Building Google Test and Google Mock for native C++

This guide describe how to build Google Test and Google Mock libraries to use it for C++ native code unit tests.

Google's framework for writing C++ tests on a variety of platforms (Linux, Mac OS X, Windows, Cygwin, Windows CE, and Symbian). Based on the xUnit architecture. Supports automatic test discovery, a rich set of assertions, user-defined assertions, death tests, fatal and non-fatal failures, value- and type-parameterized tests, various options for running the tests, and XML test report generation.

Getting StartedAfter downloading Google Test, unpack it, read the README file and the documentation wiki pages (listed on the right side of this front page). Who Is Using Google Test?In addition to many internal projects at Google, Google Test is also used by the following notable projects: The Chromium projects (behind the Chrome browser and Chrome OS) The LLVM compiler Protocol Buffers (Google's data interchange format)

If you know of a project that's using Google Test and want it to be listed here, please let googletestframework@googlegroups.com know. Google Test-related open source projectsGoogle Test UI is test runner that runs your test binary, allows you to track its progress via a progress bar, and displays a list of test failures. Clicking on one shows failure text. Google Test UI is written in C#. GTest TAP Listener is an event listener for Google Test that implements the TAP protocol for test result output. If your test runner understands TAP, you may find it useful.

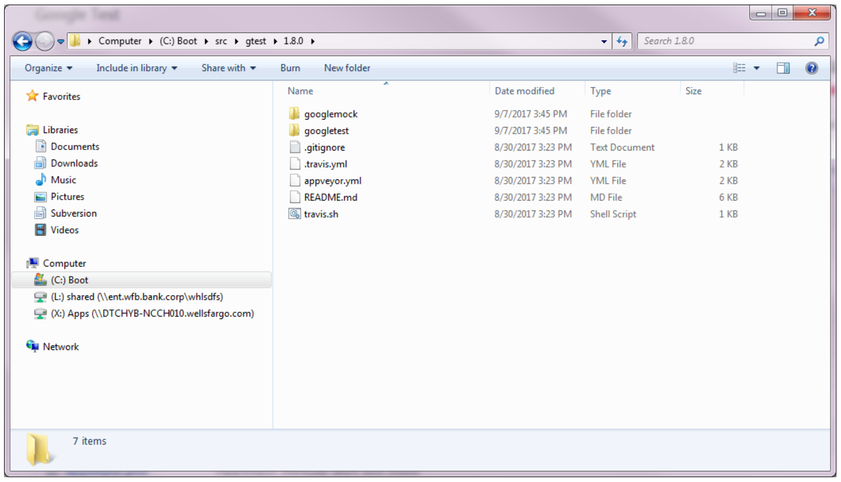
Step-by-step guide for Windows 7 64bit

Download the source code from

https://cppra00a0114.wellsfargo.com:8112/oer/index.jsp (09/07/2017 version 1.8.0 is available)

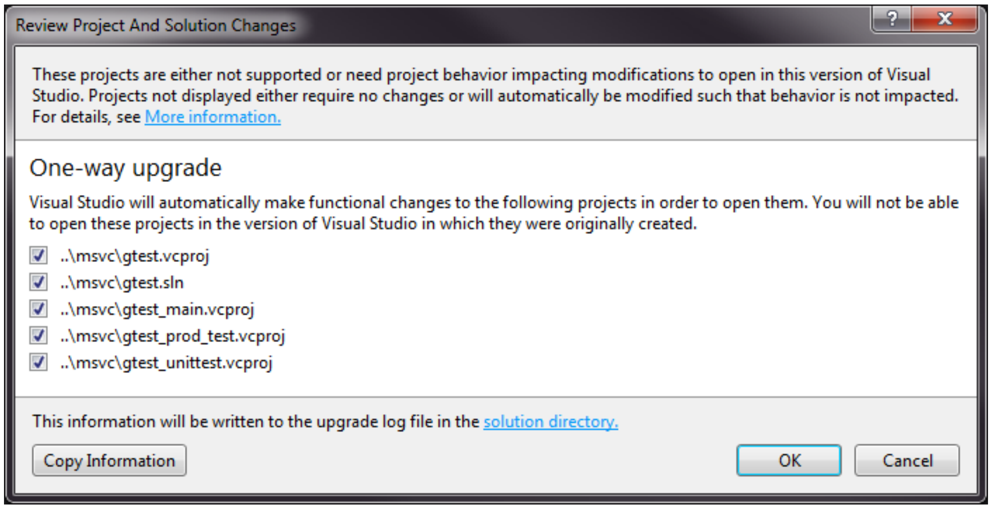
or from original source https://github.com/google/googletest/ in local directory (e.g. C:\src\gtest\)

Place the file googletest-release-1.8.0.zip into the folder c:\src\gtest and unpack zip archive. Delete the .zip file. After that rename the directory googletest-release-1.8.0 to 1.8.0. Such a source tree should be on the local computer after all of these manipulations.



Name	Date modified	Type	Size
googlemock	9/7/2017 3:45 PM	File folder	
googletest	9/7/2017 3:45 PM	File folder	
.gitignore	8/30/2017 3:23 PM	Text Document	1 KB
.travis.yml	8/30/2017 3:23 PM	YML File	2 KB
appveyor.yml	8/30/2017 3:23 PM	YML File	2 KB
README.md	8/30/2017 3:23 PM	MD File	6 KB
travis.sh	8/30/2017 3:23 PM	Shell Script	1 KB

Go to the C:\src\gtest\1.8.0\googletest\msvc, right click on the gtest.sln and Open with Visual studio 2015. VS2015/VS2012 asks to one-way upgrade - press OK.



Review Project And Solution Changes

These projects are either not supported or need project behavior impacting modifications to open in this version of Visual Studio. Projects not displayed either require no changes or will automatically be modified such that behavior is not impacted. For details, see More information.

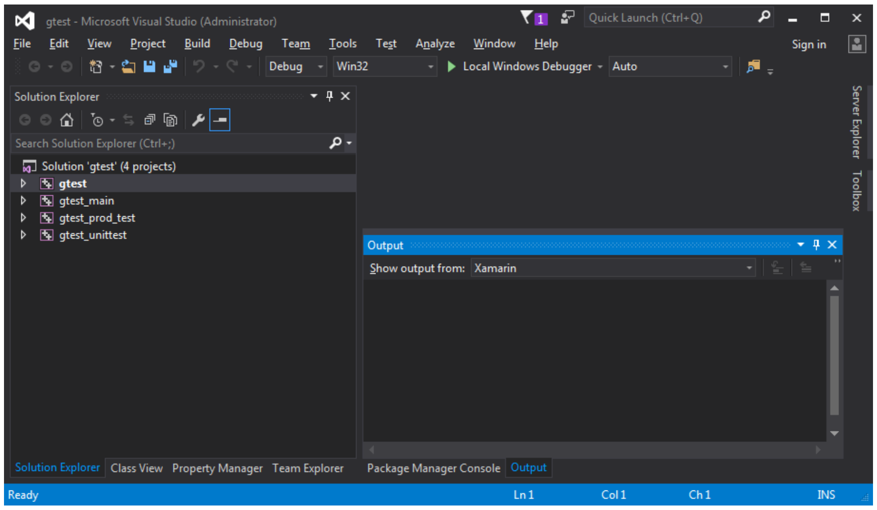
One-way upgrade

Visual Studio will automatically make functional changes to the following projects in order to open them. You will not be able to open these projects in the version of Visual Studio in which they were originally created.

- [✓] ..\msvc\gtest.vcproj
- [✓] ..\msvc\gtest.sln
- [✓] ..\msvc\gtest_main.vcproj
- [✓] ..\msvc\gtest_prod_test.vcproj
- [✓] ..\msvc\gtest_unittest.vcproj

This information will be written to the upgrade log file in the solution directory.

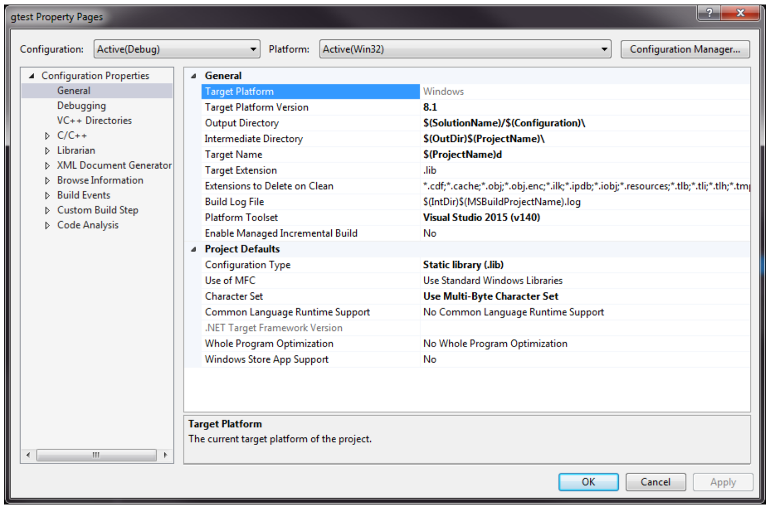
This is a source tree in the Visual Studio 2015 IDE



Right click on the gtest and choose Build. On the VS2015 it should be built ok but two warnings are issued

```
1>------ Build started: Project: gtest, Configuration: Debug Win32 ------

1>  gtest-all.cc

1>C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V140\Microsoft.CppBuild.targets(1357,5): warning
MSB8012: TargetPath(C:\src\gtest\1.8.0\googletest\msvc\gtest/Debug\gtest.lib) does not match the Library's
OutputFile property value (C:\src\gtest\1.8.0\googletest\msvc\gtest\Debug\gtestd.lib). This may cause your
project to build incorrectly. To correct this, please make sure that $(OutDir), $(TargetName) and
$(TargetExt) property values match the value specified in %(Lib.OutputFile).

1>C:\Program Files (x86)\MSBuild\Microsoft.Cpp\v4.0\V140\Microsoft.CppBuild.targets(1359,5): warning
MSB8012: TargetName(gtest) does not match the Library's OutputFile property value (gtestd). This may cause
your project to build incorrectly. To correct this, please make sure that $(OutDir), $(TargetName) and
$(TargetExt) property values match the value specified in %(Lib.OutputFile).

1>  gtest.vcxproj -> C:\src\gtest\1.8.0\googletest\msvc\gtest/Debug\gtest.lib

========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

To fix it right click on the gtest and choose Properties and add **d** symbol to the General->Target Name


gtest Property Pages

Configuration: Active(Debug)   Platform: Active(Win32)

General	
Target Platform	Windows
Target Platform Version	**8.1**
Output Directory	**$(SolutionName)/$(Configuration)**
Intermediate Directory	**$(OutDir)$(ProjectName)**
Target Name	**$(ProjectName)d**
Target Extension	.lib
Extensions to Delete on Clean	*.cdf;*.cache;*.obj;*.obj.enc;*.ilk;*.ipdb;*.iobj;*.resources;*.tlb;*.tli;*.tlh;*.tm
Build Log File	$(IntDir)$(MSBuildProjectName).log
Platform Toolset	**Visual Studio 2015 (v140)**
Enable Managed Incremental Build	No
Project Defaults	
Configuration Type	**Static library (.lib)**
Use of MFC	Use Standard Windows Libraries
Character Set	**Use Multi-Byte Character Set**
Common Language Runtime Support	No Common Language Runtime Support
.NET Target Framework Version	
Whole Program Optimization	No Whole Program Optimization
Windows Store App Support	No

Target Platform
The current target platform of the project.

After that rebuild the library and no warnings should be issued.

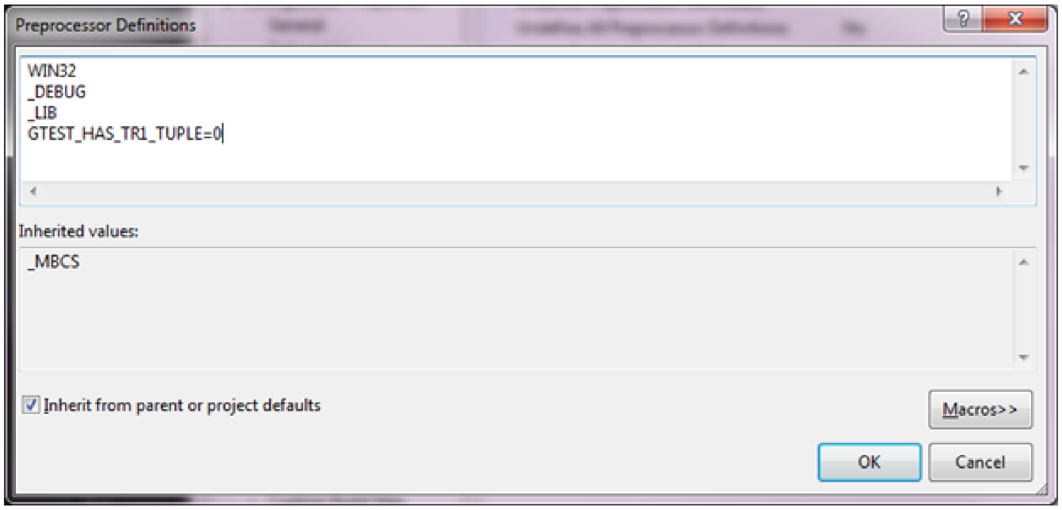
As a target you get C:\src\gtest\1.8.0\googletest\msvc\gtest/Debug\gtestd.lib static library.

NOTE for VS2012

If you use VS2012 you should add GTEST_HAS_TR1_TUPLE=0 definition to all of these 4 projects

- gtest,
- gtest_main,
- gtest_prod_test,
- gtest_unittest

In properties of the each project go to the C++->Preprocesor->Preprocessor definitions


Preprocessor Definitions

```
WIN32
_DEBUG
_LIB
GTEST_HAS_TR1_TUPLE=0
```

Inherited values:

```
_MBCS
```

☑ Inherit from parent or project defaults

FYI: Choosing a TR1 Tuple Library

Some Google Test features require the C++ Technical Report 1 (TR1) tuple library, which is not yet available with all compilers. The good news is that Google Test implements a subset of TR1 tuple that's enough for its own need, and will automatically use this when the compiler doesn't provide TR1 tuple.

Usually you don't need to care about which tuple library Google Test uses. However, if your project already uses TR1 tuple, you need to tell Google Test to use the same TR1 tuple library the rest of your project uses, or the two tuple implementations will clash. To do that, add

-DGTEST_USE_OWN_TR1_TUPLE=0

to the compiler flags while compiling Google Test and your tests. If you want to force Google Test to use its own tuple library, just add

-DGTEST_USE_OWN_TR1_TUPLE=1

> *to the compiler flags instead.*
>
> *If you don't want Google Test to use tuple at all, add*
>
> *-DGTEST_HAS_TR1_TUPLE=0 (set this for the compilation for VS2012)*
>
> *and all features using tuple will be disabled.*
>
> *See*
>
> *https://github.com/appveyor/ci/issues/742*
>
> *https://msdn.microsoft.com/en-us/library/bdscwf1c.aspx*

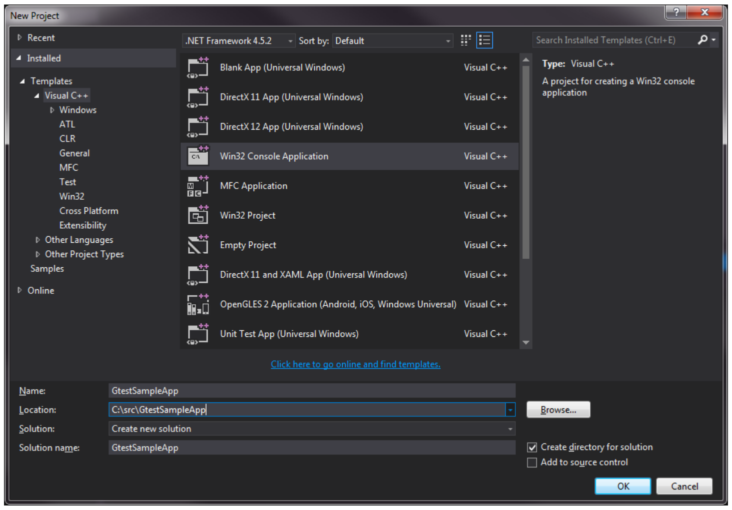
Also in C/C++ ->Code generation->Runtime library set /MTd option

Create test application

Now let's create test application

In VS2015\VS2012 File->New->Project->Win32 Console Application

Name: GtestSampleApp


New Project
Recent
Installed
Templates
Visual C++
Windows
ATL
CLR
General
MFC
Test
Win32
Cross Platform
Extensibility
Other Languages
Other Project Types
Samples
Online
.NET Framework 4.5.2
Sort by: Default
Search Installed Templates (Ctrl+E)
Blank App (Universal Windows) Visual C++
DirectX 11 App (Universal Windows) Visual C++
DirectX 12 App (Universal Windows) Visual C++
Win32 Console Application Visual C++
MFC Application Visual C++
Win32 Project Visual C++
Empty Project Visual C++
DirectX 11 and XAML App (Universal Windows) Visual C++
OpenGLES 2 Application (Android, iOS, Windows Universal) Visual C++
Unit Test App (Universal Windows) Visual C++
Click here to go online and find templates.
Browse...
OK
Cancel

Type: Visual C++

A project for creating a Win32 console application

Name: GtestSampleApp

Location: C:\src\GtestSampleApp

Solution: Create new solution

Solution name: GtestSampleApp

[✓] Create directory for solution

[ ] Add to source control

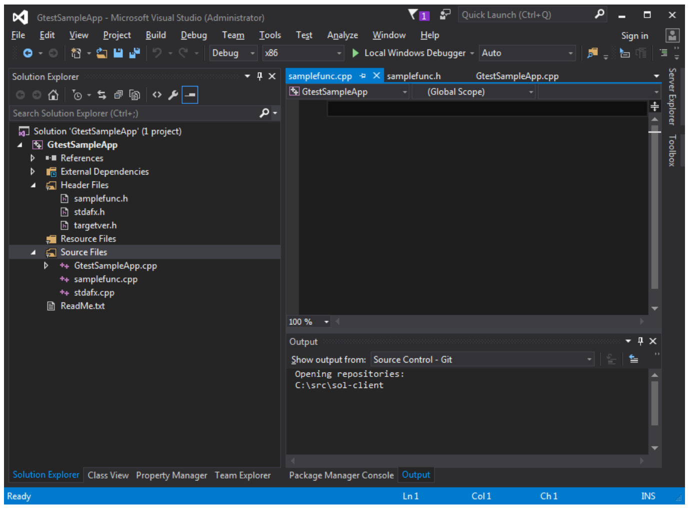
Click Next, remove the mark from Security Development Lifecycle (SDL) checks option, click Finish

Let's create simplest function to test - make separate files to emulate larger project.

Click on the subfolder Header Files inside GtestSampleApp solution and right click. Choose Add->New Item and in new dialog box Header File(,h). Change name of the file in the Name: field to samplefunc.h.

After add the same way a file samplefunc.cpp to the Source Files subfolder.

The source tree looks now:


GtestSampleApp - Microsoft Visual Studio (Administrator)
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Sign in
Debug x86 Local Windows Debugger Auto
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'GtestSampleApp' (1 project)
GtestSampleApp
References
External Dependencies
Header Files
samplefunc.h
stdafx.h
targetver.h
Resource Files
Source Files
GtestSampleApp.cpp
samplefunc.cpp
stdafx.cpp
ReadMe.txt
samplefunc.cpp samplefunc.h GtestSampleApp.cpp
GtestSampleApp (Global Scope)
100 %
Output
Show output from: Source Control - Git
Solution Explorer Class View Property Manager Team Explorer
Package Manager Console Output
Ready Ln 1 Col 1 Ch 1 INS
Server Explorer Toolbox

```
Opening repositories:
C:\src\sol-client
```

The source of the samplefunc.h

```
#ifndef __SAMPLEFUNC_H__

#define __SAMPLEFUNC_H__

int multiplyByTwo(int k);

#endif

// #ifndef __SAMPLEFUNC_H__
```

The source of the samplefunc.cpp

```
#include "samplefunc.h"

int multiplyByTwo(int k)

{

return k * 2;

}
```

Right on the GtestSampleApp and go to the Properties

Then create two more files for the create tests. Despite the fact all of these things may be placed to one file with main functions I try to show the important concept of the Google Tests - tests may (and should.. and must) be undependable from main code.

So create files tests.h and tests.cpp respectively.

Related articles

- Building Google Test and Google Mock for native C++