Exiv2 - Bug #1205
For Sharpness use Exif.CanonPr.Sharpness instead of Exif.CanonCs.Sharpness in easyaccess.cpp

12 Aug 2016 14:06 - Sridhar Boovaraghavan

Status:	Assigned	**Start date:**	12 Aug 2016
Priority:	Normal	**Due date:**	
Assignee:	Sridhar Boovaraghavan	**% Done:**	0%
Category:	metadata	**Estimated time:**	0.00 hour
Target version:	0.28		

Description

Exif.CanonPr.Sharpness returns values that are more in tune with the Canon user interface. It returns 3 in my case which is a value that I can immediately correlate to the sharpness that I have chosen (either through the built-in Picture Styles or a custom Picture Style).

Exif.CanonCs.Sharpness is undefined (i.e. has the value 7FFF) which is interpreted as undefined by exiftool.

Thus, I think Exif.CanonPr.Sharpness is a better tag to query from than Exif.CanonCs.Sharpness.

I would recommend replacing Exif.CanonCs.Sharpness in line 38 of source:easyaccess.cpp with Exif.CanonPr.Sharpness or at least adding Exif.CanonPr.Sharpness as a higher priority tag to query from.

Regards,
Sridhar

Associated revisions

Revision 4438 - 26 Aug 2016 05:36 - Sridhar Boovaraghavan

This is mainly a fix for #1206, but also interprets missing Canon Exif Tags in exiv2 with the help of Phil Harvey's exiftool (see http://www.sno.phy.queensu.ca/~phil/exiftool/TagNames/Canon.html).

Even with these changes (toward #1204 and #1205), exiv2 lags behind exiftool in some areas of interpretation of Canon tags. Ideally, a catch-up effort to bring the code in source: canonmn.cpp in line with lib/Image/ExifTool/Canon.pm. v10.25 of exiftool was used as reference for this change.

#1206 seeks to address the fact that when Canon does not have data for certain tags, they use specific default values in those fields. These default values need to be ignored and not displayed. This change brings this feature to exiv2, something that already exiftool does.

With regards to implementation, the struct TagInfo in source: tags.hpp is extended with four new fields.

The first field is a bool that if set to true (default false), denotes that this field has ignorable default values.

The second field is the default value that needs to be ignored. This can be of four types (String, Long, Float, Rational). These four types were chosen as they had conversion functions in the Value class.

The third field is the comparison type (default equal_to). There are six comparison types possible (equal_to, not_equal_to, less, less_equal, greater, greater_equal). This is the comparison applied to the value stored in the tag's field and the default value specified above. For e.g. if the value in the tag Exif.CanonCs.RecordMode is -1, then it needs to be ignored.

The fourth field is the data type (default Long). This could have been guessed from the type of the second field, but that would necessitate making this structure into a template calling for changes in multitude of files.

Usage: In source: canonmn.cpp, several exif tags now have ignorable default properties. I will list a few examples.

1. Exif.CanonCs.FocusMode: TagInfo(0x0007, "FocusMode", N_("Focus Mode"), N_("Focus mode setting"), canonCsId, makerTags, signedShort, 1, EXV_PRINT_TAG(canonCsFocusMode)),

There are no changes - i.e. this is an example of how the TagInfo structure was being populated.

2. Exif.CanonCs.RecordMode: TagInfo(0x0009, "RecordMode", N_("Record Mode"), N_("Record mode setting"), canonCsId, makerTags, signedShort, 1, EXV_PRINT_TAG(canonCsRecordMode), true, s_1_),

Take a look at the two new arguments. The first one (true) specifies that there is a default value that can be ignored. The second one s_1_ specifies the value (-1, in this case) to be ignored.

```
const UShortValue CanonMakerNote::s_1_(65535, unsignedShort); // Till bug is resolved
```

Note s_1_ is temporarily having the value 65535 till #1203 that causes signedShorts to be interpreted as unsignedShorts is resolved.

3. Exif.CanonSi.TargetAperture: TagInfo(0x0004, "TargetAperture", N_("Target Aperture"), N_("Target Aperture"), canonSiId, makerTags, unsignedShort, 1, printSi0x0015, true, us0_, TagInfo::less_equal),

Note the third argument TagInfo::less_equal. This combined with the second argument us0_ (the number 0) signifies that any values in this tag that are less than or equal (<=) to 0 should be ignored.

4. TagInfo(0x0028, "ImageUniqueID", N_("Image Unique ID"), N_("Image Unique ID"), canonId, makerTags, asciiString, -1, printValue, true, s0x16_, TagInfo::equal_to, TagInfo::String),

The previous examples have all been of Long type. This shows a case where the default value is a string.

```
const AsciiValue CanonMakerNote::s0x16_("0000000000000000");
```

Once these tag values have been defined, the actual mechanics of ignoring these default values happens in Image::exifData().

Before the exifData is returned, we loop through the data, ask the data whether it needs to be ignored (which in turn checks its underlying tagInfo and compares it with the default value, if specified) and if so, deletes that element.

A compile-time switch called EXV_DONT_IGNORE_UNDEFINED which when set
to a non-zero value will cause the behavior to revert back to the
original where all values are reported irregardless of the fact that
they need to be ignored.

History

#1 - 12 Aug 2016 16:20 - Robin Mills

- Category set to metadata

- Status changed from New to Assigned

- Assignee set to Sridhar Boovaraghavan

- Target version set to 0.28

Sridhar

Can you attach a test file for this issue? Does it report both Exif.CanonPr.Sharpness and Exif.CanonCs.Sharpness. Looking in the code, there are two tables dealing with Sharpness. //! Contrast, Saturation Sharpness, tags 0x000d, 0x000e, 0x000f

```
extern const TagDetails canonCsLnh[] = {
    { 0xffff, N_("Low")    },
    { 0x0000, N_("Normal") },
    { 0x0001, N_("High")   }
};
//! Sharpness Frequency Values
extern const TagDetails canonSharpnessFrequency[] = {
    { 0, N_("n/a")      },
    { 1, N_("Lowest")   },
    { 2, N_("Low")      },
    { 3, N_("Standard") },
    { 4, N_("High")     },
    { 5, N_("Highest")  }
};
```

Can you define what is wrong and behaviour you would like to see.

#2 - 13 Aug 2016 01:16 - Sridhar Boovaraghavan

Robin,

No, neither of these is what I want. I thought I was explicit, but let me try to clarify further.

In easyaccess.cpp, there are functions for tags that are probably most used by people. One of them is Exiv2::sharpness. It currently cycles through multiple tags to report the first value that it finds. The first one is Exif.Photo.Sharpness and the second one is Exif.CanonCs.Sharpness.

As you have seen from my other report, Exif.CanonCs.Sharpness is incorrectly handled. Since this tag reports 32767, this value is reported by Exiv2::sharpness. This is wrong. The real sharpness value is something between -3 and 3, or possibly between 0-7 in some other Canon cameras. After digging through the code to understand why I was getting a sharpness value of 3 from exiftool and Exif.CanonCs.Sharpness was 32767, I realized that the correct tag to query sharpness was Exif.CanonPr.Sharpness. This returns 3 correctly in exiv2.

What I am requesting in this bug is to change easyaccess.cpp, specifically line 38 and replace CanonCs with CanonPr.

Regards,

Sridhar

#3 - 13 Aug 2016 08:56 - Robin Mills

- Subject changed from For Sharpness use Exif.CanonPr.Sharpness instead of Exif.CanonCs.Sharpness to For Sharpness use Exif.CanonPr.Sharpness instead of Exif.CanonCs.Sharpness is easyaccess.cpp

Thanks for the clarification. I've never looked at easyaccess.cpp. I can't remember anybody ever mentioning it.

#4 - 13 Aug 2016 14:40 - Robin Mills

- Subject changed from For Sharpness use Exif.CanonPr.Sharpness instead of Exif.CanonCs.Sharpness is easyaccess.cpp to For Sharpness use Exif.CanonPr.Sharpness instead of Exif.CanonCs.Sharpness in easyaccess.cpp

#5 - 30 Aug 2016 16:12 - Robin Mills

Development of this feature has been deferred for v0.27 and to be developed in branches/develop. r4449