) is the place to put the modification in. Since the method reads an ASCII string (and I suspect the signature of the read function is fixed - no way to make it take a std::wstring), I suggest we simply let unicode escapes be passed into the function and decode them there to UCS-2 characters.

So you can have:

```
exifData["Exif.Photo.UserComment"] = "charset=\"Unicode\" \\u1f8f is a Greek Character"
```

Note that we escape the backslash, as the decoding of the \u1f8f sequence is done in CommentValue::read.

This would make it easy to specify any Unicode character (Including linebreaks) on command lines, and in command files without having to worry about encoding issues there.

How does the above sound to you? Did I understand things correctly that the std::string passed to CommentValue::read is a byte-string (char)? Or

can it be a wchar_t string?

#4 - 18 Dec 2009 22:53 - Andreas Huggel

> How does the above sound to you? Did I understand things correctly that the std::string passed to CommentValue::read is a byte-string (char)? Or can it be a wchar_t string?

Yes, only a simple char_t string is supposed to be used.

And yes, CommentValue, is the class that requires modifications.

Conceptually, I think we should follow Vladimir's suggestion and internally maintain the comment in UTF-8.

CommentValue::read(const std::string &buf) can make an attempt to translate whatever data it receives to UTF-8 (can it?) and store that in CommentValue. That function is used to set strings passed by the application/user, as in your example above, so we would expect the input to be Ascii or UTF-8 encoded text (or Windows Latin, I'm not familiar with that), i.e., nothing to do here in the first round.
The other read method, CommentValue::read (const byte *buf, long len, ByteOrder byteOrder) is used to read the comment value from the binary data stored in the Exif tag, i.e., we would expect the input to be in UCS-2 for charset=Unicode comments. It can make an attempt to verify that assumption and make a better decision if necessary, and in any case, convert the value to UTF-8, using the byte order passed in.

CommentValue::write and CommentValue::comment then output the UTF-8 value, that's straightforward. Finally, CommentValue::copy should output the correctly encoded data, depending on the charset indicated in the CommentValue and the byte order. I.e., for charset=Unicode, it should encode the string to UCS-2. And for everything to work as expected, size() and count() both will need to return the number of bytes in the output of copy().

With this, I suspect it is no longer practical to derive CommentValue from StringValueBase (even though it will still use an std::string value_), it may be better to derive it from Value directly now.

#5 - 19 Dec 2009 03:21 - Andreas Huggel

- Target version deleted (0.18.2)

#6 - 19 Dec 2009 03:35 - Leo Sutic

Yes, Vladimir's suggestion makes sense.

I still think we should derive it from StringValueBase. The UserComment is basically a string value with an encoding for serialization attached.

Since read() is used both for user input as well as application "input", I'll have to think it over. We need one way to accept already UTF8 encoded strings, and one for user input with Unicode escape sequences.

I am thinking about letting read() only accept UTF8 strings, and put the escape sequence decoding ahead of that. Maybe in the command line parser.

#7 - 19 Dec 2009 03:51 - Andreas Huggel

> I still think we should derive it from StringValueBase. The UserComment is basically a string value with an encoding for serialization attached.

No problem if it's still possible.

> I am thinking about letting read() only accept UTF8 strings, and put the escape sequence decoding ahead of that. Maybe in the command line parser.

I like that better too. It's not something the library needs to provide.

#8 - 19 Dec 2009 04:38 - Leo Sutic

> > I am thinking about letting read() only accept UTF8 strings, and put the escape sequence decoding ahead of that. Maybe in the command line parser.
>
> I like that better too. It's not something the library needs to provide.

It is also dangerous, since if the application reads the UserComment, and then sets it to the same value that was read, there is a second decoding of unicode escapes.

I think this decoding of unicode escapes can be useful for all values on the command line. How about inserting it after line 1084 of exiv.cpp, at the end

of parseLine? That way, we ensure that when the command line is parsed, **all** values are UTF-8. So the spec is: **The command line and all command files are to be in Ascii with Unicode escapes ("\uXXXX"). They are parsed into UTF8 for internal use.** It means that the CLI interface is firmed up a bit, but on the upside the results will be more predictable.

#9 - 19 Dec 2009 22:43 - Andreas Huggel

> I think this decoding of unicode escapes can be useful for all values on the command line. How about inserting it after line 1084 of exiv.cpp, at the end of parseLine?

Yes, that's where it should go. Other things can also be done there once we have an escape character (like newlines and escaped quotes). But that's all secondary right now.

> So the spec is: *The command line and all command files are to be in Ascii with Unicode escapes ("uXXXX").

What if my input is already UTF-8? Many people have set their terminals to use UTF-8 character encoding. They will ecpect things to "just work" without escaping anything.

Anyway, let's concentrate on the library changes, i.e., CommentValue first. For that all we need to do is specify that the input of CommentValue::read(const std::string &buf) is in UTF-8.

#10 - 21 Dec 2009 08:27 - Leo Sutic

Andreas Huggel wrote:

> > I think this decoding of unicode escapes can be useful for all values on the command line. How about inserting it after line 1084 of exiv.cpp, at the end of parseLine?
>
> Yes, that's where it should go. Other things can also be done there once we have an escape character (like newlines and escaped quotes). But that's all secondary right now.
>
> > So the spec is: *The command line and all command files are to be in Ascii with Unicode escapes ("uXXXX").
>
> What if my input is already UTF-8? Many people have set their terminals to use UTF-8 character encoding. They will ecpect things to "just work" without escaping anything.

If the input is UTF-8 and it contains Unicode escapes, those escapes will be interpreted as such. So if someone wants to insert the literal text "\u1234", then they **must** escape the backslash ("\\u1234"). I don't think there is any good way to get around that. Maybe strings with Unicode escapes should be a different data type, but I think this is an interface change that is worth it. Having a uniform way to input strings is needed and should be the default.

How about this rule: Interpret Unicode escapes, but just copy everything else? That way, Ascii with escapes will give the expected result, and UTF-8 with escapes will, too, as the backslash is a single-byte UTF-8 character and will not be mistaken for a component of a UTF-8 multibyte character (all components have the highest bit set, and '\' == 0x5c).

I'll have a quick go at this later this evening, but I leave for the holidays and will be back on the January 6, 2010.

I think we have the whole thing pretty much figured out, though.

#11 - 21 Dec 2009 09:01 - Leo Sutic

I am having trouble compiling.

I got Cmake to Configure everything (I think, this is my first time using CMake):

```
>C:/PROGRA~2/MinGW/bin/mingw32-make.exe
 ================> ICONV_LIBRARIES :
 -- None:
 -- Debug:             -g
-- Release:           -O3 -DNDEBUG
-- RelWithDebInfo:    -O2 -g
-- MinSizeRel:        -Os -DNDEBUG
-- ------------------------------------------------------------
 -- Building PNG support:            NO
 -- Building shared library:         NO
 -- XMP metadata support:            NO
 -- Building static libxmp:          NO
 -- Native language support:         NO
 -- Conversion of Windows XP tags:   YES
```

```
-- Nikon lens database:             NO
-- commercial build:                NO
-- ------------------------------------------------------------
----------------> ICONV_LIBRARIES :
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/leo/Documents/Development/Exiv/final
[  1%] Building CXX object src/CMakeFiles/exiv2.dir/convert.cpp.obj
C:\Users\leo\Documents\Development\Exiv\exiv2\src\convert.cpp:53:20: iconv.h: No such file or directory
(...)
mingw32-make.exe[2]: * [src/CMakeFiles/exiv2.dir/convert.cpp.obj] Error 1
mingw32-make.exe[1]:  [src/CMakeFiles/exiv2.dir/all] Error 2
mingw32-make.exe: ** [all] Error 2
```

So basically, the compiler can't find iconv.h. I have set ICONV_INCLUDE_DIR in CMake.

If I look in src\CMakeFiles\exiv2.dir\flags.make, I see:

```
CXX_FLAGS = -IC:\Users\leo\Documents\Development\Exiv\final -IC:\Users\leo\Documents\Development\Exiv\exiv2\xm
psdk\include -Wall -Wcast-align -Wpointer-arith -Wformat-security  -Wmissing-format-attribute -Woverloaded-vir
tual -W
```

...and it seems to me that the ICONV_INCLUDE_DIR i have specified should be in there somewhere.

#12 - 21 Dec 2009 16:33 - Andreas Huggel

> How about this rule: Interpret Unicode escapes, but just copy everything else? That way, Ascii with escapes will give the expected result, and UTF-8 with escapes will, too, as the backslash is a single-byte UTF-8 character and will not be mistaken for a component of a UTF-8 multibyte character (all components have the highest bit set, and '\' == 0x5c).

ok

#13 - 21 Dec 2009 16:43 - Andreas Huggel

> I am having trouble compiling.
>
> I got Cmake to Configure everything (I think, this is my first time using CMake):
> [...]

Oops, that was not intended to be a trap. CMake is still a highly experimental feature and not well tested (I will exclude it from the upcoming release). While it's interesting to see it in action, it requires work. I recommend you don't bother and use the good old configuration files. Essentially "make config; ./configure; make". See the README for details and how to configure it for iconv. I have that working here on MinGW/MSYS (although not used recently).

Have a good holiday!

#14 - 07 Jan 2010 07:57 - Leo Sutic

Ok, I'm back.

I have written the escape-sequence decoder now. It works by first converting everything to UCS-2, and then using iconv to convert that to UTF-8. For laziness I used the convertStringCharset function from convert.cpp. It is very useful. Could we move it to Util.cpp/hpp?

#15 - 07 Jan 2010 09:00 - Andreas Huggel

hi!

With r1998 the function is now part of the API, declared in convert.hpp for want of a better place for now (utils.cpp is not part of the library, it's only used by the exiv2 utility.)

#16 - 07 Jan 2010 09:52 - Leo Sutic

- File unicode_support.patch added

Here's the first attempt. Can you tell me how it works for you?

I had to modify convert.cpp to count the number of output bytes when converting to UCS-2, due to zero-bytes in the output.

#17 - 11 Jan 2010 02:15 - Andreas Huggel

Thanks! I finally had a look at it. I have only reviewed the code, not tested it. Some comments:

- Please use StringValueBase::value_ instead of a second string CommentValue::_comment (it is public), so we don't have 2 strings for essentially the same thing.
- In CommentValue::read(const byte* buf, long len, ByteOrder byteOrder): a conversion is only required if charsetId_ unicode (ignoring whatever conversion would be needed for "jis")
- CommentValue::encode(ByteOrder byteOrder): Similarly, only needed if charsetId_ unicode. The first 8 bytes are the charset identifier according to the Exif specs, that part is never converted to UCS-2.
- Style: Please change "this->*charsetId" to "charsetId*" to be consistent with the existing code.
- parseEscape: The conversion of the input to UCS-2 implies that the input is in ASCII. But as discussed, it may also be UTF-8 encoded. Instead, I prefer to require (and assume) that the input is in UTF-8 and simply insert the escaped characters into the string, according to your rule above

#18 - 11 Jan 2010 03:30 - Leo Sutic

Some comments on your comments:

1. The difference between value_ and *comment (which should be comment*) is that value_ contains the charset specifier (charset="...") given by the user. Should this be dropped and only the actual comment value be stored?
2. Ok
3. Ok
4. Ok
5. Ok, that one was embarrassing... will fix.

#19 - 11 Jan 2010 04:49 - Andreas Huggel

> 1. The difference between value_ and *comment (which should be comment*) is that value_ contains the charset specifier (charset="...") given by the user. Should this be dropped and only the actual comment value be stored?

Since you introduced CommentValue::charsetId_ I was under the impression this now replaces that charset specifier in the comment itself. Either way is fine with me, but we should avoid maintaining the information redundantly, esp since the members of these value classes are public.

#20 - 11 Jan 2010 07:54 - Leo Sutic

I thought I'd fix all the character conversions while I was at it - in particular, convert JIS to UTF8 and UTF8 to JIS on read/copy. Do you know which character encoding to use for JIS? The Exif 2.2 spec mentions JIS X 208-1990, but that is only a character set and not a specific encoding of characters into bytes. I don't think it is a problem if we just leave it untranslated, but it would be nice to get the three major character encodings (Ascii, Unicode and JIS) sorted in one big go.

#21 - 11 Jan 2010 14:06 - Leo Sutic

I am about to write testcases for the comment handling. I tried running the existing tests by cd'ing into test/ and running "make". That produced a lot of errors. Is this a known issue, or have I just broken exiv2?

#22 - 11 Jan 2010 15:09 - Leo Sutic

- File exiv2-exifcomment-unicode.patch added

- File exiv2-bug662.jpg added

Second attempt.

#23 - 11 Jan 2010 22:44 - Andreas Huggel

Leo Sutic wrote:

> I am about to write testcases for the comment handling.

Good. That's commendable.

> I tried running the existing tests by cd'ing into test/ and running "make". That produced a lot of errors. Is this a known issue, or have I just broken exiv2?

They all pass here with r1999. You need to make; make install; make samples before cd test; make
I hope a few tests will still fail after this though, even if your stuff now works perfectly :) Would mean that they are useful, after all you changed the way the CommentValue interface works.

- Instead of running all the tests in one go, you can run each script individually
- Once you have validated that the reported differences are as expected based on your changes, simply copy the output from the test script in the

tmp/ directory to the data/ directory (i.e., update the expected output)
- Consider adding your test cases to bugfixes-test.sh

#24 - 11 Jan 2010 23:02 - Andreas Huggel

- Consider adding your test cases to bugfixes-test.sh

Oh you already wrote your own test script. That's fine too, of course.

#25 - 12 Jan 2010 08:27 - Leo Sutic

I'm stuck at conversions.sh, testcase 6. I have the file m.xmp, containing:

```
<rdf:li xml:lang="x-default">This is a JIS encoded Exif user comment. Or was it?</rdf:li>
```

But when I do exiv2 -PEkyct tmp/m.xmp I get:

```
Exif.Photo.UserComment                          Undefined 208  (Binary value suppressed)
```

So the comment value has suddenly balooned to 208 bytes (from 59). Doing a hexdump of it with exiv2 -PEkych tmp/m.xmp:

```
Exif.Photo.UserComment                          Undefined 208
  0000  55 4e 49 43 4f 44 45 00 00 54 00 68 00 69 00 73  UNICODE..T.h.i.s
  0010  00 20 00 69 00 73 00 20 00 61 00 20 00 4a 00 49  . .i.s. .a. .J.I
  0020  00 53 00 20 00 65 00 6e 00 63 00 6f 00 64 00 65  .S. .e.n.c.o.d.e
  0030  00 64 00 20 00 45 00 78 00 69 00 66 00 20 00 75  .d. .E.x.i.f. .u
  0040  00 73 00 65 00 72 00 20 00 63 00 6f 00 6d 00 6d  .s.e.r. .c.o.m.m
  0050  00 65 00 6e 00 74 00 2e 00 20 00 4f 00 72 00 20  .e.n.t... .O.r.
  0060  00 77 00 61 00 73 00 20 00 69 00 74 00 3f 00 69  .w.a.s. .i.t.?.i
  0070  00 73 00 20 00 69 00 73 00 20 00 61 00 20 00 4a  .s. .i.s. .a. .J
  0080  00 49 00 53 00 20 00 65 00 6e 00 63 00 6f 00 64  .I.S. .e.n.c.o.d
  0090  00 65 00 64 00 20 00 45 00 78 00 69 00 66 00 20  .e.d. .E.x.i.f.
  00a0  00 75 00 73 00 65 00 72 00 20 00 63 00 6f 00 6d  .u.s.e.r. .c.o.m
  00b0  00 6d 00 65 00 6e 00 74 00 2e 00 20 00 4f 00 72  .m.e.n.t... .O.r
  00c0  00 20 00 77 00 61 00 73 00 20 00 69 00 00 00 00  . .w.a.s. .i....
```

That is, it looks like some kind of buffer overflow. I've tried to trace the path of the comment data through the code, but am completely stuck.

#26 - 12 Jan 2010 09:19 - Andreas Huggel

If it's an overflow, valgrind usually says useful things. The conversion from XMP to Exif is in convert.cpp, Converter::cnvXmpComment, if I'm not mistaken.

In the meantime I've checked-in your 2nd patch and subsequently modified it a bit. Please have a look at the related revisions and svn update.

Besides the issues with the conversion tests there is one left in bugfixes-test: If we write to an image with an existing comment shorter than 8 bytes (i.e., an invalid user comment), this "fixes" the comment in passing. I don't quite like that, but we can leave it for the moment. It's late here now, I'm signing off.

#27 - 12 Jan 2010 09:22 - Leo Sutic

I get everything to work expect the bugfixes-test.sh script.

It produces a 62428 byte output file. The correct file (data/bugfixes-test.out) is 7477 bytes.

Help.

#28 - 12 Jan 2010 09:31 - Andreas Huggel

Strange, why is your data/bugfixes-test.out so small? Mine is and always was much larger:

```
andreas@mowgli:~/src/exiv2/trunk/test/data$ svn up -r1999
U    bugfixes-test.out
Updated to revision 1999.
andreas@mowgli:~/src/exiv2/trunk/test/data$ ls -la bugfixes-test.out
-rw-r--r-- 1 andreas andreas 62428 13-Jan-10 bugfixes-test.out
andreas@mowgli:~/src/exiv2/trunk/test/data$ svn up
U    bugfixes-test.out
Updated to revision 2002.
andreas@mowgli:~/src/exiv2/trunk/test/data$ ls -la bugfixes-test.out
-rw-r--r-- 1 andreas andreas 64491 13-Jan-10 bugfixes-test.out
```

#29 - 12 Jan 2010 09:34 - Andreas Huggel

And this is the output that I get, related to the issue mentioned above:

```
andreas@mowgli:~/src/exiv2/trunk/test$ ./bugfixes-test.sh
Files ./tmp/bugfixes-test.out-stripped and ./data/bugfixes-test.out differ
55c55
< Exif.Photo.UserComment                        Undefined   8
---
> Exif.Photo.UserComment                        Undefined   1
```

#30 - 12 Jan 2010 09:40 - Leo Sutic

- File convert_bug.patch added

Andreas Huggel wrote:

> If it's an overflow, valgrind usually says useful things. The conversion from XMP to Exif is in convert.cpp, Converter::cnvXmpComment, if I'm not mistaken.
>
> In the meantime I've checked-in your 2nd patch

There are some serious bugs in that one (in convertStringCharset in particular - the line where I do outstr.append I use outbytesProduced, which is the cumulative number of output bytes produced, but I should only use the number of bytes produced in this particular call to iconv). Apply this patch to fix.

#31 - 12 Jan 2010 09:41 - Leo Sutic

Andreas Huggel wrote:

> And this is the output that I get, related to the issue mentioned above:
>
> [...]

Yes, since the size of the comment is now 8 bytes (character encoding) plus nothing more, you get 8 bytes for an empty comment.

#32 - 12 Jan 2010 10:09 - Leo Sutic

- File exiv2-exifcomment-unicode.patch added

Ok, third (and hopefully final) patch.

This includes the convertStringCharset patch, and fixes printing of comment values (changes to tags.cpp). It passes all except exiv2-test.sh here. I really don't know what is happening. I get crazy errors like:

```
Files ./tmp/exiv2-test.out and ./data/exiv2-test.out differ
217d216
< exiv2-empty.jpg: No Exif data found in the file
219a219
> exiv2-empty.jpg: No Exif data found in the file
```

Can you apply the patch and see if the tests work for you?

#33 - 12 Jan 2010 22:23 - Andreas Huggel

Leo Sutic wrote:

> Ok, third (and hopefully final) patch.

Looks good, thanks :)

> This includes the convertStringCharset patch, and fixes printing of comment values (changes to tags.cpp). It passes all except exiv2-test.sh here. I really don't know what is happening. I get crazy errors like:
>
> [...]

I get these too on some systems, but ignore them...

> Can you apply the patch and see if the tests work for you?

Will check it in tonight. As for the exifcomment-encoding-test.sh script, I moved the new testcases to bugfixes-test.sh last night, so this is not required anymore.

#34 - 13 Jan 2010 19:27 - Andreas Huggel

- Status changed from New to Resolved

- Target version set to 0.20

- % Done changed from 0 to 100

For Exif UNICODE comments, Exiv2 now expects the input encoded in UTF-8 and serializes it to UCS-2. In addition, special characters can be escaped in the input. The output is also in UTF-8.

Applications which previously worked around this problem by setting UNICODE comments in UCS-2 will need to be changed accordingly.

#35 - 13 Jan 2010 19:41 - Andreas Huggel

Leo,

If an image has a UNICODE comment encoded in UTF-8 (e.g., set using a previous version of Exiv2), would it be possible to detect that and not convert it on read then?

There is actually existing code to determine if text is UTF-8 encoded in IptcData::detectCharset(), iptc.cpp. I'm wondering if that algorithm could be generalized to also detect UCS-2 incl. the byte-order used. Then we could also deal with comments that have an UCS-2 comment but encoded using a different byte-order than the rest of the Exif data (which is another problem that an Exiv2 application could have previously introduced).

In any case, thanks again for your contribution so far! The escaping logic also fixes #572.

Andreas

#36 - 14 Jan 2010 01:38 - Leo Sutic

Andreas Huggel wrote:

> If an image has a UNICODE comment encoded in UTF-8 (e.g., set using a previous version of Exiv2), would it be possible to detect that and not convert it on read then?

I think we went over this at the start of this bug... The conclusion seems to be: Yes, but you are bound to mis-detect something. There are just so many UTF-8 byte sequences that actually are valid UCS-2 sequences that you can't just depend on detecting when the bytes cannot possibly be UCS-2. Exiftool, for example, tries to do some smart decoding, but even it fails on old exiv2 comments...

The only options I can think of is to add a method to the CommentValue class:
//! Decodes the raw comment data (that we squirrel away on read) as the given charset
std::string CommentValue::commentDecodedAs(std::string iconvCharset)
//! Returns the raw comment data (that we squirrel away on read)
CommentValue::data()

The read(std::string) method would set the "raw data" to the output of CommentValue::encode().

We can then have a utility that fixes the comments by reading them as UTF8 and writing UCS-2. It would be a one-time thing to run them on all files. The utility can be set to not convert comments that contain zero bytes, or only comments that contain UTF8 multibyte sequences. I think that would be preferable, as we would then know that the input **most likely** is UTF-8 (otherwise the user wouldn't run the utility).

> There is actually existing code to determine if text is UTF-8 encoded in IptcData::detectCharset(), iptc.cpp. I'm wondering if that algorithm could be generalized to also detect UCS-2 incl. the byte-order used.

Only heuristically. I think we need some kind of input from the user saying that "this comment **needs** autodetection" - otherwise you'll end up not converting when you should for some locale.

Or maybe run with autodetection as default for six months and then turn it off for the next version.

#37 - 14 Jan 2010 06:46 - Andreas Huggel

Ok. So the exiv2 utility could then work like this:

- By default, it autodetects the comment charset (via CommentValue::write)
- If that doesn't give the correct result, the user can indicate the charset to be used with a new command line option (in this case the proposed new function is used to decode the comment)
- The utility gets a new command to fix the comment encoding. Similarly, this command also uses autodetection by default or the charset specified with the same new command line option (and all it does is decode the comment using the specified charset and set it again)

And all of this is only applicable for UNICODE comments.

As for the implementation, I think this means that CommentValue internally stores (only) the raw comment (again). copy() becomes trivial, decoding is done in write (from UCS-2/the autodetected charset) and in the new function (from the specified charset) and encoding is done in read(const std::string&). The other proposed new function is not necessary since value_ is public.

What do you think?

#38 - 14 Jan 2010 09:02 - Andreas Huggel

I've refactored the implementation as sketched above, see r2005. It was slightly more hairy than expected because read() and write() don't know the byte order of course (copy() is not trivial because of that). I think the autodetection logic could now be added quite easily. The tests pass except for the special case that we mentioned earlier (that now again behaves as it used to).

#39 - 15 Jan 2010 03:33 - Andreas Huggel

Added detectCharset() and an optional parameter for the encoding to comment(). The rules to actually detect the character encoding are missing, if you have time to add them, please do, I will leave this as it is now for the next few days.

#40 - 21 Jan 2010 15:27 - Leo Sutic

Andreas Huggel wrote:

> Added detectCharset() and an optional parameter for the encoding to comment(). The rules to actually detect the character encoding are missing, if you have time to add them, please do, I will leave this as it is now for the next few days.

Hi, sorry for not getting back to you sooner.

I'll have a look at stuff tomorrow or over the weekend. The changes you've made sound fine.

#41 - 21 Jan 2010 18:30 - Andreas Huggel

Ok, great. There is some code to detect UTF-8 text in IptcData::detectCharset (iptc.cpp), maybe that should become a utility function so that it can be re-used. Anyway, I'll leave this issue alone over the weekend.

#42 - 24 Jan 2010 16:30 - Leo Sutic

Andreas Huggel wrote:

> Ok, great. There is some code to detect UTF-8 text in IptcData::detectCharset (iptc.cpp), maybe that should become a utility function so that it can be re-used. Anyway, I'll leave this issue alone over the weekend.

For the life of me, I can't find the new command line parameter (to override the autodetection).

The charset detection looks like a simple job, but where do you want it factored out to? Converter?

#43 - 24 Jan 2010 17:18 - Andreas Huggel

> For the life of me, I can't find the new command line parameter (to override the autodetection).

That's because there is none yet. Maybe what I said above is misleading. Only the library part was refactored, the new utility functionality has not been added yet.

> The charset detection looks like a simple job, but where do you want it factored out to? Converter?

I was thinking of another new function in convert.[ch]pp, like what we did with convertStringCharset.

#44 - 12 Feb 2010 07:38 - Andreas Huggel

> For the life of me, I can't find the new command line parameter (to override the autodetection).

Added with r2027.

#45 - 29 May 2010 10:42 - Andreas Huggel

- Status changed from Resolved to Closed

Files

unicode_support.patch	11.8 KB	07 Jan 2010	Leo Sutic
exiv2-exifcomment-unicode.patch	14.8 KB	11 Jan 2010	Leo Sutic
exiv2-bug662.jpg	1.34 KB	11 Jan 2010	Leo Sutic
convert_bug.patch	1.05 KB	12 Jan 2010	Leo Sutic
exiv2-exifcomment-unicode.patch	6.64 KB	12 Jan 2010	Leo Sutic