Exiv2 - Feature #1041

CMake toolchain for windows

15 Mar 2015 16:44 - Daniel Kaneider

Status:	Closed	**Start date:**	15 Mar 2015
Priority:	Normal	**Due date:**	
Assignee:	Robin Mills	**% Done:**	100%
Category:	build	**Estimated time:**	150.00 hours
Target version:	0.26		

Description

Hi,

I appended a .zip file which helps in compiling exiv2 on windows. Basically you start with a completely empty directory, put the two batch files inside, start the script and end up with:
- downloaded, compiled and installed zlib
- downloaded, compiled and installed expat
- downloaded, compiled and installed exiv2

To run the scripts you need Cygwin (outside the compiler), CMake, subversion binaries (for svn rev), and a VC compiler of course. Edit setenv.cmd and adapt the paths to your environment. The script should automatically detect the used compiler version. If you have multiple compilers installed, you might copy the three lines and put them at the end. To launch the build, simply execute build.cmd in your VC commandline shell.

To get this to better work, I did small adjustments to some CMake files (the other patch). Except for one little change, it just affects windows. I removed the lines which copy some zlib/expat binaries in some bogous way. It's not needed anymore with this builds scripts.

The aim of the scripts is similar to the 'cm.bat' file in exiv2-svn, just a little more powerful. I suggest to put the two batch files inside a new 'build' subdir.

Using my scripts, you end up with some binaries you can actually run, without having to copy .dlls around. This helps when doing automatic testing. However, your way of testing is still not very clear to me. As far as I understood, you need to call the right batch (.sh) file with the right arguments...

Extending my scripts to ssl and curl, shouldn't be that difficult. I'll eventually look into that.

Regards,
Daniel

Related issues:

Related to Exiv2 - Feature #1031: CMake build broken when using zlib/expat in...	**Closed**	**15 Feb 2015**
Related to Exiv2 - Feature #536: Switch to a unified build system based on cmake	**Closed**	
Related to Exiv2 - Patch #1119: libxmp missing from the install folder when b...	**Closed**	**23 Sep 2015**
Related to Exiv2 - Feature #1121: Visual Studio support for v0.27	**Closed**	**24 Sep 2015**
Related to Exiv2 - Bug #1123: libexiv2 should hide XMP-SDK symbols	**Closed**	**03 Oct 2015**

Associated revisions

Revision 3958 - 22 Sep 2015 13:37 - Robin Mills

#1041 Thanks you to Emmanuel for the patch. Discussed here: http://dev.exiv2.org/boards/3/topics/2232

Revision 4007 - 09 Nov 2015 20:18 - Robin Mills

#1041. Changes discussed with Daniel on Skype.

Revision 4021 - 16 Nov 2015 19:21 - Robin Mills

#1041. Work in progress. Derivative of Daniel's scripts for my personal use. See contrib/cmake/msvc/ReadMe.txt for details.

Revision 4022 - 16 Nov 2015 20:52 - Robin Mills

#1041. Correction to r4021. Reinstating Daniel's original code. Apologies to Daniel.

Revision 4024 - 17 Nov 2015 17:30 - Robin Mills

#1041: Store tested copies support libraries for use with exiv2 . These will be used by contrib/cmake/build.cmd immediately, and later by msvc2005/configure.py.

Revision 4026 - 17 Nov 2015 19:15 - Robin Mills

#1041. Lots of changes. Added a command-line parser to build.cmd. Build.cmd obtains source from the Exiv2 Team Repository.

Revision 4027 - 17 Nov 2015 20:29 - Robin Mills

#1041. Simplified building and linking curl etc. curl etc aren't building (something to do with CMake). Good day's progress.

Revision 4028 - 18 Nov 2015 09:53 - Robin Mills

#1041. Expanded the libraries. This avoids the need for cygwin (tar) on the build machine.

Revision 4029 - 18 Nov 2015 13:18 - Robin Mills

#1041. Simplification and documentation. webready associated libraries not working yet.

Revision 4030 - 18 Nov 2015 15:21 - Robin Mills

#1041. Tested/fixed build.cmd to work with different versions of Visual Studio

Revision 4031 - 18 Nov 2015 15:46 - Robin Mills

#1041: Tested/fixed the manual build instructions in ReadMe.txt

Revision 4032 - 18 Nov 2015 15:51 - Robin Mills

#1041. Corrections to r4031.

Revision 4033 - 18 Nov 2015 20:22 - Robin Mills

#1041 Test/fix several versions of Visual Studio. Added vcvars.bat script for setting environment.

Revision 4035 - 19 Nov 2015 09:39 - Robin Mills

#1041. Adding correct expat-2.1.0.tar.gz

Revision 4036 - 19 Nov 2015 10:06 - Robin Mills

#1041 Removed libraries. Only store compressed libraries and use 7z to decompress in scripts. team/libraries/libssh-0.5.5, team/libraries/openssl-1.0.1j, team/libraries/zlib-1.2.8, team/libraries/expat-2.1.0, team/libraries/curl-7.39.0

Revision 4037 - 19 Nov 2015 11:09 - Robin Mills

#1041. Download prebuilt openssl and updated documentation.

Revision 4038 - 19 Nov 2015 18:16 - Robin Mills

#1041 Getting better. Still not building curl and libssh.

Revision 4041 - 22 Nov 2015 09:11 - Robin Mills

#1041 Fixed bug in libssh-0.7.2.tar.gz ConfigureChecks.cmake

Revision 4042 - 22 Nov 2015 20:37 - Robin Mills

#1041. Use 'badger curl' which I believe has a solid CMake environment. https://github.com/bagder/curl/releases

Revision 4048 - 24 Nov 2015 11:03 - Robin Mills

#1041 rename build.cmd as cmakeBuild.cmd and setenv.cmd as cmakeDefaults.cmd
I'm giving up on CMake/Visual Studio. Works OK with exiv2/zlib/expat. Hopeless for webready.
CMake cannot build openssl with VS
CMake/VS/curl is poor (issues a warning)
CMake/VS/libssh is broken (bug and fix reported)

Revision 4051 - 26 Nov 2015 18:47 - Robin Mills

#1041. Adding option --test and --bash to cmakeBuild.cmd. Correction to vcvars.bat to respect 32 bit builds by setting PROCESSOR_ARGUMENT.

Revision 4052 - 26 Nov 2015 19:48 - Robin Mills

#1041. Added script cmakeRebuildAll.cmd. Documentation Update. Fix to vcvars 2005 32

Revision 4053 - 26 Nov 2015 21:51 - Robin Mills

#1041. Fixing typos and cosmetic stuff.

Revision 4055 - 27 Nov 2015 17:44 - Robin Mills

#1041. Minor costmetic fixes (see ReadMe.txt for details).

Revision 4056 - 27 Nov 2015 18:44 - Robin Mills

#1041. Fixed issues with VS2008 incorrectly setting EXV_HAVE_STDINT_H

Revision 4057 - 27 Nov 2015 19:40 - Robin Mills

#1041. Correction to r4056

Revision 4058 - 27 Nov 2015 20:15 - Robin Mills

#1041. Correction to r4056

Revision 4059 - 28 Nov 2015 11:24 - Robin Mills

#1041. More polishing to get every version of VS to build using CMake.

Revision 4061 - 29 Nov 2015 22:35 - Robin Mills

#1041. Finally fixed stdint.h on all platforms. CMake builds exiv2 with VS 2005, 2008, 2010, 2012, 2013 and 2013 on both laptop and buildserver.

Revision 4062 - 29 Nov 2015 22:58 - Robin Mills

#1041. Documentation Update.

Revision 4063 - 03 Dec 2015 20:14 - Robin Mills

#1041. Trying again to build curl with nmake.exe. It builds from the command line, but not from cmakeBuild.cmd.

Revision 4064 - 05 Dec 2015 15:59 - Robin Mills

#1041. curl is now building using nmake. exiv2+zlib+expat are OK. --webready still broken. libssh has broken, although OK yesterday.

Revision 4066 - 05 Dec 2015 20:20 - Robin Mills

#1041. Fixed --webready. Hip hip hip, hurrah for me. Documentation Update.

Revision 4067 - 07 Dec 2015 00:33 - Robin Mills

#1041. We don't need openssl! curl is the only person using openssl and can be built with winssl support instead.
Code polishing. Documentation update.

Revision 4068 - 07 Dec 2015 10:06 - Robin Mills

#1041. Fix libssh to build on VS2005 & VS2008

Revision 4069 - 07 Dec 2015 12:00 - Robin Mills

#1041. Fix libssh to find _snprintf and _vsnprintf using VS2015

Revision 4070 - 07 Dec 2015 15:30 - Robin Mills

#1041. Adding openssl 1.0.1p for VS2005 (created by contrib/cmake/msvc/cmakeOpenssl.bat)

Revision 4071 - 07 Dec 2015 16:03 - Robin Mills

#1041. openssl-1.0.1p.tar.gz isn't used by cmakeBuild.cmd. However, for completeness, it makes sense to provide this on svn.

Revision 4072 - 07 Dec 2015 16:25 - Robin Mills

#1041. Fixes for VS2005/8/15 and libssh. Documentation update.

Revision 4076 - 08 Dec 2015 11:06 - Robin Mills

#1041 Adding jenkins_daily.sh to build CMake/msvc

Revision 4077 - 08 Dec 2015 13:41 - Robin Mills

#1041. Updating jenkins_daily.sh. I'll leave this to run for a few days asking him to build everything and publish the artefacts.

Revision 4078 - 08 Dec 2015 16:09 - Daniel Kaneider

#1041. Adding better NMake support

Revision 4079 - 08 Dec 2015 20:42 - Robin Mills

#1041. Updating jenkins_daily.sh to build in <exiv2dir>/build

Revision 4080 - 08 Dec 2015 21:32 - Robin Mills

#1041 More changes to get jenkins_daily.sh to run from Jenkins on W10.

Revision 4081 - 09 Dec 2015 11:21 - Robin Mills

#1041 Still trying to persuade jenkins_daily.sh to build AND run the test suite on buildserver

Revision 4082 - 09 Dec 2015 11:53 - Robin Mills

#1041: Still trying to get test suite to run after build.

Revision 4083 - 09 Dec 2015 12:21 - Robin Mills

#1041 Still trying to get the tests to run. Gggrrr. Windows10+Cygwin don't like each other.

Revision 4084 - 09 Dec 2015 13:38 - Robin Mills

#1041 Get jenkins_daily.sh to run test suite. I'm making hard work from a simple task.

Revision 4085 - 09 Dec 2015 18:20 - Robin Mills

#1041. Yet another effort to get the test suite to run.

Revision 4086 - 09 Dec 2015 18:34 - Robin Mills

#1041. Another attempt.

Revision 4087 - 09 Dec 2015 19:22 - Robin Mills

#1041. Changes to ensure cmakeBuild.cmd selects win32/x64 correctly.

Revision 4088 - 10 Dec 2015 03:08 - Robin Mills

#1041 Still struggling with the test suite

Revision 4089 - 10 Dec 2015 04:00 - Robin Mills

#1041 We often build Win32 although we request x64. I suspect a rogue copy of vcvars.bat on the builder's path.

Revision 4090 - 10 Dec 2015 09:06 - Robin Mills

#1041 Enable user to pass arguments to jenkins_daily.sh. Better 'no build generated' handling.

Revision 4091 - 10 Dec 2015 09:12 - Robin Mills

#1041 Correction to r4090

Revision 4092 - 10 Dec 2015 09:30 - Robin Mills

#1041. Strengthen default opt handling in vcvars.bat

Revision 4093 - 10 Dec 2015 10:05 - Robin Mills

#1041. debugging vcvars.bat

Revision 4094 - 10 Dec 2015 10:26 - Robin Mills

#1041: Fixed vcvars.bat. jenkins bash environment has PROCESSOR_ARCHITECTURE=x86, although the Jenkins builder configure panel (and script console) says AMD64.

Revision 4095 - 10 Dec 2015 11:12 - Robin Mills

#1041 Polishing jenkins_daily.sh

Revision 4096 - 10 Dec 2015 19:27 - Robin Mills

#1041 get jenkins_daily.sh to work on Mac/Cygwin/MSVC/Linux and publish builds on userContent/builds { daily/weekly/monthly}

Revision 4097 - 10 Dec 2015 21:34 - Robin Mills

#1041 Fixes to jenkins_daily.sh for macosx and linux. Disabled cygwin on Jenkins to avoid the Windows10/cygwin/fork crash.

Revision 4098 - 11 Dec 2015 13:00 - Robin Mills

#1041. Build server is working fine from command-line on Linux/Mac/Cygwin/msvc (MinGW not tackled yet).
Cygwin/msvc have been crashing in Jenkins due to cygwin fork issue. I've totally reinstalled cygwin on the buildserver. If this doesn't fix it, I think a cron-job on the build-server can run jenkins_daily over ssh on windows. Almost there. Very pleased with progress.

Revision 4099 - 11 Dec 2015 18:31 - Robin Mills

#1041 Minor changes to jenkins_daily.sh for Kubuntu and Mac

Revision 4100 - 11 Dec 2015 18:47 - Robin Mills

#1041. Correction to typos in r4099

Revision 4101 - 11 Dec 2015 19:08 - Robin Mills

#1041. Correction to r4099 for Linux

Revision 4102 - 11 Dec 2015 19:26 - Robin Mills

#1041: Changes to set PLATFORM correctly.

Revision 4103 - 12 Dec 2015 09:17 - Robin Mills

#1041 Fixed msvc to run from ssh

Revision 4104 - 12 Dec 2015 09:27 - Robin Mills

#1041 Detech PLATFORM as msvc using JOB_NAME

Revision 4105 - 12 Dec 2015 10:21 - Robin Mills

#1041 Adding a comment to jenkins_daily.sh

Revision 4106 - 12 Dec 2015 11:32 - Robin Mills

#1041 set LD_LIBRARY_PATH to ensure the test suite uses the newly built library

Revision 4107 - 12 Dec 2015 11:58 - Robin Mills

#1041 date handling is locale independent

Revision 4108 - 12 Dec 2015 19:10 - Robin Mills

#1109 and #1041 Added documentation and log files to the published builds.

Revision 4109 - 13 Dec 2015 14:46 - Robin Mills

#1109 and #1041 Corrections to ReadMe.txt.

I still haven't persuaded cygwin to link correctly. Something horribly wrong. I've realised that the buildserver sshd is 32 bit and therefore builds a 32 bit library. I might attempt the configure sshd to run cygwin64, however it took a lot of effort to get everything to build over sshd, I'm reluctant to modify anything about sshd. I'll feel differently when I work on this tomorrow.

Revision 4110 - 13 Dec 2015 16:14 - Robin Mills

#1109 and #1041 More corrections to ReadMe.txt

I used regedit (run as administrator) to change the Windows services to run c:\cygwin64\bin\cygrunsrv.exe and sshd restarted effortlessly in 64 bits. samples/exifprint.cpp compiles/links/runs as documented. Very pleased.

Revision 4111 - 14 Dec 2015 11:48 - Robin Mills

#1109 and #1041 Documentation corrections. First cut of buildserver/test_daily.sh script

Revision 4112 - 14 Dec 2015 11:49 - Robin Mills

#1109 and #1041 Detab script

Revision 4113 - 14 Dec 2015 13:06 - Robin Mills

#1109 and #1041 Work in progress. Adding linux/cygwin/msvc support to test_daily.sh

Revision 4114 - 14 Dec 2015 14:51 - Robin Mills

#1109 and #1041 Always build 64 bit libraries.

Revision 4115 - 14 Dec 2015 16:56 - Robin Mills

#1109 and #1041 Fixes for msvc

Revision 4117 - 14 Dec 2015 21:58 - Robin Mills

#1109 and #1041. Use /usr/local/bin/cmake to build (because I can guarantee it on the buildserver). Documentation Update.

Revision 4118 - 14 Dec 2015 22:30 - Robin Mills

#1109 and #1041 Only use /usr/local/bin/curl because I've built and installed 7.45.0 there.

Revision 4119 - 15 Dec 2015 05:41 - Robin Mills

#1109 and #1041 Use xmllint -html to parse output from webserver.

Revision 4120 - 15 Dec 2015 05:50 - Robin Mills

#1109 and #1041 Process the last build when there's more than 1.

Revision 4121 - 15 Dec 2015 15:10 - Robin Mills

#1109 and #1041. Reinstate recursive cygwin bash

Revision 4122 - 15 Dec 2015 15:25 - Robin Mills

#1109 and #1041 Debugging cygwin/jenkins_build.sh I don't know why this has broken as it has been working perfectly for at least a year.

Revision 4123 - 15 Dec 2015 16:20 - Robin Mills

#1109 and #1041. Don't call cygwin recursively. I replaced c:\cygwin yesterday with 64bit cygwin to avoid the Windows10 fork/crash. Use CXXFLAGS=-m64 - just like unix.

Revision 4124 - 15 Dec 2015 21:19 - Robin Mills

#1109 and #1041 Move scripts to contrib/buildserver

Revision 4125 - 15 Dec 2015 22:22 - Robin Mills

#1109 and #1041 Adding cmake_daily.sh and testDailyAll.sh

Revision 4126 - 16 Dec 2015 10:52 - Robin Mills

#1109 and #1041 Documentation update

Revision 4127 - 16 Dec 2015 19:50 - Robin Mills

#1109 and #1041 Adding contrib/buildserver/spread which runs from test-cmake-daily to update links in builds/Categorized

Revision 4128 - 17 Dec 2015 18:46 - Robin Mills

#1109 and #1041 Simplification of daily cmake build/test/publish/categorize (Work in Progress)

Revision 4129 - 17 Dec 2015 19:48 - Robin Mills

#1109 and #1041 Rename categorize as categorize.sh Documentation update.

Revision 4130 - 19 Dec 2015 21:24 - Robin Mills

#1109 and #1041. Factored out common code into functions.so Documentation update.

Revision 4131 - 19 Dec 2015 21:49 - Robin Mills

#1109 and #1041. Correction to r4130

Revision 4132 - 19 Dec 2015 22:21 - Robin Mills

#1109 and #1041 Update to jenkins_build.sh surprise: msvc build did not execute. The "what kind of build is this?" code has been in service for

about 1 year.

Revision 4137 - 21 Dec 2015 19:21 - Robin Mills

#1109 and #1041 categorize.sh rewritten

Revision 4145 - 25 Dec 2015 13:43 - Robin Mills

#1109 and #1041 Changes for Parallels 11.0 running on buildserver (Parallels 11 is a Christmas to myself).

Revision 4146 - 26 Dec 2015 09:45 - Robin Mills

#1109 and #1041 Build with cmakeBuildAll.cmd

Revision 4157 - 29 Dec 2015 20:10 - Daniel Kaneider

#1041. cmakeBuild with static compilation

Revision 4261 - 28 Mar 2016 19:53 - Robin Mills

#1041 Fix to msvc/test_daily.sh

Revision 4262 - 28 Mar 2016 20:52 - Robin Mills

#1041 Correction to r4261

Revision 4263 - 28 Mar 2016 21:50 - Robin Mills

#1041 Debugging msvc/test_daily.sh

Revision 4264 - 28 Mar 2016 22:54 - Robin Mills

#1041 Debugging msvc/test_daily.sh

Revision 4265 - 29 Mar 2016 09:39 - Robin Mills

#1041 Debugging msvc/test_daily.sh

Revision 4266 - 29 Mar 2016 11:38 - Robin Mills

#1041 Debugging msvc/test_daily.sh

Revision 4267 - 29 Mar 2016 12:03 - Robin Mills

#1041 Debugging msvc/test_daily.sh

Revision 4268 - 29 Mar 2016 15:12 - Robin Mills

#1041 Debugging msvc/test_daily.sh

Revision 4272 - 08 Apr 2016 06:16 - Robin Mills

#1041 Debugging cmake_daily.sh

Revision 4273 - 08 Apr 2016 13:08 - Robin Mills

#1041 Debugging cmake_daily.sh

Revision 4274 - 08 Apr 2016 13:50 - Robin Mills

#1041 Debugging cmake_daily.sh

Revision 4275 - 08 Apr 2016 14:18 - Robin Mills

#1041 Debugging cmake_daily.sh

Revision 4276 - 08 Apr 2016 15:18 - Robin Mills

#1041 Fix to msvc/test_daily.sh

Revision 4277 - 08 Apr 2016 16:34 - Robin Mills

#1041 Fix to mesvc/test_daily.sh

Revision 4311 - 04 Jun 2016 05:56 - Robin Mills

#1187 and #1041 Fixing CMake/MSVC build breaker

Revision 4312 - 04 Jun 2016 17:08 - Robin Mills

#1187 and #1041. Fixing CMake/MSVC 2013/15 build breakers

Revision 4313 - 05 Jun 2016 06:15 - Robin Mills

#1187 and #1041. Fixing CMake/Linux build breakers concerning libpthread

Revision 4323 - 13 Jun 2016 13:18 - Robin Mills

#1187 and #1041. Fixing CMake to install include/exiv2/rwlock.hpp (detected by contrib/buildserver/test_daily.sh)

Revision 4324 - 13 Jun 2016 15:14 - Robin Mills

#1187 and #1041. Fixing src/Makefile to install include/exiv2/rwlock.hpp (detected by contrib/buildserver/test_daily.sh)

Revision 4335 - 18 Jun 2016 14:06 - Robin Mills

#1041 CMake/MinGW fix. I have never persuaded CMake/MinGW to work.

Revision 4338 - 23 Jun 2016 18:30 - Robin Mills

#1041 Reorganized and simplified config header files.

Revision 4474 - 08 Sep 2016 18:52 - Robin Mills

#1041 Adding support for MinGW daily 32-bit build using autotools. CMake/MinGW isn't working.

Revision 4475 - 08 Sep 2016 19:05 - Robin Mills

#1041 Correction to r4474

Revision 4476 - 08 Sep 2016 19:08 - Robin Mills

#1041 Correction to r4474

Revision 4486 - 12 Sep 2016 19:22 - Robin Mills

#1041 Work in progress. Updating MinGW/daily build to populate directory dist/mingw.

Revision 4487 - 12 Sep 2016 20:14 - Robin Mills

#1041 Work in progress. Corrections to r4486 to run test suite and restrict copy from /usr/local to dist/mingw to expat/zlib/exiv2 files

Revision 4488 - 12 Sep 2016 20:59 - Robin Mills

#1041 Work in progress. Corrections to r4486 to copy the sample from bin/*.exe and other fixes

Revision 4489 - 13 Sep 2016 07:16 - Robin Mills

#1041 Work in progress. More corrections to r4486. Updated dist/ReadMe.txt to deal with MinGW

History

#1 - 15 Mar 2015 17:16 - Robin Mills

- Status changed from New to Assigned

- Target version set to 0.25

Thank you for providing these files, Daniel. I'll look at them in the next few days and update this issue report.

Out test suite is written in bash and may be run from Cygwin or MinGW. An environment variable EXIV2_BINDIR is used to define the directory in which to find the built executables and DLLs. Our msvc2005/msvc2003 environments build directly (without copying) to <exiv2dir>/msvc2005/bin/$platform/$config. where platfom := Win32|x64 and config = Debug|DebugDLL|Release|ReleaseDLL. With MSVC builds, I normally use the script testMSVC.sh to set EXIV2_BINDIR and run the test suite.

Thank You for doing this. I appreciate the time and effort you have put into this.

Robin

#2 - 30 Mar 2015 20:29 - Robin Mills

Daniel

I have news for you. Gilles and Islam (a new GSoC 2015 student) are going to accept responsibility for CMake/MSVC. The last few weeks have been really hard going for me and I put CMake to the bottom of the stack to focus on other matters. Friends have arrived from California and we're going off for a break until April 10. However we have more engineers coming to work on Exiv2 and something will be done about this.

#3 - 26 Apr 2015 07:32 - Robin Mills

Islam isn't going to join us after all. He's going to do a Rails project. I have agreed to do the webp/webm project #1048

I do not intend to anything about CMake/MSVC It states clearly in the README-CMAKE:

- CMake scripts are "work in progress".
  Use them only if you're prepared to fix them.
  See TODO-CMAKE for known pending tasks.

I am not even willing to submit a patch. Somebody else must take ownership of CMake/MSVC because I am overloaded.

#4 - 26 Apr 2015 08:36 - Robin Mills

- Assignee deleted (Robin Mills)

#5 - 26 Apr 2015 08:38 - Robin Mills

- Tracker changed from Bug to Feature

- Status changed from Assigned to New

#6 - 26 Apr 2015 08:45 - Robin Mills

Under no circumstances will I do any further work on CMake/MSVC. I am overloaded. I appreciate that Daniel has provided some assistance and encouragement. However, I can do nothing more about this matter.

#7 - 02 Jun 2015 23:42 - Alan Pater

- Target version changed from 0.25 to 0.26

#8 - 22 Sep 2015 14:04 - Emmanuel d'Angelo

I can't commit to maintain this part (as I have no windows PC outside work). However, I could fix a few things, see this forum thread.
My next step for the windows build is to check why we add issues with cmake's FindZlib.

Currently, our build workflow (that successfully works) is the following:
- build zlib and expat separately
- pass the path to these outputs to exiv2 via the cmake options
- build successfully (both inside MSVC's IDE and by calling devenv).

#9 - 22 Sep 2015 14:50 - Robin Mills

Thank, Emmanuel. I've understood that you're not committing to dealing Exiv2/CMake/Visual Studio. However, it's easier to understand different activity by you and Daniel and I when we discuss things in one location.

#10 - 22 Sep 2015 16:01 - T Modes

I have problems with the the last 2 revisions: 3948 and 3958:

1. Rev 3948:
ZLIB_LIBRARIES contains already the extension when using CMake gui. It should not be added by CMake itself. So this changeset breaks my compilations on Windows.
Adding unconditional the extension will also break later when the zlib detection is revised. Please revert this revision.

2. Rev 3958:
This revision removes
#if defined WIN32 && !defined *CYGWIN* && !defined *MINGW*

1. define EXV_UNICODE_PATH
   #endif
   with the consequence that all programs which are using exiv2 as library have to add this define in its own code (when the want to use the UNICODE code) which is ugly and can result in mismatches between library and calling program. And so there is no way for the calling program to test if this feature is available or not.
   If you want to make it configurable then the EXV_UNICODE_PATH define should be remain in config.h, maybe using some #cmakedefine variant, but not in the build system where it is unavailable for external programs.

(Furthermore adding EXV_UNICODE_PATH only to the build environment, it is then missing in the supplied project files.)

#11 - 22 Sep 2015 16:24 - Robin Mills

- Status changed from New to Assigned

If I revert r3948, this will break the build for Emmanuel who claimed this was needed.

I'm surprised that cmake/GUI and cmake/command-line interpret commands in CMakeList.txt file differently. Are you sure this isn't a different in CMake versions? Is there a way to say "Add zlib...lib" with CMake is invoked from the command-line and "Add zlib" when it's invoked by the GUI.

I agree concerning r3958. We should put EXV_UNICODE_PATH into config/config.h.cmake as a cmakedefine. We shouldn't dynamically create compiler defines such as -DEXV_UNICODE_PATH

I've suffered greatly with CMake. If you wish to provide a patch, I will submit it. I'm unwilling to actively develop CMake/Exiv2.

#12 - 22 Sep 2015 16:25 - Robin Mills

- Assignee set to Robin Mills

#13 - 22 Sep 2015 16:50 - T Modes

I don't know if there are version differences in CMake. But it assume this behaviour is the same in all recent versions. (Otherwise there would be more complains after last release, which is using the old version)

As far as I tested, all FIND_PACKAGE routines return the full name with extension of the libraries and the ZLIB_LIBRARIES (as other other libraries) is by default a list item:
(from FindZLIB.cmake)
ZLIB_LIBRARIES - **List** of libraries when using zlib.
Just adding the extension will only concat 2 strings, but not add the extension to each list item. So this approach does not work in this way.
So I assume Emmanuel is only supply the filename without extension to CMake. He should supply the full filename with extension.

#14 - 22 Sep 2015 16:58 - Robin Mills

Groan. I'll look at this later. I really dislike CMake. It's full of black magic such as FIND_MAGIC. I don't understand cmake nor how it works. It's just trouble with Visual Studio. Lots of folks have told me that it's really great - however none of its supporters have stepped forward to help by taking this crap off my back.

#15 - 23 Sep 2015 07:43 - Robin Mills

- Assignee deleted (Robin Mills)

I will not work on CMake issues.

#16 - 23 Sep 2015 08:59 - Emmanuel d'Angelo

Well, we supply the exact name (with extension) of the target zlib. I checked the FindZLib script but couldn't detect anything that helps. I have still to check between cmake 3.3.1 and previous cmake versions, because we didn't have this issue with an older version of cmake (so this may be due to a discrepancy between 3.3 and older instead of a discrepancy between cmake and cmake-gui). Checking the generated sln file does indeed show that the generated linker command is **without** extension on all our test machines (win8, Visual Studio 2013, cmake 3.3.1). If I find a better solution I will post it here.

For the unicode flag we have the exact opposite problem. It doesn't get defined automatically from the build environment, and we used to manually patch the sources before building. Although that may sound a bit controversial, I do anyway prefer to explicitly set definitions and compile-time options. This makes easier to check what really happens and allows easier compatibility (consistency) with third party code that may not embed the same magic-fu for guessing environment settings (and we have scripting when command lines become too long to type once in a while).

#17 - 23 Sep 2015 09:45 - Robin Mills

- Assignee set to Robin Mills

Thanks, Emmanuel. I now have Visual Studio 2013 working on my laptop. I don't know why it won't install correctly on the build server.

And thank you T. I regret that you're frustrated by this. We are all frustrated by this. I'll try to get this resolved today. I'm more suspicious of the version of CMake being used than your thought that the GUI/command-line behave differently.

I didn't think too much before accepting Emmanuel's proposal to add the .lib to line 272 in src/CMakeLists.txt (submitted in r3948) . Two lines later in the code, the extension is used and that code has been there for some time. Here's the svn "blame" output:

```
   268      2746 robinwmill IF( EXIV2_ENABLE_PNG )
   269      3008 robinwmill     IF( ZLIB_FOUND )
   270      3008 robinwmill         IF( MSVC )
   271      3767 robinwmill             if ( EXIV2_ENABLE_SHARED )
   272      3948 robinwmill                 TARGET_LINK_LIBRARIES( exiv2lib optimized ${ZLIB_LIBRARIES}.lib de
bug ${ZLIB_LIBRARIES}d.lib )
   273      3008 robinwmill             else()
   274      3008 robinwmill                 TARGET_LINK_LIBRARIES( exiv2lib optimized zlibstatic.lib debug zli
bstaticd.lib )
```

```
   275      3008 robinwmill             endif()
   276      3008 robinwmill         ELSE()
   277      3008 robinwmill             TARGET_LINK_LIBRARIES( exiv2lib ${ZLIB_LIBRARIES} )
   278      3008 robinwmill         ENDIF()
   279      3008 robinwmill     ENDIF()
   280      2746 robinwmill ENDIF()
```

Line 274 was only reformatted (without tabs) on 3008. It appears to have been in the code-base since mid-2012 and possibly longer.

I wonder how that FIND_PACKAGE(ZLIB) magic works? There are different versions of FindZLIB.cmake on my laptop (yes, I know it's a Mac). The point is we know we have different versions of CMake. Could different versions of FindZLIN.cmake be the culprit?

```
531 rmills@rmillsmbp:~/gnu/exiv2/trunk $ ls -alt $(locate "*.cmake" | grep -i zlib)
-rw-r--r--  1 root    admin  4214 13 Aug 15:57 /opt/local/share/cmake-3.3/Modules/FindZLIB.cmake
-rw-r--r--@ 1 rmills  admin  4214 10 Mar  2015 /Applications/CMake.app/Contents/share/cmake-3.2/Modules/FindZL
IB.cmake
-rw-r--r--+ 1 rmills  staff  4254 22 Jul  2014 /Users/rmills/gnu/libssh/libssh-0.6.3/cmake/Modules/FindZLIB.cm
ake
-rwxr-xr-x@ 1 rmills  staff  4254 16 Jul  2013 /Users/rmills/gnu/libssh/libssh.0.5.4/cmake/Modules/FindZLIB.cm
ake
-rw-r--r--@ 1 rmills  staff   385 21 Jun  2013 /Users/rmills/gnu/curl/curl-7.31.0/CMake/FindZLIB.cmake
-rw-r--r--@ 1 rmills  staff  2033  7 Nov  2007 /Users/rmills/gnu/podofo/podofo-0.9.3/cmake/modules/FindZLIB.cm
ake
532 rmills@rmillsmbp:~/gnu/exiv2/trunk $
```

#18 - 23 Sep 2015 11:55 - Emmanuel d'Angelo

It seems that in cmake 3.1.0 there was a change in FindZLib that has to deal with the library extensions. I don't completely understand the cmake language, but it could explain the different behaviour that I observed (we use cmake 3.3.1, so posterior to this commit). If it's the root cause, then we can probably add a conditional statement based on cmake version.

#19 - 23 Sep 2015 14:45 - Robin Mills

Thanks for digging into the changes in versions of FindZLIB.cmake.

I'll have a look into that later today. I'm not very enthusiastic to use the CMake version to decide about the .lib because I believe scripts like FindZLIB.cmake are not always found in the default cmake location. For sure, curl, libssh, podofo have variants of that file in their source tree. This is complex magic (complex magic = more that one recipe for the magic).

CMake has commands for string regex matching. Maybe all I need to do is to something like the following pseudo code:

```
ZLIB_OPTIM = ${ZLIB_LIBRARIES}
ZLIB_DEBUG = ${ZLIB_LIBRARIES}d
if ( not ${ZLIB_OPTIM}( REGEX MATCH "lib$") )
   ZLIB_OPTIM = $ {ZLIB_OPTIM}.lib
endif()
if ( not ${ZLIB_DEBUG}( REGEX MATCH "lib$") )
   ZLIB_DEBUG = $ {ZLIB_DEBUG}.lib
endif()
TARGET_LINK_LIBRARIES( exiv2lib optimized ${ZLIB_OPTIM} debug ${ZLIB_DEBUG} )
```

I hope something like that will make you and T happy.

#20 - 23 Sep 2015 15:45 - Emmanuel d'Angelo

The FindZLib script from Cmake seems to be shipped with standard installs. I still have some work on Linux to do today and tomorrow morning, but I'll try to reboot on windows and compare the behaviour between different cmake versions tomorrow afternoon (continental Europe time, that is).

Also, I though a bit about this unicode flag (besides the fact that I prefer explicit flags instead of magic). We do a lot of work from the git bash interpreter, but the final build of the lib can be done directly in windows's command tool. This could explain why the environment doesn't get detected as genuine windows in our setup.

#21 - 23 Sep 2015 15:48 - Emmanuel d'Angelo

Oh, and maybe you can't make everyone happy at the same time: https://xkcd.com/1172/ ;)

#22 - 23 Sep 2015 18:26 - Robin Mills

No question, we can't please everybody all of the time. However we can aim to make everybody happy!

I'm in England, so I'm one hour behind you.

I investigated the changes between FindZLIB.cmake on 3.0.0 and 3.1.0. I didn't see anything odd or windows specific. I have Visual Studio 2013 and 2015 installed and working properly now and I'll run some builds on Windows/CMake. I'll make you and T happy on Thursday or Friday. You'll see.

The interactions between the build tools and your shell are:

1. The arguments you pass to the tools
2. Your environment strings

I believe the Git/bash shell is about the same as Cygwin which is the windows shell on our buildServer. Our buildServer is Jenkins and runs on a Mac. The server has 3 "nodes": MacOS-X, Linux and Windows (Linux and Windows are VMs running on the Mac). Jenkins knows the IP addresses of the nodes and uses ssh to run scripts on the nodes. The Windows node is a Cygwin server running bash. The build script is jenkins_build.sh

To build with MSVC, jenkins_build.sh invokes CMD.exe to run jenkins_build.bat which runs devenv.exe.
To build on MinGW, jenkins_build.sh invokes the MinGW bash interpreter to perform the build.

The script buildJenkins.sh is the most complex bash script I've ever engineered. (complex != complicated. complex = invokes other magic). It's not intended to be magic. It's how I made it work reliably.

#23 - 24 Sep 2015 11:06 - Robin Mills

- % Done changed from 0 to 30

- Estimated time set to 100.00 h

#24 - 24 Sep 2015 12:38 - Emmanuel d'Angelo

I found a more correct way of designating a custom zlib folder to the cmake build script: we were using the option -DZLIB_LIBRARY=some\path\to\zlib (which gave us troubles) to set the path to the library and the name of the file to use, but using instead the option -DZLIB_ROOT=path\to\custom\zlib\folder yields the correct behaviour for the release build.
By correct, I mean that the generated Visual Studio project file tries to link with zlib.lib in release. In debug, it tries to link with zlib.libd (we have the release and debug libs in the same folder and use the cmake debug postfix set to d).

I'm still investigating how to designate a particular library file in this root folder.

Thanks for your time!

#25 - 24 Sep 2015 13:53 - Robin Mills

- File SteelBeam.jpg added

Thanks very much for this insight. That zlib1d.lib stuff is an obstacle that we could do without - however it's part of the zlib way of doing things. I was struggling to build with VisualStudio 2013/cmake this morning and dealing with building stuff. We are building/remodelling our house. We've got the steel beam to into place this morning (photo below).

I'm enjoying working with you. This issue has reminded me why I dislike CMake. However I think we'll be successful here and maybe get to like CMake (unlikely).

For your information: the 3 libraries for remoteIO (curl, openssl and libssh) are optional. There is a basic implementation of http provided (in http.cpp). So all versions of libexiv2 support http. You only need curl and friends for more complex protocols such as https, ftp, and ssh.

Thanks for keeping me aware of your progress. I'll do some work this evening and give you an update. SteelBeam.jpg

#26 - 24 Sep 2015 14:20 - Emmanuel d'Angelo

Hi Robin!

I have good news for you! I think I finally found the issue that I was struggling with.

I checked the output of the FindZLib module once again, since T Modes found that it worked on his side. It is actually **also** working for us, except that... we build the release then the debug versions successively in the same script. As a consequence, I could only have a look at the Debug build output, and I'm sorry but there is a small bug that leaped into exiv2's debug build linker command.

If you go to src/CmakeLists.txt, line 272, you can read:

```
TARGET_LINK_LIBRARIES( exiv2lib optimized ${ZLIB_LIBRARIES} debug ${ZLIB_LIBRARIES}d )
```

i.e., Release builds use the direct output from FindZLib and Debug builds append a d at the end. This causes errors (at least on windows) if we pass the full name + extension of zlib as suggested by T Modes:

- in Release builds the linker will search for @${ZLIB_LIBRARIES} = zlib.lib$ => it works nicely
- in Debug builds the linker will search for @${ZLIB_LIBRARIES}d = zlib.libd$ => link failure.

#27 - 24 Sep 2015 14:48 - Emmanuel d'Angelo

Sorry, I tried to *Preview* and hit the wrong button...

Anyway, back to the story. From here, there are several possibilities to fix:

- changing the linker line to TARGET_LINK_LIBRARIES(exiv2lib ${ZLIB_LIBRARIES}) since it works, and users who want a debug build need to set it manually to zlibd.lib (I couldn't get cmake to use it instead, although we use the option -DCMAKE_DEBUG_BUILD_POSTFIX=d)
- stripping the lib extension from the ${ZLIB_LIBRARIES} before adding the d.

This could be (should be?) conditional to the fact that the user has provided a direct path to zlib.

Sorry, I'm a bit confusing and thinking out loud, but I hope this helps you!

Thanks,

Emmanuel

#28 - 24 Sep 2015 15:55 - Robin Mills

Thanks for letting me know about your discoveries. I hope the user does NOT have to specify stuff like DCMAKE_DEBUG_BUILD_POSTFIX. To the user, it should "just work". There is pattern match/replace stuff in CMake very much like sed:

```
$ echo '/how/now/brown/cow/zlib.lib' | sed -E -e 's/(.*)(\.lib)/\1d\2/'
/how/now/brown/cow/zlibd.lib
$ echo '/how/now/brown/cow/zstaticlib.lib' | sed -E -e 's/(.*)(\.lib)/\1d\2/'
/how/now/brown/cow/zstaticlibd.lib
$
```

I was rather surprised to discover that I instructed the user to edit expat-2.1.0/CMakeLists.txt. I remember the pain involved, but not the reason. Presumably it was the only way I could get it to work at all.

This morning I saw the following very helpful and totally meaningless message from CMake:

```
-- Could NOT find EXPAT (missing:  EXPAT_LIBRARY) (found version "2.1.0")
```

WTF?

I like Visual Studio. There's a simple linear relationship between foo.vcproj files, the GUI, and the generated command-lines. I have never worked inside .vcxproj files and assume they are much the same. Unlike Xcode project.pbxproj file, it's easy (and safe) to edit Visual Studio project files.

With CMake, you have a weird declarative language that imposes a build abstraction from which the .vcproj files are generated. The FindFOO.cmake files are volatile/random magic at best. The projects which are generated by CMake don't feel "Native" to me. They have weirdo build targets and config names. And the user must generate different project files for every version of Visual Studio using generator names which are very difficult to type or remember. I don't believe you can create a single project from which the user can choose architecture (32/64) or library types (static/shared). And of course it keeps running CMake when it builds - so if you "tweak" a project setting in the GUI, CMake will "untweak" it for you. All of this adds up to horrible.

Anyway, thanks to you, we're making progress. Several users have complained about Exiv2 support for CMake/Visual Studio. They criticise and tell me that CMake is easy and great. You're the first to provide constructive help. Thank You very much.

#29 - 25 Sep 2015 07:34 - Emmanuel d'Angelo

I'm a bit surprised by your EXPAT errors. Currently, I can successfully build exiv2 on Windows with custom zlib + expat using the following script (omitting non-relevant parts):

```
cmake ^
-G "Visual Studio 12 2013 Win64" ^
-DCMAKE_BUILD_TYPE=Release ^
-DCMAKE_INSTALL_PREFIX=%tmp_install_path% ^
-DZLIB_INCLUDE_DIR=%some_include_path%\zlib ^
-DZLIB_LIBRARY=%some_lib_path%\zlib.lib ^
-DEXPAT_INCLUDE_DIR=%some_include_path%\expat ^
-DEXPAT_LIBRARY=%some_lib_path%\libexpat.lib ^
%build_path%

devenv %build_path%\build\exiv2.sln /Build "Release|x64" /Project INSTALL
```

I guess that parts of the errors on our side come from the fact that we specify everything (build type + library paths and names) and that we should maybe delegate this to the call to devenv instead. This should hold for the CMAKE_DEBUG_POSTFIX option too. I still think it's ok to specify it, it's actually useful for the build result and I don't think that it has any impact on the input library names.
We should have something working and robust for 0.26, the goal is not too far anymore :-)

As an off topic, on the other hand I like Xcode and its file format (most of its core is actually legacy from the NextStep dev tools). Not a big fan of vcxproj formats, but it's probably because I'm not used to windows build environment. I like studying these formats by looking at the changes in my source control system (git), it's a nice way to see the impact of clicking on buttons.

I also came to like cmake as a replacement of hand crafted Makefiles + scripting magic to explore folders, sub-folders, add targets... It's a nice tool to use on the command line, ie, a nice tool for *nix and mac os. What I really dislike about cmake is its lack of documentation (besides the usual *getting started* stuff). I'm not even sure that buying the book is a good idea, since it's based on older versions of the tool.

#30 - 25 Sep 2015 09:37 - Robin Mills

Thanks very much. I've almost got it to build with your command-line. That's great. I have to get on with the house building this morning, so I'll have a more careful look later today.

I have a feeling that the result of our cooperation is going to be a rewrite of README-CMAKE for Visual Studio and perhaps a few little code changes. What a fantastic improvement that will be.

I have the CMake book. I bought it to work on this for Exiv2. The book is a reprint of the man pages on the Kitware web-site. I see they are selling them cheaply today, so I expect they have a new edition coming soon. I don't know if I would recommend the book or not. I can send you my copy (I never look at it) in the mail and you can decide. Email me your address: robin@clanmills.com

The Xcode xcconfig feature is really great. Sadly, if you edit almost anything in project.pbxproj, Xcode refuses to open the file. And that file is full of GUIDS.

Autotools is a black art. Sure it works, although nobody knows how or why. The next time you are on a plane (be it an Airbus or a Boeing), just remember that the flight control software (and the air-traffic control software) was almost certainly build using this stuff. Pray for a safe landing.

#31 - 25 Sep 2015 14:22 - Emmanuel d'Angelo

There are two different cases to handle:

- users who just build the library using regular, system-wide libraries, in which case the cmake should already work kind of automagically
- users like our company who customize a lot of things. In this case, I don't think that the cmake script should over-guess the configuration, as we are supposed to know what we are doing. It should merely try to respect the parameters that we give as inputs (which is already almost done, except for this d suffix in the debug configuration zlib).

I will have a little spare time next week(s) to finalize all this and update the README-CMAKE.

Thanks for the offer! But I'll wait for a next version of the book (if any), and I'm already disliking the man pages (I find the policies description confusing at best).
And I got my first degree from an aerospace engineering school (I came to software engineering later in my career), I do already pray for a lot of things when I fly ;-)

#32 - 25 Sep 2015 14:56 - Robin Mills

Aerospace, eh! I did Civil Engineering and worked as a Structural Engineer for a couple of years after University. I think there's a lot in common between Structures and Software. I retired last year after 41 years of hard work. The last 14 years were in Silicon Valley (Adobe, Apple, Google). I never miss work.

When I worked on the PostScript Interpreter, we'd often put pixels on the wrong place on the page. One pixel on a A4 page at 600dpi is about the same resolution as a city like Moscow relative to planet Earth. We are in more danger from our software engineers than the enemy. I have my doubts about the reliability and accuracy of a Trident Missile. Let's hope the politicians are never stupid enough to use the damn things.

It would be totally awesome for you to do a little work on README-CMAKE. Thanks very much.

Have a nice weekend. Very nice weather forecast for England this weekend. I hope the weather's good for you.

#33 - 25 Sep 2015 16:27 - T Modes

> This morning I saw the following very helpful and totally meaningless message from CMake:
>
> -- Could NOT find EXPAT (missing: EXPAT_LIBRARY) (found version "2.1.0")
>
> WTF?

Simple ;-): It has found the expat header, they are from version 2.1.0, but the expat library was not found. As a result expat is marked as "not found", because without the lib you can't link.

> I don't believe you can create a single project from which the user can choose architecture (32/64) or library types (static/shared).

Yes, you can't choose architecture, but different library types are possible. You need only one add_library call for each type. Something like

```
if( EXIV2_ENABLE_STATIC )
ADD_LIBRARY( exiv2lib_static STATIC ${LIBEXIV2_SRC} ${LIBEXIV2_HDR} )
endif()
if(EXIV2_ENABLE_SHARED)
ADD_LIBRARY( exiv2lib SHARED ${LIBEXIV2_SRC} ${LIBEXIV2_HDR} )
endif()
```

(you need only the check before that at least one option is enabled.)

At Hugin we are using a modified FindZlib.cmake module (see http://hg.code.sf.net/p/hugin/hugin/file/8f75794f051f/CMakeModules/FindZLIB.cmake)

This handles also the debug variants. Maybe this can give you some ideas.

@Emmanuel

> cmake ^
> -G "Visual Studio 12 2013 Win64" ^
> -DCMAKE_BUILD_TYPE=Release ^

<snip>

I think you don't need to specify CMAKE_BUILD_TYPE. The project files contain always all 4 variants. You select the variant in Visual Studio or on the command line (as you do with "devenv build_path\build\exiv2.sln /Build "Release|x64" /Project INSTALL"). So you leave the parameter on the cmake command line.

#34 - 25 Sep 2015 17:26 - Robin Mills

Thank you, T. This is very helpful. Cooperation is 100 times more effective than criticism. We're really moving now on this issue. Thanks for helping.

And thanks for explaining the message: *Could NOT find EXPAT (missing: EXPAT_LIBRARY) (found version "2.1.0")* That makes good sense when you explain it.

I know that CMake can build shared or static libraries. However they are CMake options. Once you're in Visual Studio, can you say "build shared", "build static". I like the way things are in the existing msvc2005 projects. You open the solution and can select 32/64, shared/static, debug/release. So it's a one stop shop with the 8 common builds available. Is it possible to generate this with CMake?

#35 - 25 Sep 2015 17:59 - T Modes

> You open the solution and can select 32/64, shared/static, debug/release. So it's a one stop shop with the 8 common builds available. Is it possible to generate this with CMake?

32/64 bit: not possible in one solution
shared/static: I think this is not possible with a simple switch in the MSVC project file. Workaround is to add 2 library variants (2 calls to add_library, but maybe more work intensive if there are further sub-libraries.) But in this case always 2 variants are built (or the variants you build with CMake options). This variant is used by zlib's CMake system (you get an zlib and a zlibstatic target).
debug/release: works out of the box with CMake. The generated MSVC project files contains (always?) 4 solutions sets: Debug/Release/MinSizeRel/RelWithDebInfo

Maybe I can help you with the FindFOO magic.
As far as I understood there are mainly 4 parts:
1. Look for path of header files, e.g.

```
FIND_PATH(FOO_INCLUDE_DIR
NAME foo.h
PATHS search paths...
)
```

2. Look for the library

```
FIND_LIBRARY(FOO_LIBRARIES
NAMES foo libfoo
PATHS search paths ...
)
```

(in Hugin we are using a own written variant for better handling of debug libs:

```
include(FindLibraryWithDebug)
find_library_with_debug(FOO_LIBRARIES
WIN32_DEBUG_POSTFIX d
NAMES foo libfoo
PATHS search paths
)
```

3. Check if all was found (this handles also FOO_FOUND variable) and handle standard arguments of find_package call (e.g. required, quiet switches)

```
include(FindPackageHandleStandardArgs)
find_package_handle_standard_args(FOO DEFAULT_MSG FOO_INCLUDE_DIR FOO_LIBRARIES)
```

4. Often the variables are marked as advanced, so there are not shown by default in the GUI (except you tick the advanced box), when running from the command line this has no consequences.

```
MARK_AS_ADVANCED(FOO_INCLUDE_DIR FOO_LIBRARIES )
```

(Adding version checking depends on the library, because there are different ways how libraries define their version numbering.)

#36 - 25 Sep 2015 19:06 - Robin Mills

Thank You once more. Very helpful. My concerns about CMake are totally restricted to Visual Studio. CMake works very well on Cygwin/MinGW/Mac/Linux with GCC and Clang. And it's clear how it works on those platforms. No autotools magic required. We had students working with us on Google Summer of Code in 2013 and they all liked CMake much more than the autotools. I took care of Visual Studio on those projects, so they didn't offer an opinion about Visual Studio.

> 2. Look for the library
> FIND_LIBRARY(FOO_LIBRARIES
> NAMES foo libfoo
> PATHS search paths ...
>)

What are the search paths on Windows?

#37 - 26 Sep 2015 12:52 - T Modes

> What are the search paths on Windows?

On Windows there is not a single place where the libs are stored. So we have to take some reasonable assumptions: assume the user store all libraries in the same directory level side by side.
So in the main CMakeLists.txt we store the main path:

```
# For Win32 builds, assume that all of the libraries are located
# one directory above the current CMakeLists directory
IF(WIN32)
  STRING(REGEX REPLACE "(.*)/[^/]+$" "\\1" work "${CMAKE_SOURCE_DIR}")
  # create the cache entry, editable by the user
  SET(SOURCE_BASE_DIR ${work} CACHE FILEPATH "parent dir of hugin source root")
ENDIF(WIN32)
```

and in FindFOO.cmake you could use e.g.
${SOURCE_BASE_DIR}/foo/include
${SOURCE_BASE_DIR}/foo/lib
as search path for the header and lib.

#38 - 26 Sep 2015 19:59 - Robin Mills

Well this seems to be a big step forward. I'm getting errors because the linker cannot find expat. I'll figure this out on Sunday.

Gentlemen: I've promoted you both to be an 'Administrator' of the Forum. This grants you permission to edit posts. So, T, may I ask you to fix the formatting in #37 - or (better) - delete #38 and update #37 to include the information in #38.

I do not have an issue with zlibd.lib and zlib1.lib. However I have to resolve the expat puzzle first. Here are the link commands for exiv2.exe.

Debug:

```
/OUT:"C:\cygwin64\home\rmills\gnu\exiv2\trunk\bin\x64\Dynamic\Debug\exiv2.exe" /MANIFEST /NXCOMPAT /PDB:"C:/cy
gwin64/home/rmills/gnu/exiv2/trunk/bin/x64/Dynamic/Debug/exiv2.pdb" /DYNAMICBASE "kernel32.lib" "user32.lib" "
gdi32.lib" "winspool.lib" "shell32.lib" "ole32.lib" "oleaut32.lib" "uuid.lib" "comdlg32.lib" "advapi32.lib" ".
.\bin\x64\Dynamic\Debug\exiv2.lib" "..\bin\x64\Dynamic\Debug\xmp.lib" "..\..\expat-2.1.0\Debug\expat.lib" "..\
..\zlib-1.2.7\Debug\zlibd.lib" /IMPLIB:"C:/cygwin64/home/rmills/gnu/exiv2/trunk/bin/x64/Dynamic/Debug/exiv2.li
b" /DEBUG /MACHINE:X64 /INCREMENTAL /PGD:"C:\cygwin64\home\rmills\gnu\exiv2\trunk\bin\x64\Dynamic\Debug\exiv2.
pgd" /SUBSYSTEM:CONSOLE /MANIFESTUAC:"level='asInvoker' uiAccess='false'" /ManifestFile:"exiv2.dir\Debug\exiv2
.exe.intermediate.manifest" /ERRORREPORT:PROMPT /NOLOGO /LIBPATH:"C:/cygwin64/home/rmills/gnu/exiv2/trunk/bin/
x64/Dynamic/Debug" /LIBPATH:"C:/cygwin64/home/rmills/gnu/exiv2/trunk/bin/x64/Dynamic/Debug/Debug" /TLBID:1
```

Release:

```
/OUT:"C:\cygwin64\home\rmills\gnu\exiv2\trunk\bin\x64\Dynamic\Release\exiv2.exe" /MANIFEST /NXCOMPAT /PDB:"C:/
cygwin64/home/rmills/gnu/exiv2/trunk/bin/x64/Dynamic/Release/exiv2.pdb" /DYNAMICBASE "kernel32.lib" "user32.li
b" "gdi32.lib" "winspool.lib" "shell32.lib" "ole32.lib" "oleaut32.lib" "uuid.lib" "comdlg32.lib" "advapi32.lib
" "..\bin\x64\Dynamic\Release\exiv2.lib" "..\bin\x64\Dynamic\Release\xmp.lib" "..\..\expat-2.1.0\Release\expat
.lib" "..\..\zlib-1.2.7\Release\zlib.lib" /IMPLIB:"C:/cygwin64/home/rmills/gnu/exiv2/trunk/bin/x64/Dynamic/Rel
ease/exiv2.lib" /MACHINE:X64 /INCREMENTAL:NO /PGD:"C:\cygwin64\home\rmills\gnu\exiv2\trunk\bin\x64\Dynamic\Rel
ease\exiv2.pgd" /SUBSYSTEM:CONSOLE /MANIFESTUAC:"level='asInvoker' uiAccess='false'" /ManifestFile:"exiv2.dir\
Release\exiv2.exe.intermediate.manifest" /ERRORREPORT:PROMPT /NOLOGO /LIBPATH:"C:/cygwin64/home/rmills/gnu/exi
v2/trunk/bin/x64/Dynamic/Release" /LIBPATH:"C:/cygwin64/home/rmills/gnu/exiv2/trunk/bin/x64/Dynamic/Release/Re
lease" /TLBID:1
```

#39 - 25 Oct 2015 18:06 - Daniel Kaneider

I extended the contrib/msvc build scripts. Now I was able to compile exiv2 with complete webready support (including ssl, ssh, curl). Maintaining old VS project files along with CMake is still cumbersome. Therefore I merged the exv_msvc content with config.h.cmake (used for *nix cmake)

@Robin: is there some jenkins build (*nix or msvc) which is already using CMake? It would be very helpful.

#40 - 01 Nov 2015 18:14 - Daniel Kaneider

exiv2_.svg?branch=travis

After playing around a little bit with github I activated some CI builds they offer via Travis CI free of charge (for open sourced, github project).
Builds are currently successful for Linux and OSX, both for gcc and clang. This both includes standard build and with webready. Since the update of the github repo is done through svn2github, the CI is rather semi-automatic. But, it fully uses CMake for executing the builds (.travis.yml).

#41 - 01 Nov 2015 19:41 - Thomas Beutlich

Nice. Here's the link https://travis-ci.org/danielkaneider/exiv2_

#42 - 01 Nov 2015 19:50 - Thomas Beutlich

Btw, if you add danielkaneider@b7c8b350-86e7-0310-a4b4-de8f6a8f16a3 to you GitHub emails you'll easily recognize your contributions to exiv2_.

#43 - 01 Nov 2015 22:09 - Robin Mills

I'm rather puzzled by this Travis business. I thought the focus of this topic is to build Exiv2/MSVC/CMake. At the moment, I'm been 100% unable to get the build.cmd and env.cmd to work. And now we're shifted focus to a continuous build server for Linux and Mac. The Jenkins server is already performing this task well.

In what way does Travis help with the topic: "CMake toolchain for windows"?

#44 - 02 Nov 2015 20:54 - Daniel Kaneider

Robin, I get it. Nobody seems to care about CMake. Why else does Jenkins still use the make projects?
This issue is not just about CMake on windows, but also for Unix/Mac. Some of the features (mainly testing) are still not at the point they should be. So, Travis adds some CI builds to check if some builds work on Unix/Mac. You are free to create an additional build config for CMake in Jenkins. Just have a look at the .travis.yml to see which CMake commands I currently execute.

And regarding CMake on Windows, maybe it's time that you send me some log/build output. I cannot do more than offering my help. At least on my machine I can confirm that it does work (VS2013, Cmake 2.8 and CMake 3.3 for Win64) :-)

#45 - 17 Nov 2015 17:01 - Robin Mills

- % Done changed from 30 to 50

Daniel and I had a one-2-one discussion on Skype about this on 2015-11-09. Very helpful. Daniel actually has this thing building.

I've copied Daniel's scripts CP -r contrib/build contrib/cmake and I'm developing them. However this does not imply that I will replace Daniel's scripts with mine. The purpose of this exercise is to restore my confidence the CMake/Visual Studio does work. How we intend to proceed to CMake/Visual Studio in v0.26 is still to be determined and Daniel and I will discuss later (I hope before Christmas 2015).

build.cmd now has a command-line parser and successfully builds with VS2005/64. I haven't succeeded in getting it to build with NMake and haven't tried other versions of Visual Studio. I have also disabled building/linking webready and associated libraries at this time. A very promising start. Thanks to Daniel for such great support and encouragement.

#46 - 26 Nov 2015 21:27 - Robin Mills

- % Done changed from 50 to 60

Here's what Daniel and I have discussed by email:

1) Robin will build and test CMake + Visual Studio to build exiv2 + zlib + expat on VS 2005/8/10/12/13/15 (done and mostly OK - see below).
2) Daniel to get CMake + Visual Studio to build 'webready' on VS 2005/8/10/12/13/15
3) Robin with build and test CMake + Visual Studio to build exiv2 + zlib + expat + webready on VS 2005/8/10/12/13/15
4) When the system has passed item (3) we will consider deprecating msvc2005 solution and project files

The script cmakeRebuildAll.cmd is working fine and almost everything builds OK. However there's something wrong with my installation of VS2010 and CMake refuses to work with him. I'll investigate and fix this. I'm also a little surprised that the test output from bash is echoed correctly to the console - however it escapes from tee.exe and isn't recorded.

Robin intends to continue to develop VS Solution/Projects for v0.26 (#1121). If we agree to deprecate msvc2005, this will be announced with v0.26. However vs2005 (and vs2003) will ship with v0.26.

Although Gilles has requested that we support CMake/nmake, I don't believe this is necessary. cmake can build from a shell using cmake --build .

#47 - 27 Nov 2015 17:43 - Robin Mills

- File rebuildAll.txt added

I've rerun the tests. Still something wrong on the laptop with 2008 and 2010. I've investigate next week. Very pleased with the progress.

#48 - 28 Nov 2015 11:23 - Robin Mills

Still trouble with VS 2012, 2013 and 2015. Very frustrating and tedious.

I'm beginning to wonder if it's possible to install all those VS Editions on the same machine. Between the laptop and the buildserver, I believe I have a working copy of every version of Visual Studio. However I can't get them all to work on a single machine.

```
84 rmills@rmillsmm-w7:/c/gnu/exiv2/build $ grep -e ^generator -e ^version -e ^bits -e FAILED rebuild.txt
generator = Visual Studio 11 2012 Win64
generator = Visual Studio 10 2010 Win64
bits=64
version=10.00 (2010)
generator = Visual Studio 14 2015 Win64
generator = Visual Studio 12 2013 Win64
Done Building Project "C:\gnu\exiv2\build\temp\exiv2\src\exiv2lib.vcxproj" (default targets) -- FAILED.
Done Building Project "C:\gnu\exiv2\build\temp\exiv2\samples\addmoddel.vcxproj" (default targets) -- FAILED.
Done Building Project "C:\gnu\exiv2\build\temp\exiv2\ALL_BUILD.vcxproj" (default targets) -- FAILED.
Build FAILED.
generator = Visual Studio 9 2008 Win64
bits=64
version=9.00 (2008)
generator = Visual Studio 8 2005 Win64
bits=64
version=8.00 (2005)
generator = Visual Studio 11 2012
generator = Visual Studio 10 2010
bits=32
version=10.00 (2010)
generator = Visual Studio 14 2015
generator = Visual Studio 12 2013
Done Building Project "C:\gnu\exiv2\build\temp\exiv2\src\exiv2lib.vcxproj" (default targets) -- FAILED.
Done Building Project "C:\gnu\exiv2\build\temp\exiv2\samples\addmoddel.vcxproj" (default targets) -- FAILED.
Done Building Project "C:\gnu\exiv2\build\temp\exiv2\ALL_BUILD.vcxproj" (default targets) -- FAILED.
Build FAILED.
generator = Visual Studio 9 2008
bits=32
version=9.00 (2008)
generator = Visual Studio 8 2005
bits=32
version=8.00 (2005)
85 rmills@rmillsmm-w7:/c/gnu/exiv2/build $
```

#49 - 28 Nov 2015 21:13 - Robin Mills

I've discovered that stdint.h on VS 2012 and 2013 cause many compiler errors. Fixed that. And I've enhanced cmakeBuild to build into directories such as dist/2008/x64/dll/Release/. This will make it easy to have a "daily/weekly/monthly" builds archived for users. r4060

Still have trouble with VS2010 on the laptop. Here's how I test that the compiler is working.

```
C:\temp>type foo.cpp
#include <stdio.h>
int main(int,const char**)
{
        return printf("%d sizeof(void*) = %d",_MSC_VER,sizeof(void*));
}

C:\temp>
```

And I run him through a couple of loops:

```
C:\temp>for %a in (64 32) do for %v in (2005 2008 2010 2012 2013 2015) do cmd /C "vcvars %v %a>NUL && nmake -a
 foo.exe >
/NUL 2>/NUL && foo.exe"

C:\temp>for %v in (2005 2008 2010 2012 2013 2015) do cmd /C "vcvars %v 64>NUL && nmake -a foo.exe >/NUL 2>/NUL
 && foo.ex
e"

C:\temp>cmd /C "vcvars 2005 64>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
```

```
1400 sizeof(void*) = 8
C:\temp>cmd /C "vcvars 2008 64>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1500 sizeof(void*) = 8
C:\temp>cmd /C "vcvars 2010 64>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1600 sizeof(void*) = 8
C:\temp>cmd /C "vcvars 2012 64>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1700 sizeof(void*) = 8
C:\temp>cmd /C "vcvars 2013 64>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1800 sizeof(void*) = 8
C:\temp>cmd /C "vcvars 2015 64>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1900 sizeof(void*) = 8
C:\temp>for %v in (2005 2008 2010 2012 2013 2015) do cmd /C "vcvars %v 32>NUL && nmake -a foo.exe >/NUL 2>/NUL
 && foo.ex
e"

C:\temp>cmd /C "vcvars 2005 32>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1400 sizeof(void*) = 4
C:\temp>cmd /C "vcvars 2008 32>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1500 sizeof(void*) = 4
C:\temp>cmd /C "vcvars 2010 32>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1600 sizeof(void*) = 4
C:\temp>cmd /C "vcvars 2012 32>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1700 sizeof(void*) = 4
C:\temp>cmd /C "vcvars 2013 32>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1800 sizeof(void*) = 4
C:\temp>cmd /C "vcvars 2015 32>NUL && nmake -a foo.exe >/NUL 2>/NUL && foo.exe"
1900 sizeof(void*) = 4
```

Visual Studio 2010 (version 1600) is alive and well. CMake (3.4.0) is working with VS 2010 on the buildserver. CMake (3.3.2 and 3.4.0) refuses to work on the laptop with VS2010. I'll sync the VS2010 directory from the buildserver to the laptop and hopefully the tools will be working everywhere.

#50 - 29 Nov 2015 22:39 - Robin Mills

- % Done changed from 60 to 70

CMake/Visual Studio is building correctly (and passing the test suite) on 2005/8/10/12/13/15 x64 and win32 on both the laptop and buildserver using contrib/cmake/cmakeRebuildAll.cmd

Over to you now Daniel to get webready to build.

#51 - 05 Dec 2015 20:21 - Robin Mills

- % Done changed from 70 to 80

Three Cheers for Me cmakeBuild now builds webready by using nmake to build curl. r4066 Setting the done to 80%. Almost there.

I'll have to test a bit more and I'm guessing it will take about 3-4 hours to run 64/32 bit builds on all 6 versions of Visual Studio.
I will build openssl with msvc2005 and package it for download from team/libraries.
I'm not going to spend any more effort trying to get curl to build with cmake.

#52 - 07 Dec 2015 16:44 - Robin Mills

r4072 Builds 32/64 on Visual Studio 2005/8/10/12/15 with/without --webready.
r4070 Added pre-built openssl-1.0.1p for vs2005
Everything builds including curl and libssh.

Daniel has reported a libssh/VS2015 build issue: https://red.libssh.org/issues/205 I have updated that issue with my work-around.

I have reported a libssh/VS2005 (&2008) build issue and work-around: https://red.libssh.org/issues/214

Because cmakeBuild.cmd gets the libssh-0.72.0 source code from svn://dev.exiv2.org/svn/team/libraries, I have updated these to include the fixes for 205 and 214. However, any 'vanilla' libssh bundle will not have these fixes.

I still have three remaining tasks:

1. get Jenkins to build and publish artefacts and logs
2. --static builds (not tested at all)
3. more testing

#53 - 12 Dec 2015 19:30 - Robin Mills

- % Done changed from 80 to 90

We are now publishing the build artefacts and logs on the build server. The buildserver UI has been updated to provide a link to daily/weekly/monthly

builds. The script jenkins_daily.sh also includes a ReadMe.txt for the build which includes instructions on the use of the build. I have a couple of outstanding matters:

1. --static builds (still not tested)
2. publish mingw builds

At the moment, I only build VS 2013/x64 builds. It's a tiny script change to publish 2005/8/10/12/13/15 (call cmakeBuildAll instead of cmakeBuild in jenkins_daily.sh). However this will push the daily build time to 1.5 hours AND increase the msvc bundle from 4mb to 50mb. However, I think I will do this. The buildserver is sitting idle in the middle of the night. Eventually we'll have 50 daily, 50 weekly and 60 monthly builds which will consume less than 10Gbtyes on the build server. The machine has a 1Tbyte Fusion drive (hybrid SSD/Hard-drive storage unit).

#54 - 31 Mar 2016 10:30 - Robin Mills

- % Done changed from 90 to 80

- Estimated time changed from 100.00 h to 150.00 h

#55 - 17 Apr 2016 19:10 - Robin Mills

- % Done changed from 80 to 90

I had a quite a lot of trouble getting all 12 MSVC builds to execute with CMake and to pass the daily_test.sh. It's been stable now for two weeks. I think CMake/Visual Studio is more-or-less complete. This has been a lot of work, however it is one of the major new features of the v0.26 release.

#56 - 24 Apr 2016 05:13 - Robin Mills

- Status changed from Assigned to Closed

- % Done changed from 90 to 100

Moving towards code complete for v0.26. CMake build on all supported platforms is stable and being used by the buildserver for the daily build.

Files

CMake-nmake-toolchain.patch	3.62 KB	15 Mar 2015	Daniel Kaneider
win-build-cmake.zip	1.82 KB	15 Mar 2015	Daniel Kaneider
SteelBeam.jpg	185 KB	24 Sep 2015	Robin Mills
rebuildAll.txt	1.29 MB	27 Nov 2015	Robin Mills