Install, Compile, and Test



Table of Contents

[1 Overview 6](#_Toc465264663)

[2 Building GenICam on Windows 6](#_Toc465264664)

[2.1 Tools Required for Building the Libraries 6](#_Toc465264665)

[2.2 Tools Required for Building the Documentation 7](#_Toc465264666)

[2.3 Required Tools for Building the Python Bindings 7](#_Toc465264667)

[2.4 Required Tools for Building the Java Bindings 7](#_Toc465264668)

[2.5 Optional Tools Supported 8](#_Toc465264669)

[2.6 Optional Tools for GenCP Test 8](#_Toc465264670)

[2.7 Getting the Source Code 8](#_Toc465264671)

[2.8 Building and Testing 9](#_Toc465264672)

[2.9 Building and Testing Step by Step 11](#_Toc465264673)

[2.10 Python Bindings 18](#_Toc465264674)

[2.11 Java Bindings 18](#_Toc465264675)

[2.12 Installing GenICam 18](#_Toc465264676)

[3 Building GenICam on Linux and Mac OS X 19](#_Toc465264677)

[3.1 Getting the sources 19](#_Toc465264678)

[3.2 Minimum Required Tools 19](#_Toc465264679)

[3.3 Creating the packages for distribution 21](#_Toc465264680)

[4 Generating a Release Candidate 21](#_Toc465264681)

[4.1 Building a full GenICam or Software Module Release Candidate 21](#_Toc465264682)

[4.2 Generating a Release Candidate for Non Software Modules 21](#_Toc465264683)

[5 Collaboration Rules 22](#_Toc465264684)

[5.1 Roles and Modules 22](#_Toc465264685)

[5.2 Communication 23](#_Toc465264686)

[5.3 Reporting a Bug 23](#_Toc465264687)

[5.4 Fixing a Bug 24](#_Toc465264688)

[5.5 Maintenance Releases 24](#_Toc465264689)

[5.6 Adding a Feature 25](#_Toc465264690)

[5.7 Versioning Scheme 26](#_Toc465264691)

[5.8 Feature Releases 26](#_Toc465264692)

History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Changed by | Change |
| 1.0 | 09.01.2006 | Fritz Dierks, Basler | First Draft |
| 1.1 | 13.01.2006 | Fritz Dierks, Basler | Added the HTML Workshop |
| 1.2 | 16.01.2006 | Fritz Dierks, Basler | Added DOT and split between run-time and source |
| 1.3 | 28.04.2006 | Fritz Dierks, Basler | Added msxsl.exe as prerequisite |
| 1.4 | 07.05.2006 | Fritz Dierks, Basler | Local Environment; split solutions; xalan instead of msxsl; renamed |
| 1.5 | 15.06.2006 | Christoph Zierl, MVTec  Fritz Dierks, Basler | Added release instructions as decided on the GenICam meeting in Montreal |
| 1.6 | 30.06.2006 | Christoph Zierl, MVTec  Fritz Dierks, Basler | Review and some changes |
| 1.7 | 02.08.2006 | Christoph Zierl, MVTec | Updated required tools (Java SDK, Graphviz) and instruction in Section 3 (~SetGenICamRoot.cmd should be copied, not renamed); corrected cross references. |
| 1.8 | 18.07.2007 | Thomas Köller, Basler | Described Linux build |
| 1.9 | 31.08.2007 | Vincent Rowley, Pleora | Covered other GenICam modules. |
| 1.10 | 17.03.2008 | Natalia Weinstein, Basler | Updated installation instructions (DOT, HTML Help Workshop, PsInfo, Visual Studio additional components).  Updated the settings for running the tests from the GenApi solution.  Updated the names of solutions and files. |
| 1.11 | 24.03.2008 | Fritz Dierks, Basler | Added instructions how to run tests with different schemas |
| 2.0 | 22.09.2009 | Fritz Dierks, Basler | Changed the Build process to CMake |
| 2.1 | 15.01.2010 | Fritz Dierks, Basler | Added description how to switch to a new service pack |
| 2.2 | 09.02.2010 | Fritz Dierks, Basler | Added a comment to use the Release configuration for release builds. |
| 2.3 | 10.02.2010 | Fritz Dierks, Basler | Added VisualStudio 10.0 support |
| 2.4 | 08.03.2010 | Fritz Dierks, Basler | Added Ant |
| 2.5 | 11.02.2010 | Fritz Dierks, Basler | Made msm2msi part of the required toolchain |
| 2.6 | 26.03.2010 | Fritz Dierks, Basler | Made Cmake v2.8.1 mandatory because of OUTPUT\_DIRECTORY |
| 2.7 | 07.04.2010 | Fritz Dierks, Basler | Added security update to VS2005 requirements |
| 2.8 | 29.10.2010 | Vincent Rowley, Pleora | Updated Versioning Scheme section.  Specified that ballot should specify components that are voted on. |
| 2.9 | 01.06.2011 | Mark Jones, The MathWorks | Added Mac OS X |
| 2.10 | 10.06.2011 | Fritz Dierks, Basler | Explained how to build Xerces & Xalanc |
| 2.11 | 27.09.2011 | Thomas Hopfner, MVTec | Added Linux and revised Mac OS X |
| 2.12 | 28.10.2011 | Fritz Dierks, Basler | Added CppCheck as optional tool |
| 2.13 | 13.12.2011 | Stefan Battmer, MATRIX VISION | Added Linux ARM |
| 2.14 | 06.12.2012 | Fritz Dierks, Basler | Added VisualStudio 2012 support |
| 3.0 | 19.01.2014 | Fritz Dierks, Basler | Updated to GenICam 3.0 |
| 3.1 | 20.02.2014 | Stefan Battmer, MATRIX VISION | Increased the min. versions of CMake and Graphviz |
| 3.2 | 07.07.2014 | Hartmut Nebelung, Basler | Updated component list of xsltproc |
| 3.3 | 03.09.2014 | Ryan Robe, National Instruments | Updated the “Building and Testing” section |
| 3.4 | 28.09.2014 | Fritz Dierks, Basler | Removed msm2msi tool |
| 3.5 | 04.01.2015 | Fritz Dierks, Basler | Updated the Windows part of the “Building and Testing” section |
| 3.6 | 06.01.2015 | Fritz Dierks, Basler | Enhance the tool section for xsltproc |
| 3.7 | 24.03.2015 | Stefan Battmer, MATRIX VISION | Fixed a typo, further clarified the xsltproc section |
| 3.8 | 16.10.2015 | Thomas Hopfner, MVTec | Updated Linux and Mac OS X section |
| 3.8.1 | 19.03.2016 | Fritz Dierks, Basler | Fixed a bug |
| 3.9 | 27.10.2016 | Mattias Josefsson, SICK | Added Java bindings |

# Overview

This document describes how to build, test and release GenICam software modules. It also covers the maintenance and release of all GenICam modules.

# Building GenICam on Windows

In order to be able to get the GenICam source code you need to become GenICam member.

Membership is free. The application form comes as part of the GenICam license and can be found [here](http://www.emva.org/cms/upload/Standards/GenICam/GenICam_License_20140921.pdf).

As GenICam member you get access to the Trac/SVN system where the source code lives and you are placed to the GenICam mailing list where you may find support in case of trouble.

## Tools Required for Building the Libraries

* **PowerShell 2.0** or higher which comes pre-installed with **Windows 7** or higher.
* Microsoft Visual C++ VS120 2013, Update 4

This compiler is used as *MainCompiler* for building the official binaries. Make sure to install the service packs.

* Microsoft Visual C++ VS80 2005

This compiler is used as *TestCompiler* in order to ensure that client code using the GenICam libraries can be built with any of the following MSVC versions.

* VS80 2005
* VS90 2008
* VS100 2010
* VS110 2012
* VS120 2013

The build scripts compile the GenICam test code with the oldest complier VC80 and link it against the GenICam libraries built with the newest compiler VC120 libraries. If this is successful it is assumed that all compilers between VC80 and VC120 can also be used. If you have a system where all compilers are installed you can perform a full test if you like. This however takes some time ☺

* **TortoiseSVN** version 1.8.0 or higher. This program is free and can be downloaded from [tortoisesvn.tigris.org](http://tortoisesvn.tigris.org/).
* **CMake** version 3.1.0 or higher. This program is free and can be downloaded from [www.cmake.org](http://www.cmake.org). You need to have this program in the PATH which is an option in the installer dialog.
* **XstlProc** version 1.1.26 or higher. This program is free and can be downloaded from <http://xmlsoft.org/XSLT/xsltproc2.html>.  
    
  The Win32 binaries can be found [here](ftp://ftp.zlatkovic.com/libxml/). Note that you need to download several packages (ZIP files). Copy the content of the contained bin folders in a directory

C:\Program Files (x86)\XsltProc.

and make sure it contains the following files

* **xsltproc.exe** (part of **libxslt-1.1.26.win32.zip** or greater)
* **iconv.dll** (part of **iconv-1.9.2.win32.zip** or greater)
* **libexslt.dll** (part of **libxslt-1.1.26.win32.zip** or greater)
* **libxml2.dll** (part of **libxml2-2.7.8.win32.zip** or greater)
* **libxslt.dll** (part of **libxslt-1.1.26.win32.zip** or greater)
* **zlib1.dll** (part of **zlib-1.2.5.win32.zip** or greater)

This folder has to be in the PATH.

* **7-Zip** version 9.2 or higher. This program is free and can be downloaded from [www.7-zip.org.](http://www.7-zip.org/) You don’t need to have this program in the PATH.

## Tools Required for Building the Documentation

* **Doxygen** version 1.8.9 or higher. This program is free and can be downloaded from [www.stack.nl/~dimitri/doxygen](http://www.stack.nl/~dimitri/doxygen). You don’t need to have this folder in the PATH.
* **Microsoft HTML Help Compiler** version 4.74 or higher. This program is free and can be downloaded from [download.microsoft.com](http://download.microsoft.com/) (search for “Microsoft HTML Help Workshop”). You don’t need to have this program in the PATH.
* **GraphViz** v2.38 or higher. This program is free and can be downloaded from [[www.graphviz.org](http://www.graphviz.org/).](http://www.boostpro.com/download) You don’t need to have this program in the PATH.
* **Highlight** v3.2 or higher. This program is free and can be downloaded from [www.andre-simon.de/](http://www.andre-simon.de/). You don’t need to have this program in the PATH.

## Required Tools for Building the Python Bindings

* **SWIG** version 3.0.7 a tool making the C++ code of GenICam available to Python. It can be downloaded from <http://www.swig.org>. The swig tool has to be on your PATH.
* **Python** version 3.4 Download from <http://www.python.org/download/>  
  Make sure to install also the sphinx tool. After installing Python you can add this tool
* **Sphinx** documentation tool with extensions. Install via PIP  
  >pip install sphinx  
  >pip install sphinxcontrib-plantuml

## Required Tools for Building the Java Bindings

* **SWIG** version 3.0.7 a tool making the C++ code of GenICam available to Java. It can be downloaded from <http://www.swig.org>. The swig tool has to be on your PATH.
* **Java** JDK version 1.8 or higher, either x86 and/or x64 version depending on the desired platform. It can be downloaded from <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

## Optional Tools Supported

* **CppCheck** version 1.68 or higher. This program is free and can be downloaded from [cppcheck.sourceforge.net](http://cppcheck.sourceforge.net)
* **C-Cover** version 8.8.4 or higher. This is a commercial condition/decision measurements tool available from [www.bullseye.com](http://www.bullseye.com) .

## Optional Tools for GenCP Test

* **com0com** version 2.2.2.0 or higher. This program is free and can be downloaded from <http://sourceforge.net/projects/com0com/files/com0com/2.2.2.0/>

After Installation of com0com is this to configure with one bridge between two comports. It is recommended to use a pc-comport in range from com1 to com20 and the camera comport as com21. CLProtocolTest checks and opens comports in range from com1 to com20. In top of GenCPTest.cpp are the comports to update.

* **Python** version 2.7.x This program is free and can be downloaded from <http://www.python.org/download/releases/2.7/>
* **Pyserial** version 2.7 for python 3.x. This program is free and can be downloaded from

<https://pypi.python.org/pypi/pyserial>

Camera-Simulator works now with python 2.7.

## Getting the Source Code

The code is maintained in an SVN repository.

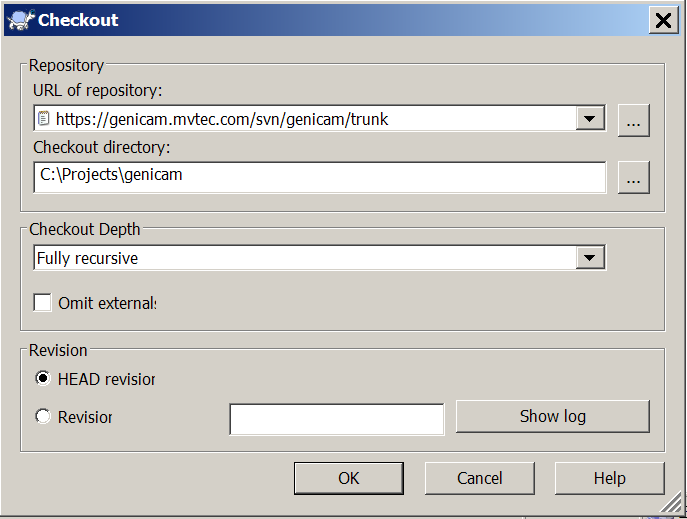
GenICam’s trunk URL is <https://genicam.mvtec.com/svn/genicam/trunk>

However while the v3.0 is still under development you need to use the URL of its development branch which is <https://genicam.mvtec.com/svn/genicam/branches/V3_0>

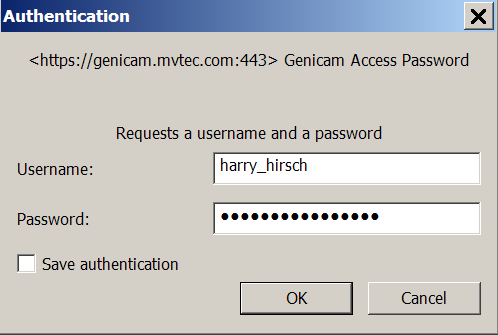
1. Create a folder, e.g. **C:\Projects\genicam**
2. From context menu of that folder choose .



1. Fill in the URL of the trunk and the directory just created as shown below



1. After clicking the ok button you will be asked for authentication. Use your login data for the Trac system which you got when you became GenICam member.



Note: you may need to add your proxy settings to the TortoiseSVN’s settings .This is done in the TortoiseSVN context menu under Settings > Network.

## Building and Testing

Building is performed by PowerShell scripts. You can start PowerShell either via

Start > Programs > Accessories > Windows PowerShell > PowerShell

or use the StartPowerShell.cmd batch file in the root of the GenICam folder.

By default PowerShell will for security reasons run only signed scripts. In order to change this behaviour you need to execute the following command from the shell

PS> Set-ExecutionPolicy RemoteSigned

This will allow local scripts to be run unsigned while any script downloaded from un-save locations such as the internet. Changing the execution policy is permanent so you need to do this only once.

A first quick test can be performed by running

PS> .\ BuildAndTestMinimum.ps1

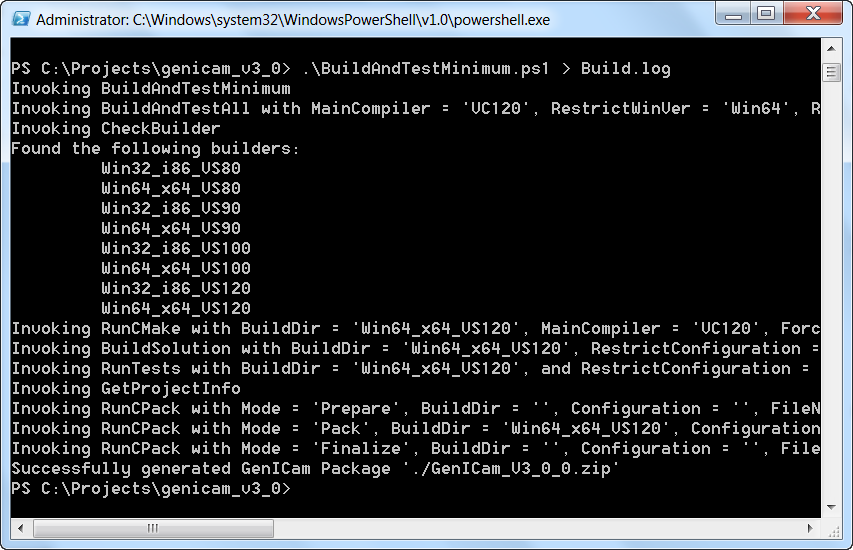
This will build the library with VC120 Win64 Debug and run the test code. If the build process fails for some reason you may want to do the build step by step. How this is done is explained in the next section.

You may want to pipe the build output to a file

PS> .\ BuildAndTestMinimum.ps1 > Build.log

so your screen will not be cluttered with details but look more like this

u



If you don’t have VC120 installed you can use a different compiler, e.g. VC100, by invoking

PS> .\ BuildAndTestMinimum.ps1 VC100

After the first quick test succeeds you should to a full build and test by running

PS> .\ BuildAndTestReasonable.ps1

This will build the library and run the tests with VC120 x {Win32, Win64} x {Debug, Release} and then build and run the tests with VC80 x {Win32, Win64} x {Debug, Release} using the VC120 libraries built in the first step. Note that this may time a while ☺

If the build is successful the installer files can be found in

.\GenICam\_V3\_0\_0.zip

If you are a code maintainer please run BuildAndTestReasonable each time before you check in a code change to SVN.

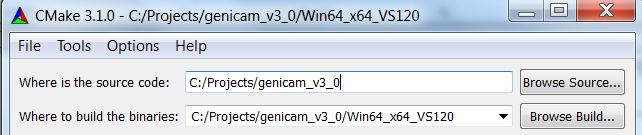
## Building and Testing Step by Step

If you develop code for GenICam or need to debug the build process you may want to perform the building step by step.

1. Start with checking out the code.
2. Start the CMake GUI which can be stared via

Start > Programs > CMake > CMake GUI

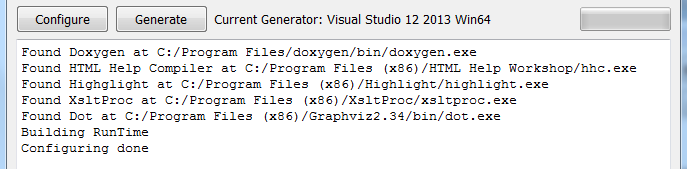
1. Fill in the source code directory and the directory for the binaries as shown below.



The binary directory must be s subdirectory of the source code whose name is composed of the Windows version (Win32 or Win32) the target platform (i86 or x64) and the VisualStudio version (VS80, VS90, VS100, VS110, VS120).

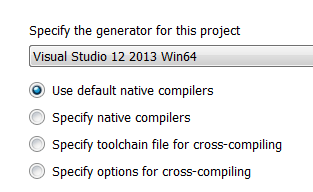
Examples are Win64\_x64\_VS120 or Win32\_i86\_VS80. Note the S (like Studio) in VS80 as compared to C (like Compiler) in VC80.

1. Press the Configure button. This should give you the output shown below.

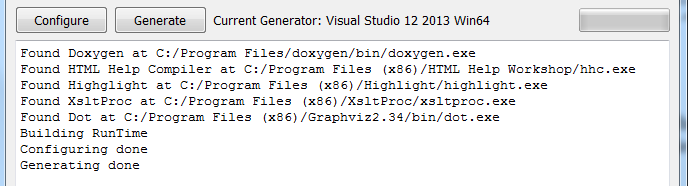


You will get an error if one of the tools is not found. Make sure the tools are present and if so check the files in the ./CMakeFinders directory which describe where the software is searched for.

If the build directory does not yet exist you will be asked for the compiler to use. For Win64\_x64\_VS120 this would be for example



1. Press the Generate button. This should give you the output shown below. Make sure that Visual Studio is not running at this point in time.

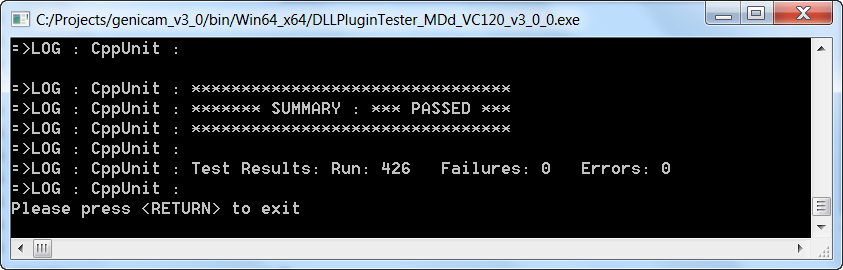


1. Go to the binary directory and open the solution file in Visual Studio by clicking on

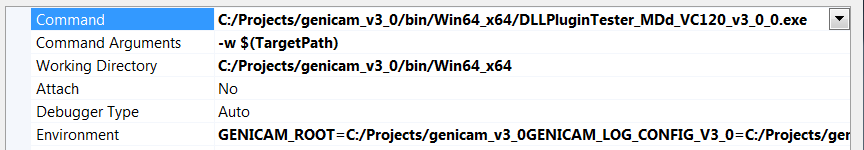
.\Win64\_x64\_VS120\GenICam.sln

Then build the solution.

1. Make GenApiTest the startup project and run the debugger. If all tests run successfully you should get an output like shown below



If the tests don’t start check the GenApiTest’s debug properties and make sure the command, the working directory and the environment is set correctly like shown below.



If that is not the case you most probably had the VisualStudio open when running CMake. When closed VisualStudio stores any changes the user made in the settings in a user specific file named

GenApiTest.vcxproj.GERMANY.fdierks.user

If you start and the file above does not exist VisualStudio taked the content of the file

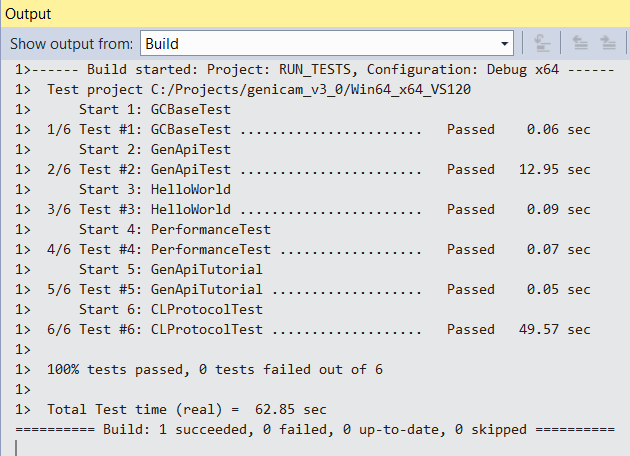
GenApiTest.vcxproj.user

As default. Cmake deletes all user specific files and creates a default file containing the required settings. This however does only make sense if VisualStudio is not open at that time and creates a new user specific file when close which will then prevent VisualStudio from loading the default file at the next start.

1. Compile only the RUN\_TESTS from the context menu

Projects only > Build only RUN\_TESTS

This will run all tests in the solution and should result in an output like shown below.



1. You may use the PowerShell scripts to perform further tests. The key script is

BuildAndTest.ps1

which is used by

BuildAndTestMinimum.ps1

BuildAndTestReasonable.ps1

and in turn uses

RunCMake.ps1

BuildSolution.ps1

RunCPack.ps1

RunCTest.ps1

In order to get help, e.g. for the BuildAndTest script, type

PS> get-help .\BuildAndTest -detailed

resulting in the output shown below.

NAME

C:\Projects\genicam\_v3\_0\BuildAndTest.ps1

ÜBERSICHT

Builds and tests the project for all compilers found;   
 creates install packages

SYNTAX

C:\Projects\genicam\_v3\_0\BuildAndTest.ps1 [[-MainCompiler] <String>]   
 [[-RestrictTestCompiler] <String>] [[-RestrictWinVer] <String>]   
 [[-RestrictConfiguration] <String>] [[-SuppressTests] <Boolean>]  
 [[-Force] <Boolean>] [[-args] <Object>] [<CommonParameters>]

BESCHREIBUNG

Cycles through all VisualStudio Compiler versions found on the system

starting with the MainCompiler

For each compiler the script

- runs CMake

- builds the resulting solutions in Debug as well as in   
 Release configuration

- runs CTest.

At last runs CPack leaving the build artefacts in directory ./packages

The scipt must be run in a directory containing a

CMakeLists.txt file describing a CMake project.

PARAMETER

-MainCompiler <String>

Possible values: VC80, VC90, VC100, VC110, VC120

Default: VC120

GenICam has a MainCompiler version which is used to build   
 the libraries.

Different compiler versions can be used, though they create only the

test prgramms and link to the libraries created with the

MainCompiler. This allows to test for compiler   
 version interoperability

tests. Note that in order for this to work you must always

build with the MainCompiler first

-RestrictTestCompiler <String>

Possible values: VC80, VC90, VC100, VC110, VC120, none, empty

Default: empty

If this parameter is set only the given RestrictTestCompiler is   
 used for

verifying compiler interoperability. If this parameter is not given

(or an empty string) all ViusualStudio compiler versions found   
 on the system

are used for testing (which can take a while)

-RestrictWinVer <String>

Possible values: Win64 Win32 empty

Default: empty

Restricts the whole process to the given windows version

-RestrictConfiguration <String>

Possible values: Debug Release empty

Default: empty

Restricts the whole process to the given configuration.

-SuppressTests <Boolean>

Possible values: $true $false

Default: $false

Supresse the test runs

-Force <Boolean>

Possible values: $true $false

Default: $true

Enforces a rebuild by deleting the BuildDir

-args <Object>

<CommonParameters>

Dieses Cmdlet unterstützt folgende allgemeine Parameter:   
 "Verbose", "Debug",

"ErrorAction", "ErrorVariable", "WarningAction", "WarningVariable",

"OutBuffer" und "OutVariable". Weitere Informationen

erhalten Sie mit dem Befehl "get-help about\_commonparameters".

-------------------------- BEISPIEL 1 --------------------------

C:\PS>BuildAndTestAll

The full blown rebuild of everything. This may take a while :-)

-------------------------- BEISPIEL 2 --------------------------

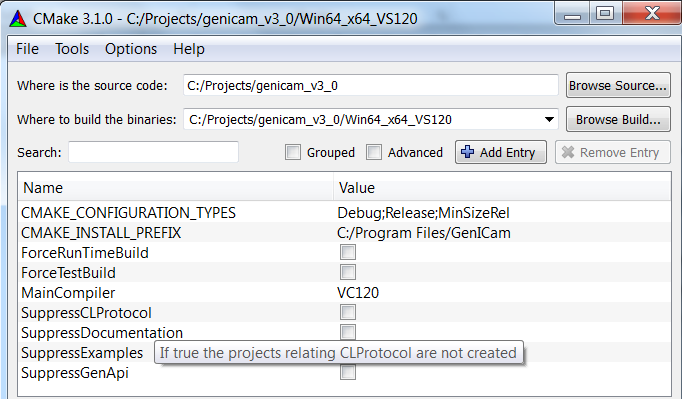
C:\PS>BuildAndTestAll VC120 VC90 Win64 Debug $false

The restricted program: biulds the libraries with VC120, uses VC90 only

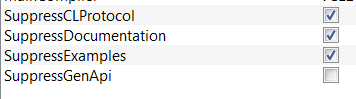
for testing, uses the Debug configuration only and re-uses the

last build

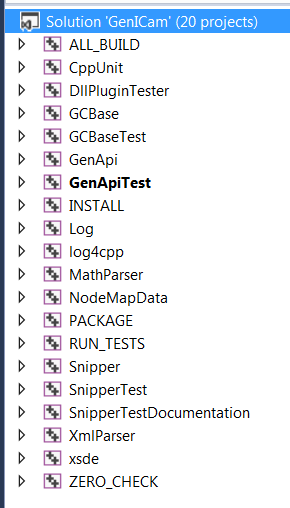
1. If you are a GenICam contributor it is sometimes helpful to create solution files which to not contain all targets but only the targets you are working on. This makes the round trip much faster. You can do this using the variables shown in the CMake GUI (see below).



The tooltip shows help. For example if you want to work on the GenApi module you may want to suppress the creating of the CLProtocol related targets, the Documentation, and the examples which can be performed like this



resulting in a much leander solution like this



If you use a different compiler than the MainCompiler (default is VC120) only the test code will be built. With the following varaibles



you can (temporarily) change the main compiler, enforce a RunTime build even on a TestCompiler of a TestBuild on the MainCompiler.

## Python Bindings

The Python bindings come in a Win32 and a Win64 version. However due to some quirks of the CMake and Python interaction currently there is no cross-compilation possible, i.e. if you build on a 64 bit OS you’ll get only the Win64 version of the bindings and on a 32 bit OS only the Win32 version.

## Java Bindings

Java bindings can be built for both x86 and x64. Make sure the appropriate JDK version is installed for the desired platform.

In order to run the tests for GenTL Java bindings TLSimu must be built. The *TLSimu.cti* and *VirtualFG.dll* needs to be located in the output directory of the bindings. The unit tests are run automatically from CMake when the bindings are built, unless the CMake flag *SuppressTest* is true.

The bindings are built from the main CMake, and are thus located in the main GenICam project, making development of them both easier. They can be suppressed with the CMake flag *SuppressJavaBindings*.

For examples and usage see the test code

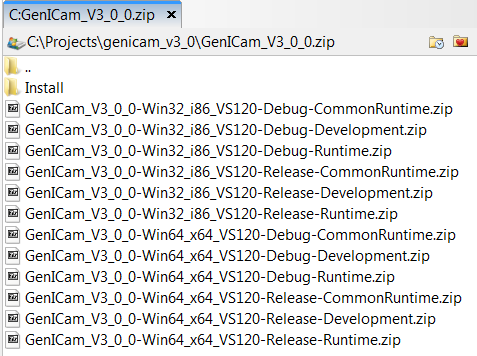
* tests/GenApi/GenApiTestJava
* tests/GenTL/GenTLTestJava

## Installing GenICam

Starting with V3.0 GenICam provides no central installation anymore. Instead you can simply copy the DLLs side by side to the client executable.

In order to make this happen the code has been changed so that no extension of the PATH or any other environment variables are required any more. (TODO: this is not exactly true yet but will for the release)

The user of GenICam is assumed to integrate the GenICam DLLs in his own installer. The necessary files are found in .\GenICam\_V3\_0\_0.zip which is just a zipped version of the .\Packages folder which has the following content:



Each of the contained ZIP files contain parts of the files you may need for GenICam distribution with your own products. The files are tagged by platform (Win32, Win64), by configuration (Release, Debug) and by CMake install group

* RunTime : contains the GenICam DLLs and the license files
* CommonRun : contains the MSVC CRT DLLs
* Development : contains the header file, the implibs and parts of the documentation

For your convenience the content of all the ZIP files is copied together in the install subdirectory of .\GenICam\_V3\_0\_0.zip which gives you the directory tree you would have seen in former versions of GenICam created by the NSIS installer.

# Building GenICam on Linux and Mac OS X

This chapter describes native builds and cross builds of GenICam on other platforms than Windows. Native build means that the compiler toolchain runs on the same platform it builds binaries for. In contrast a cross build generates code which is intended to run on a different platform, usually an embedded system.

Also for some of the steps a GUI can be used you will usually need a command shell like the bash to input the following commands.

## Getting the sources

First you need to download the GenICam sources. These can be found in a Subversion (SVN) repository, so you need Subversion to retrieve the sources. The repository uses the operating systems native line endings for SVN checkout, so you can **not** simply **use the Windows version of the sources**! The different line endings will usually break the shell scripts. Using the Linux/Mac OS X sources on a Windows system is generally ok.

In the repository the available versions are organized by using subdirectories, i.e. below [*https://mvstandards.mvtec.com/svn/genicam/*](https://mvstandards.mvtec.com/svn/genicam/) exist three directories: *branches*, *tags* and *trunk*. If you are not a GenICam developer usually only the *tags* directory is relevant for you as it contains official, released versions. The *trunk* is the current development version which is not guaranteed to work at all and in the *branches* directory you will find various other subdirectories for feature development or maintenance. You can see all the available versions with a command like:

*svn list* [*https://mvstandards.mvtec.com/svn/genicam/tags/*](https://mvstandards.mvtec.com/svn/genicam/tags/)

After this you can check out a copy of a specific version to your local disk by using:

*svn co* [*https://mvstandards.mvtec.com/svn/genicam/tags/V3\_0\_0*](https://mvstandards.mvtec.com/svn/genicam/tags/V3_0_0)

This will create a new directory named V3\_0\_0 in your current working directory.

## Minimum Required Tools

The following tools need to be installed on your computer if you want to get the code from SVN, build it and run the tests:

* **GNU gcc-4.x** or **clang** (OS X) and necessary development tools like binutils**.**
* **Subversion** version 1.5 or higher.
* **CMake** version 3.1.1 or higher.

These tools should be available in the software repositories of your system.

Note: You will need to use a text shell to input the following commands. In the following the 64 bit Linux version is used, please adapt this to your needs. Valid targets are ‘Linux32\_i86’, ‘Linux64\_x64’, ‘Linux32\_ARMhf’, ’Linux32\_ARMsf’ and ‘Maci64\_x64’. The ‘Linux32\_ARMhf’ and ‘Linux32\_ARMsf’ version uses a minimal subset of compiler flags and options that will allow usage of the resulting binaries on architectures starting from ARMv5(thus ARM9, ARM11 and Cortex-A are supported).

Change to the directory which contains the GenICam sources (see above) and then run a script to create a build directory and run CMake to generate the appropriate files and directories:

./RunCMake.sh Linux64\_x64 -f

Now run the build script to generate the GenApi libraries:

./BuildUnix.sh Linux64\_x64

And finally run the tests:

./run\_tests.sh Linux64\_x64

Be prepared to get a very verbose output with lots of “check manually”. The output depends on the setting of the *$GENICAM\_LOG\_CONFIG\_Vx\_y* variable which usually points to *log/config-unix/TestLogging.properties.* Finally you should see something like:

[…]

=>LOG : CppUnit : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=>LOG : CppUnit : \*\*\*\*\*\*\* SUMMARY : \*\*\* PASSED \*\*\*

=>LOG : CppUnit : \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=>LOG : CppUnit :

=>LOG : CppUnit : Test Results: Run: 434 Failures: 0 Errors: 0

=>LOG : CppUnit :

[…]

[5] Generating summary

Summary: PASS

Tested scripts: 4

Failures: 0

## Creating the packages for distribution

Note: You will need to use a text shell to input the following commands. In the following the 64 bit Linux version is used, please adapt this to your needs.

Run the packaging script for your architecture:

./pack\_unix.sh Linux64\_x64

This will create three .tgz files which contain the runtime, the SDK and the tests.

# Generating a Release Candidate

## Building a full GenICam or Software Module Release Candidate

This section describes how to build a formal release candidate for a single or all GenICam software modules.

🡺 **Important**: This procedure is normally only accomplished by the responsible module maintainer (see section 5.1).

1. Add a symbolic tag regarding the RC, i.e. RC1 for the first release candidate.
2. Build GenApi for Windows (VS80) and Linux (gcc40) each for 32 and 64 bit and also for Linux ARM 32 bit and Mac OSX 64 bit. The result will be two installer files for Windows, nine .tgz files for linux(3 for each platform) and three .tgz files for Mac OSX. The Windows installers contain the Runtime, SDK and Tests in one file, for Linux and Mac OSX they are distributed in separate files. For an official release the Windows versions are built by Basler, the Linux32\_i86 and Linux64\_x64 versions are built by MVTec, the Linux32\_ARM version is built by MATRIX VISION and the Mac OSX version is built by MathWorks.
3. Publish the setup files to the GenICam working group for testing and acceptance (for details see section 5).
4. If the release candidate is accepted by the GenICam working group perform the following final steps:
   1. Add the final tag to the repository.
   2. Publish the result as described in section 5.
5. If the release candidate is not accepted, fix the problem and build a new release candidate repeating the whole procedure using the label and tag RCx where x is incremented for each release candidate by one.

## Generating a Release Candidate for Non Software Modules

Release candidates for non software GenICam modules, such as the SFNC module, are simply generated by tagging the appropriate files in the SVN repository.

🡺 **Important**: This procedure is normally only accomplished by the responsible module maintainer (see section 5.1).

For instance, the procedure to create a release candidate for the SFNC module is:

1. Add a symbolic tag regarding the RC, i.e. RC1 for the first release candidate.
2. Zip the SFNC Microsoft Word document to a file named GenICam Standard Features Naming Convention 1.5.1\_RC1.zip .
3. Publish the zip file to the GenICam working group for acceptance (for details see section 5).
4. If the release candidate is accepted by the GenICam working group perform the following final steps:
5. Add the final tag to the SVN
6. Publish the results as described in section 5.
7. If the release candidate is not accepted fix the problem and generate a new release candidate repeating the whole procedure using the label and tag RCx where x is incremented for each release candidate by one.
8. When the SFNC release candidate is accepted, follow the detailed procedure described in the “SFNC release checklist and documents location.docx” document that is located in the “\*SVN GenICam*\doc\Standard\_Feature\_List” repository.

# Collaboration Rules

## Roles and Modules

This document refers to several **roles** which are defined in this section

The **GenICam working group** consists of all associated and contributing members of the GenICam standard committee as listed on www.genicam.org.

The **maintainer** of a GenICam **module** is responsible for the integrity of this component. Normally only the maintainer may change the component. Any proposed changes (bug fixes, added features, other) must be sent to the maintainer who will include them in the component.

The following GenICam modules are currently set up:

* **GenApi Software Module** – reference implementation of code which is responsible for configuring a device
* **SFNC Module** – recommended names and types for common features
* **GenTL Software Module** – code responsible for transferring data between a device and a host
* **Validator Software Module** – code checking a camera configuration file for standard compliance
* **SVN Admin Module** – the data base holding the GenICam source code and documents, see https://genicam.mvtec.com/svn/genicam/
* **Trac Admin Module** – the wiki and ticket tracking system, see https://genicam.mvtec.com/
* **MailingList Admin Module** – the central communication tool for the GenICam working group
* **WebSite Admin Module** – the GenICam web site on http://[www.genicam.org](http://www.genicam.org)
* **FTP** **Admin Module** – the GenICam FTP server on ftp://[ftp.genicam.org](ftp://ftp.genicam.org) (tbd => create)
* **Releasing Admin Module –** preparing the official release version

A list of modules and their current maintainers can be found on XXX (tbd => SVN)

## Communication

Communication is accomplished via email. The following email addresses and aliases exist:

* [GenICam@imaging.de](mailto:GenICam@imaging.de) - the GenICam mailing list
* [GenApi@genicam.org](mailto:GenApi@genicam.org) - an email alias for the GenApi module’s maintainer and his/her deputies.
* [GenTL@genicam.org](mailto:GenTL@genicam.org) - an email alias for the GenTL module’s maintainer and his/her deputies.
* [Validator@genicam.org](mailto:Validator@genicam.org) - an email alias for the Validator module’s maintainer and his/her deputies.
* [CVS@genicam.org](mailto:CVS@genicam.org) - an email alias for the CVS maintainer and his/her deputies.
* [Mantis@genicam.org](mailto:Mantis@genicam.org) - an email alias for the Mantis maintainer and his/her deputies.
* [MailingList@genicam.org](mailto:MailingList@genicam.org) - an email alias for the mailing list maintainer and his/her deputies.
* [WebSite@genicam.org](mailto:WebSite@genicam.org) - an email alias for the web site maintainer and his/her deputies.
* [FTP@genicam.org](mailto:FTP@genicam.org) - an email alias for the FTP server maintainer and his/her deputies.
* [Releasing@genicam.org](mailto:Releasing@genicam.org) - an email alias for the FTP server maintainer and his/her deputies.
* [GenICam-contact@<Mycompany>.com](mailto:GenICam-contact@%3cMycompany%3e.com) - a local email alias of each company giving access to the GenICam related people at this company. All contributing members of the GenICam working group should provide this email alias; associated members might provide it, too.

The GenICam web site is maintained at http://[www.genicam.org](http://www.genicam.org)

A FTP site is available under ftp://[ftp.genicam.org](ftp://ftp.genicam.org)

During discussions on the mailing list the subject in the email header should not be changed in order to allow grouping the mails by thread.

## Reporting a Bug

Bugs are reported through the Trac ticket system and a copy is sent on the GenICam mailing list.

Any bug report must come with a description and a CPPUnit test case, when applicable, which reproduces the bug. The H/CPP files establishing the test case must be appended to the bug report.

The bug reporter is encouraged to provide a bug fix together with the bug report.

By choosing the correct component the bug report (=ticket) is automatically assigned to the module’s maintainer which should try to reproduce the problem within 1 week.

## Fixing a Bug

The following steps should be followed for fixing a bug in GenICam:

1. If the bug reporter submits a bug fix the maintainer of the affected module will review, possibly adapt and include the fix into the SVN and create an entry in the release notes. The maintainer may reject the implementation if it breaks the module integrity or architecture in major ways or does not follow the implementation rules such as the coding policy (see $(GENICAM\_ROOT)/doc/Coding\_Policy.doc.) in the case of a software module.
2. In case no bug fix is provided the maintainer will attempt to fix the bug in reasonable time himself.
3. The bug should be fixed in the trunk as well as all affected branches of the SVN.
4. After the fix is added to the SVN the maintainer sends out an email to the mailing list with subject “<ModuleName> BUG\_FIXED : < bug name>” including a copy of the release notes. ModuleName is the name of the GenICam module in which a bug was fixed.
5. The members of the working group then get a fresh copy from the affected SVN branch(es) and review the bug fix. They have normally 2 weeks to object against the changes.
6. If no one objects within 2 weeks the maintainer sends out an email with subject “<ModuleName> BUG\_FIX\_ACCEPTED : < bug name>” to indicate the end of the process.

## Maintenance Releases

1. A maintenance release can be requested by any member of the working group if he requires the current bug fixes to be published. The request is sent to one of the maintainers with a Cc on the mailing list and subject “<ProjectName> MAINTENANCE\_RELEASE\_REQUEST” where ProjectName is GenICam or the name of the GenICam module for which the maintenance release is requested.
2. The maintainer creates usually within 1 week a release candidate as described in section 4.1 if the maintenance involves a software module. Otherwise, a release candidate is generated as per Section 4.2.
3. He puts the release candidate on the FTP server and sends out an email with subject “<ProjectName> MAINTENANCE\_RELEASE\_CANDIDATE\_AVAILABLE”
4. The contributing members of the working group then review the release candidate. They have normally 2 weeks for accepting it.
5. If no one objects within 2 weeks the maintainer sends out an email with subject “<ProjectName> MAINTENANCE\_RELEASE\_CANDIDATE\_ACCEPTED” to indicate the end of the process.
6. The accepted release candidate is made a release and published on the GenICam web site.
7. In urgent cases the requester of the maintenance request can create a private build and provide it to customers affected by the bug. The private build must be clearly treated as such and made accessible with source code on the FTP server. The private build must only be used until the maintenance release is available. It is strongly recommended to replace the private build with the maintenance release at the affected customers.

## Adding a Feature

Features are added to the trunk only.

The following steps should be followed for adding a new feature to any GenICam module:

1. The requester should send an email with subject “<ModuleName> FEATURE\_REQUEST : <feature name>” with a brief explanation of the feature he wants to add, how it should be done, why, and which modules would be affected. ModuleName is the name of the GenICam module for which a new feature is requested.
2. The feature request should be discussed on the mailing list and typically either accepted or rejected by the contributing members of the last meeting within 1 week. The maintainer of the module can decide to extend the discussion period to more than one week based on the nature of the feature to be added.
3. If the community agrees that the feature should be added the requester should implement the feature on a local copy of the latest version on the trunk of the GenICam SVN repository. The implementation must contain test cases achieving 100% function and condition/decision coverage, when applicable, as well as complete relevant documentation.
4. The requester should send the implemented feature to the maintainer(s) of the affected module(s) which will review, possibly adapt and commit the feature into the trunk of the SVN and create an entry in the release notes. The maintainer may reject the implementation if it breaks the module integrity or architecture in major ways or does not follow the implementation rules such as the coding policy (see $(GENICAM\_ROOT)/doc/Coding\_Policy.doc) for software modules.
5. After the new feature is added to the SVN the maintainer sends out an email to the mailing list with subject “<ModuleName> FEATURE\_ADDED : <feature name>” including a copy of the release notes.
6. The members of the working group then get a fresh copy from the SVN and review the new feature. They have normally 2 weeks for objecting against the changes.
7. If no one objects within 2 weeks the maintainer sends out an email with subject “<ModuleNane> FEATURE\_ACCEPTED : <feature name>” to indicate the end of the process.

## Versioning Scheme

The versioning of a GenICam module follows the following scheme:

<Major>.<Minor>.<Sub-Minor>

An example would be “1.1.2”.

The Major number is increased when there are significant changes in functionality, the Minor number is incremented when only minor features have been added and the Sub-Minor number is incremented when minor issues are fixed.

Therefore, a major release may introduce a break in backward compatibility but it is not necessarily the case. For instance, the Major number can be increased to indicate the achievement of a significant milestone that does not necessarily introduce a break in backward compatibility. On the other hand, a minor or maintenance release must maintain backward compatibility. If the module is a software module, or includes one, then binary compatibility must be maintained for a maintenance release. On the other hand, source compatibility must at least be maintained for the minor release of a module including a software one.

## Feature Releases

A feature release is run along the following steps:

1. A feature release can be requested by any member of the working group. The request is sent to one of the maintainers with a cc on the mailing list and subject “<ProjectName> FEATURE \_RELEASE\_REQUEST” which a brief explanation why a feature release would be necessary. ProjectName is GenICam or the name of the GenICam module for which the feature release is requested.
2. The feature release request should be discussed on the mailing list and typically either accepted or rejected within 2 weeks.
3. If the community agrees the maintainer creates a release candidate as described in section 4.1 if the maintenance involves a software module. Otherwise, a release candidate is generated as per Section 4.2..
4. He puts the release candidate on the FTP server and sends out an email with subject “<ProjectName> RELEASE\_CANDIDATE\_AVAILABLE”
5. The contributing members of the working group then review the release candidate. They have normally 2 weeks for accepting it.
6. If no one objects within 2 weeks the maintainer sends out an email with subject “<ProjectName> RELEASE\_CANDIDATE\_ACCEPTED” to indicate the end of the process.
7. If the release candidate is accepted technically it must be formally voted for by the contributing members. The ballot must clearly describe the module(s) and version number(s) that are voted on. When voting on a GenICam release, the version of the release must be specified as well as the related sub-modules and their versions. Normally this voting should be take place during a meeting where the release is done. However a phone conference is also acceptable which should be organized by the maintainer.
8. If the release is formally accepted it is made available on the GenICam web site.