This article details the versions of Visual Studio required to build certain components of CRYENGINE for Windows.
Note that Visual Studio integrations for other platforms are not considered in this article, but they may limit your choices with regards to Visual Studio versions to use.

3rd party installation

Visual Studio versions supported

For CRYENGINE versions before 3.6.2, only Visual Studio 2010 Professional is supported (support for Express edition in FreeSDK only)
For CRYENGINE versions before 3.8.4, only Visual Studio 2012 Professional is supported (support for Express edition in EaaS only)
For CRYENGINE versions starting with 3.8.4, please check the following table for support:

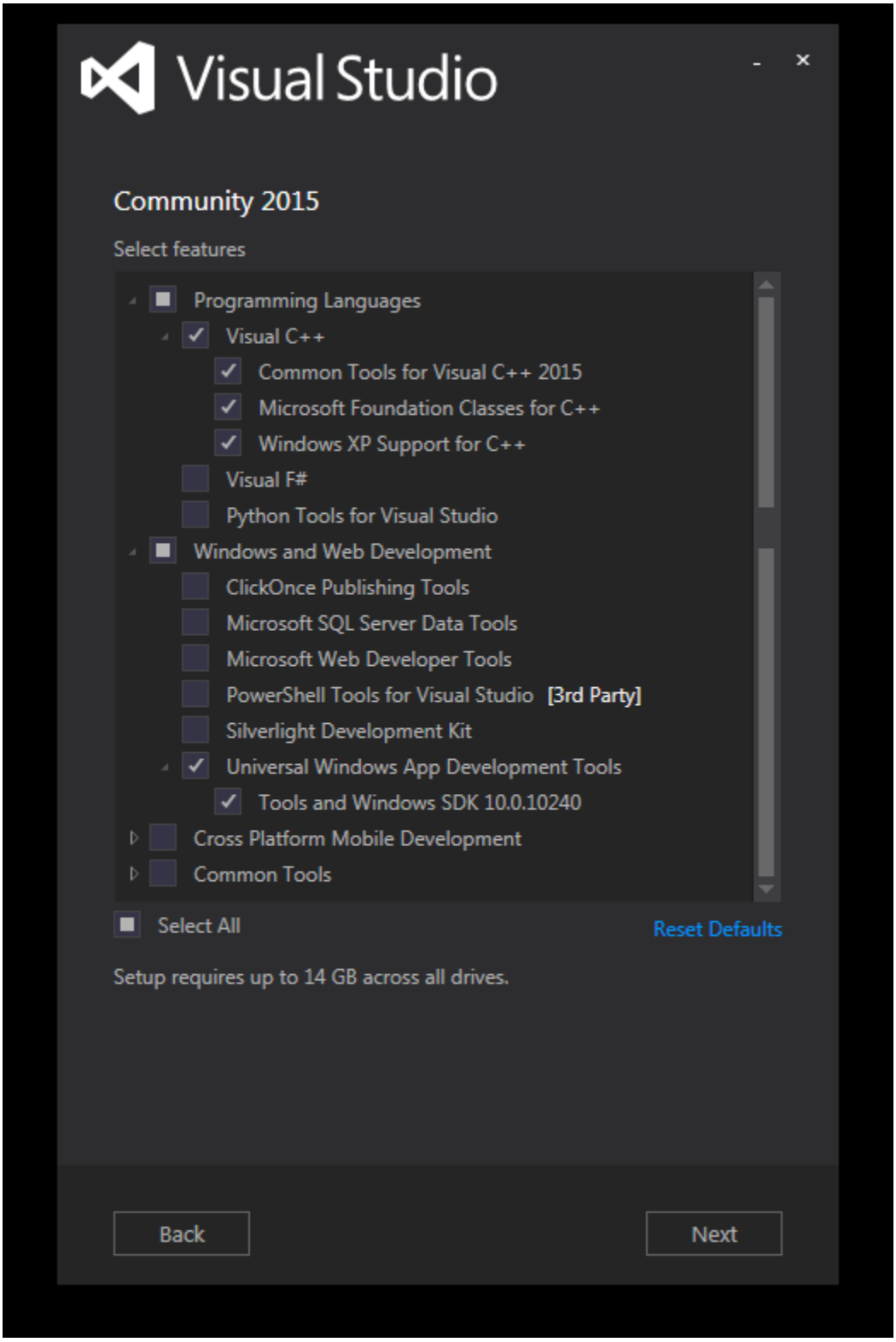
Like Sandbox only runs on 64-bit Windows, we support development only on 64-bit Windows OS.
This means that we expect to use the 64-bit hosted tool-chains for Visual C++.
To clarify: 32-bit Windows is supported as a target for the launcher (and for running CRYENGINE games on), but not as a host for compilation.
Please note that for Visual Studio SKU's of the type Ultimate and/or Enterprise are treated as Professional for this table.

CRYENGINE Component	2012 Express	2012 Professional	2013 Express	2013 Community	2013 Professional	2015 Express	2015 Community	2015 Professional
Game-code (EaaS)	✓	✓	✓	✓	✓	✓	✓	✓
Engine-code (Licensed)	✓	✓	✓	✓	✓	✓	✓	✓
Sandbox Plugins Tools	✗	✓	✗	✓	✓	✗	⚠	⚠

The items marked with ⚠ are partially supported, some plugins or features may not be available when using this version of Visual Studio, because 3rdParty dependencies may not have made available libraries compatible with this Visual studio version.
Please refer to the release notes of the particular version of CRYENGINE for details on limitations for specific versions of Visual Studio.
For 3.8.4: Important 3.8.4 Code and Data Changes

Starting with Visual Studio 2015, Visual C++ is no longer installed with Visual Studio by default.
Visual C++ and Windows SDK are required components for all CRYENGINE compilation options, so make sure to always select these features when installing Visual Studio 2015.

Here is an example of a working installation for CRYENGINE:



Windows SDK versions supported

The following Windows SDK versions are supported for each supported compiler. It's strongly recommended to use these combinations, as these are the officially supported scenario's by Microsoft, and mixing differently may lead to problems.

Visual Studio	Visual C++ version	Recommended Windows SDK	Notes
2012	11.0	8.0	
2013	12.0	8.1	
2015	14.0	10.0.10240.0	Early installations of Visual Studio 2015 may have installed 10.0.10150.0 instead. In this scenario, re-run the Visual Studio installer and select the recommended version.

Note for Visual Studio Express editions

Please note that when using Visual Studio Express editions to compile CRYENGINE code, you need to have a Windows SDK installed.
This means that you should pick the "for Desktop" variants when downloading Visual Studio.

Note for Visual Studio Community editions

Please note that when using Visual Studio Community editions to compile CRYENGINE code, you need to have a Windows SDK installed.
Usually, this can be installed or added afterwards when setting optional packages in the Visual Studio installer.

Note regarding redistributable libraries

This item only applies to WAF starting with CRYENGINE 3.8.4.
For previous versions, you may want to copy the dependencies manually.

When WAF builds certain projects, it will attempt to copy redistributable DLLs from either Windows SDK or from Visual Studio installation folders. These steps are optional (the build will not fail if the files are not present), but it's strongly recommended to include them for each of testing and automatic dependency selection.

Please consult the following table, to see where WAF will attempt to locate redistributable files.
In general, if a file is missing, a warning will be generated. In this case, consider changing/upgrading your Visual Studio or Windows SDK installation.

Package	Source Location	Files	Notes
Visual C++ runtime	Visual C++ installation folder	msvcrXXX.dll msvcpXXX.dll vcruntimeXXXX.dll concrtXXX.dll	CRT redist is optionally installed with Visual Studio
MFC runtime	Visual C++ installation folder	mfcXXX.dll mfcmXXX.dll mfcXXXenu.dll	MFC redist is optionally installed with Visual Studio Only required when building Sandbox, some plugins and some tools
D3D compiler	Windows SDK installation folder	d3dcompiler_XX.dll	D3D redist is installed with Windows SDK
dbghelp	Windows SDK debuggers folder	dbghelp.dll	Debuggers feature is optionally installed with Windows SDK

WAF Configuration

The remainder of this chapter applies to CRYENGINE version 3.8.4 and newer.

The build system distributed with CRYENGINE is WAF, and it supports the above configurations. There are configuration options to control which version of Visual C++ compiler and/or Windows SDK is used for compilation, if the defaults do not work for your project. For the remainder of this chapter, please note that we will use "MSVC" to indicate a Visual C++ compiler installation, identified by a version number as shown in the table above in the column "Visual C++ version" (specifically, we don't refer to the Visual Studio version number, such as 2012 or 2015).

Configuration options

The options that control the WAF compiler selection are stored in the _WAF_/user_settings.json file, and can be set using the WAF options dialog.
All of these options can be found in the "Compiler Detection" tab.

Option	Command-line parameter	Default value	Notes
auto_detect_compiler	--auto-detect-compiler	True	This should be set to True when using MSVC configuration.
auto_detect_verbose	--auto-detect-verbose	False	This option can be set to True to troubleshoot problems with MSVC configuration.
minimum_msvc_compiler	--minimum-msvc-compiler	11	The minimum MSVC version to support (interpreted as floating point expression). This controls the minimum version required to compile the project.
force_msvc	--force-msvc	auto	Disables default MSVC selection, and instead requires the stated version. This option can be used to override the default searching behavior. **Note**: Use this option only when the above options don't work for you.
force_winsdk	--force-winsdk	auto	Disables default Windows SDK selection, and instead requires the stated version. This option can be used to override the default searching behavior. **Note**: Use this option only when the above options don't work for you.

Default selection logic of MSVC version and Windows SDK version

Now, we will explain the default logic for selection, which may help trouble-shooting an inoperable WAF configuration.
By default, the selection of the MSVC version to use is done by the following logic:

1. Enumerate the installed MSVC on the machine by consulting the registry.
   This means that MSVC must be installed through a Microsoft provided installer.
   Note: Note that enumeration can be different for x86 and x64, if no toolchain can be found for the current target architecture.
   x64-hosted compilers are preferred over x86-hosted compilers because of the less restrictive virtual memory limit.
2. Select the most preferred version for compilation (this is set by Crytek), which was enumerated in step 1.
   For 3.8.4, the order is 11.0, then 14.0, then 12.0, then any other matching minimum requirement.
3. The behavior in step 2 can be modified through "minimum_msvc_compiler" (never select older versions) or "force_msvc" (select exactly this version)
   Note: If the version specified with "force_msvc" is not enumerated in step 1, the configuration of WAF will fail with an error.
4. If no version of MSVC was enumerated that satisfies step 2 and 3, a random version is selected and a warning is emitted.
   When you receive this warning, please install one of the supported versions as mentioned above.

Then, after MSVC is selected, WAF will select the Windows SDK version:

1. Enumerate the installed Windows SDK on the machine by consulting the registry.
   Note: This means that the SDK must be installed through a Microsoft provided installer.
2. Select the Windows SDK by using the default-supported version as listed in the table above under "Recommended Windows SDK"
3. The behavior in step 2 can be modified through "force_winsdk" (select exactly this version)
   Note: For Windows 10 and newer, you must specify the full version number (ie, "10.0.10240.0")

Troubleshooting: Using --auto-detect-verbose to find problems with WAF selection

It can be hard to find out why WAF does not work as expected given a host configuration with various versions of MSVC and Windows SDK installed.
To make it easier to find out why a problem exists, an option has been added to explain the selection logic used by WAF.

To view the extended output, open a command prompt with the current directory set to the CRYENGINE root folder, and then run:
`cry_waf.exe configure --auto-detect-verbose True --generate-vs-projects-automatically False`
WAF should provide output detailing the decisions made during configuration of MSVC and Windows SDK as explained above.

If the problem remains, open a support ticket, attaching at least the output of the above command, and the expected behavior.
Note: Only the combinations listed in the table above under the heading "Windows SDK versions supported" are directly supported by Crytek.

Troubleshooting: Compilation fails with WAF after configuration selection fails

After changing the configuration of MSVC or Windows SDK, it's possible that existing temporary files are causing problems.
Consider using "Clean Solution" from Visual Studio before building the first time after switching MSVC versions.

Troubleshooting: WAF gives warnings about being unable to find certain files

Because WAF supports various versions of MSVC, the dependencies of the compiled projects can be different depending on which version of MSVC is used during compilation. WAF will automatically attempt to copy appropriate dependencies (as listed under "Note regarding redistributable libraries") when the project is compiled. However, installations of Visual Studio as well as installations of Windows SDK do not always include the redistributable libraries (they are optional, and disabled by default). We recommend updating your installation to provide these libraries to prevent run-time issues on other machines.

In the case where the installer of your game will install the appropriate redistributables along with the game, it is safe to ignore those warnings.
Note: Redistributables for debug-configuration builds do not exist, and such builds can only reliably be run on the machines with that version of MSVC installed.

For users of Visual Studio 2015, please note that the current distribution of MSVC seems to not include the MBCS version of the MFC libraries, no matter what configuration is selected during installation. This appears to be an oversight in the installer.

As a workaround, it's possible to copy mfc140.dll and mfcm140.dll into VC\redist\x64\Microsoft.VC140.MFC folder (from Windows\System32), which should be available on your system if you selected the MFC optional package during Visual Studio installation.

This is only required when building the editor project.