Exiv2 - Bug #1157

Out of memory error with user defined to_string template

27 Jan 2016 16:59 - Robin Mills

Status:	Closed	**Start date:**	27 Jan 2016
Priority:	Normal	**Due date:**	
Assignee:	Robin Mills	**% Done:**	100%
Category:	api	**Estimated time:**	15.00 hours
Target version:	0.26		

Description

If you iterate over the Exif meta-data doing to_string(exifData[i->key()])

where

```
   template <typename T>
    string to_string(T value){
        std::ostringstream os;
        os << value;
        return os.str();
    };
```

History

#1 - 27 Jan 2016 17:06 - Robin Mills

This was reported by private email from Mikayel Egibyan.

If you iterate over the Exif meta-data doing to_string(exifData[i->key()]) many-many times (more than 1000 times) on the same stream/kernel, there is a memory crash.

#2 - 27 Jan 2016 17:11 - Robin Mills

- Status changed from Assigned to Resolved

- % Done changed from 0 to 80

I've modified samples/exifprint.cpp to use your template. Here's the output:

```
.../trunk $ bin/exifprint ~/Stonehenge.jpg  | tail -10
09 count = 208990
WGS-84          count = 208991
2015:07:16 count = 208992
JPEG (old-style) count = 208993
300 count = 208994
300 count = 208995
inch count = 208996
4442 count = 208997
10837 count = 208998
Entered count = 208999
$
```

This code doesn't seem to be leaking! I increased the constant "1000" to "10000". It runs just over one minute on the Mac. Memory in use by exifprint is constant at 1.1mb

```
$ time (./bin/exifprint ~/Stonehenge.jpg | tail -2)
10837 count = 2089998
Entered count = 2089999

real    1m16.180s
user    1m20.564s
sys    0m10.574s
$
```

What are you doing with the strings returned by to_string? If you're pushing them into a std::vector or something, perhaps they don't go out of scope and will not be deallocated.

There's nothing special about Stonehenge.jpg - it's a file I use a lot for testing. You can get it from http://clanmills.com/Stonehenge.jpg

```
// ***************************************************************** -*- C++ -*-
// exifprint.cpp, $Rev: 4108 $
// Sample program to print the Exif metadata of an image

#include <exiv2/exiv2.hpp>

#include <iostream>
#include <iomanip>
#include <cassert>

    template <typename T>
    std::string to_string(T value){
        std::ostringstream os;
        os << value;
        return os.str();
    };

int main(int argc, char* const argv[])
try {

    if (argc != 2) {
        std::cout << "Usage: " << argv[0] << " file\n";
        return 1;
    }

    if ( strcmp(argv[1],"--version") == 0 ) {
        exv_grep_keys_t keys;
        Exiv2::dumpLibraryInfo(std::cout,keys);
        return 0;
    }

    Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open(argv[1]);
    assert(image.get() != 0);
    image->readMetadata();

    Exiv2::ExifData &exifData = image->exifData();
    if (exifData.empty()) {
        std::string error(argv[1]);
        error += ": No Exif data found in the file";
        throw Exiv2::Error(1, error);
    }
    Exiv2::ExifData::const_iterator end = exifData.end();
    int count = 0 ;
    for ( int i = 0 ; i < 1000 ; i++ )
    for (Exiv2::ExifData::const_iterator i = exifData.begin(); i != end; ++i) {
#if 0
        const char* tn = i->typeName();
        std::cout << std::setw(44) << std::setfill(' ') << std::left
                  << i->key() << " "
                  << "0x" << std::setw(4) << std::setfill('0') << std::right
                  << std::hex << i->tag() << " "
                  << std::setw(9) << std::setfill(' ') << std::left
                  << (tn ? tn : "Unknown") << " "
                  << std::dec << std::setw(3)
                  << std::setfill(' ') << std::right
                  << i->count() << "  "
                  << std::dec << i->value()
                  << "\n";
#endif
        std::cout << to_string(exifData[i->key()]) << " count = " << count++ << std::endl;
    }

    return 0;
}
//catch (std::exception& e) {
//catch (Exiv2::AnyError& e) {
catch (Exiv2::Error& e) {
    std::cout << "Caught Exiv2 exception '" << e.what() << "'\n";
    return -1;
}
```

#3 - 01 Feb 2016 11:34 - Robin Mills

- Status changed from Resolved to Closed

- % Done changed from 80 to 100

#4 - 02 Feb 2016 12:56 - Robin Mills

- Status changed from Closed to Assigned

- % Done changed from 100 to 50

- Estimated time set to 4.00 h

I'm reopening this case as Mikayel has sent files by private email.

Mikayel:
Please update this issue report and attach your files.

#5 - 02 Feb 2016 13:58 - Mikayel Egibyan

- File Capture.JPG added

- File Capture1.JPG added

- File 559572663_061680528a_o.jpg added

- File 1583369269_c03be2a6b2_o.jpg added

Thanks for the investigation Robin.

Please find attached the crashes that I have using your code, and even old exifprint.cpp.

Please note that I use: /MT ReleaseDLL x64 and /MTd DebugDLL x64 builds.

Thanks,
Mikayel

#6 - 02 Feb 2016 16:28 - Robin Mills

I think you have a mixed up build environment. Normally, I use static everywhere, or dynamic everywhere.

I believe it is possible to build static DLLs and a static application and use static C run-time. I don't think I've ever done that.

Is there a reason not to use the /MD+DLL build environments in msvc2005?

#7 - 02 Feb 2016 19:23 - Robin Mills

- % Done changed from 50 to 100

I think you are in a lot of trouble with this, Mikayel. I don't know how to fix this. I've modified x64\release to generate a DLL. It uses /MT. I've linked exifprint.exe with that and reproduced your crash.

I think the problem is that exifprint.exe is using a different heap from libexiv2.dll. So when you iterate, you are using something that has been allocated from the DLL's heap. When you free this resource in exifprint.cpp, you crash because it does not recognise the object as part of its heap.

The only fix I know is to use exiv2/msvc2005 as intended. If you want to link your app as /MT, you must link the static build of libexiv2. I'm surprised your application does not use /MD as functions such as ::free() and ::malloc() are in the C-rtlib and shared by every DLL in your process.

I've looked over all the settings offered by MSVC for heap control and can't see anything to address this issue. You may want to ask for help in an MSVC Forum, however I don't believe anybody else in Team Exiv2 can be of further help with this issue.

If you discover a fix - please share it on this thread for future reference.

#8 - 03 Feb 2016 07:15 - Robin Mills

Mikayel

As you know, I always want to help. I found this blog: http://www.davidlenihan.com/2008/01/choosing_the_correct_cc_runtim.html in which the author says:

- Use the debug version only internally, release for anything that could be used by customers
- Use the DLL version except in special situations
- Use consistent settings throughout your project (the runtime library setting can be done per source file)
- Don't worry if your runtime library settings match other libraries you use (unless the library comes in multiple runtime library versions)

That article references the following article in which Microsoft warn against using C runtime objects across DLL boundaries. Regrettably, that's what you are doing: https://msdn.microsoft.com/en-us/library/ms235460.aspx

It seems to me that using /MT (or /MTd) is only suitable for very small applications in which you want to bind the whole application into a single .exe.

When you build with /MT, a copy of the run-time library is bundled into the object you build. When you build with /MD, you are using a process-wide run-time library which is a DLL. That DLL changes for different versions of MSVC. For example, if you are using Visual Studio 2005, it is called MSVCR80.dll

I don't know anything about the application from which you are calling libexiv2. You should build that application with /MD and use the DLLs generated by msvc2005. If you cannot change that application, then I believe you will have to link the static libraries generated by msvc2005. However, I don't think you can say "generate DLLs ***AND*** use /MT" because you are violating the conditions described above by Microsoft.

#9 - 03 Feb 2016 07:51 - Robin Mills

- Status changed from Assigned to Closed

- Estimated time changed from 4.00 h to 5.00 h

#10 - 08 Feb 2016 20:21 - Robin Mills

- Estimated time changed from 5.00 h to 10.00 h

Mikayel

I've done more work on this and I am certain your trouble is being caused by attempting to freeing object across DLL boundaries. Here's the proof.

1 Build x64/Release as a DLL

a) libexiv2/Properties/General
Configuration Type: Dynamic Library (dll)

b) C++/Preprocessor: add EXV_HAVE_DLL;EXV_BUILDING_LIB

c) Build libexiv2 and check that the following have been build:

```
C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release>dir *.lib *.dll
 Volume in drive C has no label.
 Volume Serial Number is 0899-EF40

 Directory of C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release

02/08/2016  07:10 PM           455,576 libexiv2.lib  <--- beautiful new Release /MT linker stub library
02/08/2016  12:30 PM           394,172 libexpat.lib
02/08/2016  12:30 PM         7,032,036 xmpsdk.lib
02/08/2016  12:30 PM           169,788 zlib1.lib

 Directory of C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release

02/08/2016  07:10 PM         3,165,696 libexiv2.dll  <--- beautiful new Release /MT DLL
               5 File(s)     11,217,268 bytes
               0 Dir(s)  20,407,341,056 bytes free

C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release>
```

2 exifprint x64/Release

a) C/C++ Preprocessor: add ;EXV_HAVE_DLL

b) Build exifprint

c) Test exifprint:

```
C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release>exifprint --version | grep -e date -e time -e
 svn -e libexiv2
date=Feb  8 2016
time=19:08:34
svn=0
library=C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release\libexiv2.dll  <-- using our beautiful
 /MT dll
have_gmtime_r=0
have_timegm=0
xmlns=mediapro:http://ns.iview-multimedia.com/mediapro/1.0/

C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release>
```

Problem is when we give him a file, the iterator crashes

```
C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release>exifprint http://clanmills.com/Stonehenge.jpg
```

```
Exif.Image.Make                              0x010f Ascii      18  NIKON CORPORATION
Exif.Image.Model                             0x0110 Ascii      12  NIKON D5300
---- crash ----

C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release>
```

3 Modify exifprint.cpp

Add our own global cplusplus new/delete operators. Only don't allow delete() to do anything!

```
C:\cygwin64\home\rmills\gnu\exiv2\trunk\msvc2005\bin\x64\release>exifprint http://clanmills.com/Stonehenge.jpg

Exif.Image.Make                              0x010f Ascii      18  NIKON CORPORATION
Exif.Image.Model                             0x0110 Ascii      12  NIKON D5300
Exif.Image.Orientation                       0x0112 Short       1  1
Exif.Image.XResolution                       0x011a Rational    1  300/1
Exif.Image.YResolution                       0x011b Rational    1  300/1
Exif.Image.ResolutionUnit                    0x0128 Short       1  2
Exif.Image.Software                          0x0131 Ascii      10  Ver.1.00
.... runs smoothly to end...
```

Here's the code change:

```
// ***************************************************************** -*- C++ -*-
// exifprint.cpp, $Rev: 4108 $
// Sample program to print the Exif metadata of an image

#include <exiv2/exiv2.hpp>

#include <iostream>
#include <iomanip>
#include <cassert>

#include <cstdio>
#include <cstdlib>
#include <exception>
#include <new>

// replacement of a minimal set of functions:
void* operator new(std::size_t sz) throw(std::bad_alloc) {
    // std::printf("global op new called, size = %zu\n",sz);
    return std::malloc(sz);
}

void operator delete(void* ptr) throw()
{
    try {
        // std::puts("global op delete called");
        // std::free(ptr);
    } catch ( std::exception&  ) {}
}

int main(int argc, char* const argv[])
try {
...
}
```

4 Commentary

This is **NOT** a work-around. This demonstrates the exact nature of the problem.

You will have to take this puzzle up with a software architect at your company. If your company insists on using /MT with libexiv2.dll, you will have to redefine new() and delete() appropriately. Your company's software architecture is creating this problem and your company will have to resolve this.

Please stop sending me private emails about this (unless you wish to thank me for the 10 hours unpaid consultancy you have been given).

#11 - 11 Feb 2016 01:17 - Mikayel Egibyan

Thanks much Robin for your help! I value it!

The solutions is to easy to discover quickly:

NEVER pass the result of Exiv2 library to 3rd party library, application, dll, etc... without making a copy.
Basically you need to ensure that nothing else but Exiv2 deletes the allocated memory. This is true for all types, especially strings.

Mikayel

#12 - 11 Feb 2016 07:03 - Robin Mills

Thanks, Mikayel

I haven't understood your fix. Can you modify exifprint.cpp to demonstrate your fix, please?

#13 - 24 Feb 2016 13:40 - Mikayel Egibyan

Pardon me, I didn't notice your message.

Here is the way that I copy the output of Exiv2 before I pass it further:

```
    size_t len = strlen(str) + 1;
    char *buf = new char[len];
    strcpy(buf, str);
    return but;
```

Mikayel

#14 - 24 Feb 2016 14:01 - Robin Mills

I'll update msvc2005/ReadMe.txt with this ingenious tip. So the recipe is "copy the string" (as you said).

Incidentally, I looked up the auto_ptr pattern in wikipedia. https://en.wikipedia.org/wiki/Auto_ptr.   It contains the following ominous warning:

> may not be used in STL containers that may perform element copies in their operations

The auto_ptr are clearly a tricky customer, and the debugger doesn't understand it either.

The auto_ptr is used extensively by exiv2. I believe auto_ptr is something to ensure that memory doesn't leak when exceptions are thrown. A very good example of a problem being created to solve a problem. Exception handling is a poor error handling strategy. It is an invisible goto that crosses function boundaries. I much prefer returning error codes. Few engineers agree with me, however I am entitled to my opinion. In this case, the exception is being thrown across DLL boundaries and the DLLs are using static libraries. So the Value auto_ptr was allocated by the DLL and deleted by the host process which is using a different heap.

#15 - 24 Feb 2016 19:42 - Robin Mills

Mikayel

I really don't believe your fix solves this matter. You may have found a way of making your code stable, however I don't believe you have resolved the underlying issue. Please read the following (mostly frightening and blood curling discussion): http://stackoverflow.com/questions/16736195/inline-class-constructor-to-avoid-vc-memory-crash

The article referenced above claims that some relief from this problem arrived with MSVC 2012. If you're using MSVC 2012 or later, perhaps you are being spared the worst cases of the situation. It is my opinion that /MT DLLs are only suitable for small applications and should be avoided. I recommend that you link all your code and DLLs with /MD. I urge you to discuss this with your company's software architects. I am willing to join that discussion on Skype (or phone, Google+ or FaceTime) if you wish.

Looking at this situation from an Exiv2 viewpoint, I feel the only correct way to fix this issue is for the host application to pass pointers to ***malloc/free/realloc*** functions to be used by Exiv2. This is a non-trivial change to Exiv2. The default values of these functions would be the system's default implementation. So existing Exiv2 applications would require no changes. Other standard C library functions such as ***fopen/fclose/fwrite/fread...*** may benefit from similar treatment.

#16 - 26 Feb 2016 17:47 - Robin Mills

I have this ***mostly*** solved. I say ***mostly*** because the solution is incomplete, however I'll explain what else is necessary later.

The solution is for the host application to define the functions to be used by the library to allocate/free memory. So exifprint.cpp becomes:

```
// Sample program to print the Exif metadata of an image

#include <exiv2/exiv2.hpp>

#include <iostream>
#include <iomanip>
#include <cassert>

int main(int argc, char* const argv[])
try {

    if (argc != 2) {
        std::cout << "Usage: " << argv[0] << " file\n";
        return 1;
    }

    if ( strcmp(argv[1],"--version") == 0 ) {
        exv_grep_keys_t keys;
        Exiv2::dumpLibraryInfo(std::cout,keys);
        return 0;
    }

    Exiv2::Image::AutoPtr image = Exiv2::ImageFactory::open(argv[1]);
    assert(image.get() != 0);
    image->readMetadata();

    Exiv2::ExifData &exifData = image->exifData();
    if (exifData.empty()) {
        std::string error(argv[1]);
        error += ": No Exif data found in the file";
        throw Exiv2::Error(1, error);
    }

    Exiv2::setMemoryFunctions(::malloc,::free); // <----- tell Exiv2 to use our malloc and free
    for (Exiv2::ExifData::const_iterator i = exifData.begin(); i != exifData.end(); ++i) {
        const char* tn = i->typeName();
        std::cout << std::setw(44) << std::setfill(' ') << std::left
                  << i->key() << " "
                  << "0x" << std::setw(4) << std::setfill('0') << std::right
                  << std::hex << i->tag() << " "
                  << std::setw(9) << std::setfill(' ') << std::left
                  << (tn ? tn : "Unknown") << " "
                  << std::dec << std::setw(3)
                  << std::setfill(' ') << std::right
                  << i->count() << "  "
                  << std::dec << i->value()
                  << "\n";
    }
    Exiv2::setMemoryFunctions(NULL,NULL); // <-- restore Exiv2 default functions

    return 0;
}
//catch (std::exception& e) {
//catch (Exiv2::AnyError& e) {
catch (Exiv2::Error& e) {
    std::cout << "Caught Exiv2 exception '" << e.what() << "'\n";
    return -1;
}
```

Now we have to declare and define setMemoryFunctions() in the library.

1 Declare Exiv2::setMemoryFunctions()

Modify include/exiv2/exiv2.hpp as follows:

```
...
#include "xmpsidecar.hpp"

typedef void* (*malloc_fp)(std::size_t);
typedef void  (*free_fp  )(void*);

namespace Exiv2 {
    static malloc_fp _malloc = NULL;
    static free_fp   _free   = NULL;
    EXIV2API void setMemoryFunctions(malloc_fp malloc, free_fp free);
}

    // replacement of a minimal set of functions:
    void* operator new(std::size_t sz) {
        // std::printf("new called, size = %zu\n",sz);
        return Exiv2::_malloc ? Exiv2::_malloc(sz) : std::malloc(sz);
    }

    void operator delete(void* ptr)
    {
        try {
            // std::puts("delete called");
            if ( Exiv2::_free ) Exiv2::_free(ptr);
            else                std::free(ptr);
        } catch ( std::exception&  ) {}
    }
#endif                                  // #ifndef EXIV2_HPP_
```

2 Define setMemoryFunctions in basicio.cpp

Update src/basicio.cpp as follows:

```
# include <windows.h>
# include <io.h>
#endif

#include <exiv2/exiv2.hpp>

// *****************************************************************************
// class member definitions
namespace Exiv2 {

#ifdef EXV_BUILDING_LIB
    EXIV2API void setMemoryFunctions(malloc_fp malloc, free_fp free)
    {
        _malloc=malloc;
        _free  =free  ;
    }
#endif
    BasicIo::~BasicIo()
    {
    }
```

3 Build and Test

Good, isn't it?

It is possible to update MSVC2005 to compile this code when building /MT DLLs. I ***will not support this*** for the following reasons:

1) There are more changes required:

Exiv2 allocates memory with operator new, ::malloc(), ::calloc(), std::malloc(), ::realloc(). I will have to identify and modify all that code.

2) All the sample applications will require modification

3) If I move the calls to Exiv2::setMemoryFunctions() to the first and last lines of exifprint.cpp, the application asserts during exit. This is some synchronisation issue with the conflicting static libraries which I don't want to debug.

4) I am sure there will be more crashes to be investigated before the test suite performs perfectly.

5) I do not wish to encourage anybody to use /MT DLLs.

6) There are users with their own MSVC build environments. C++ code changes to deal with /MT DLLs will probably damage their build environments.

#17 - 26 Feb 2016 18:06 - Robin Mills

- Estimated time changed from 10.00 h to 15.00 h

Files

Capture.JPG	81.7 KB	02 Feb 2016	Mikayel Egibyan
Capture1.JPG	214 KB	02 Feb 2016	Mikayel Egibyan
559572663_061680528a_o.jpg	675 KB	02 Feb 2016	Mikayel Egibyan
1583369269_c03be2a6b2_o.jpg	230 KB	02 Feb 2016	Mikayel Egibyan