Overview

This topic will guide you through the process of building CRYENGINE for the first time.

Requirements

This tutorial assumes that you have :

1. Downloaded engine source code
2. Downloaded 3rd Party SDKs
3. Copied engine and project assets to your project (otherwise you will be presented with a black screen)

The following pages explain each step in more detail:

- Using C++ Project Template With Custom Engine
- Using GameSDK with Custom Engine
- Getting Started With git

Chapters:

Related Pages:

- WAF References
- WAF FAQ
- WAF Interaction
- WAF Extensions
- WAF Tutorials
- Getting Started with WAF

Introduction:

WAF is a very powerful meta build system actively developed by Thomas Nagy.

- https://waf.io/
- https://waf.io/book/

With WAF, the power of CRYENGINE can be unleashed on a variety of platforms using a the most common compilers. Its Python nature makes it easier to extend and debug than other build systems. CryEngine has made heavy use of its extendibility while keeping the core mostly untouched. Extensions such as a simple GUI, integration into Visual Studio, Incredibuild support or uber file support enhance the workflow of large scale products even further.

A quick overview of support platforms and compilers can be found here:

- WAF Supported Platforms
- WAF Supported Compilers

Running WAF

Before you are able to build the CRYENGINE , you will need to configure the WAF Build System to your needs.

To do so, you will need to locate the cry_waf exectuable file. It is worth noticing that all interaction with the WAF Build System will be handled by this file.

Windows:

- Locate and execute: <*SDK_DIR>/cry_waf.exe*

Linux:

- Locate and execute: <*SDK_DIR>/cry_waf.sh* using ***Python 2.7***

When you run CryWAF the first time, the WAF Interface will ask you for a selection of user specific configuration options. The following section will guide you thorough the most important one.

Each selection can be changed individually after the configuration step has finished.
To do this execute cry_waf again to bring up the WAF GUI and select *Options*.

Automatic Solution Generation:

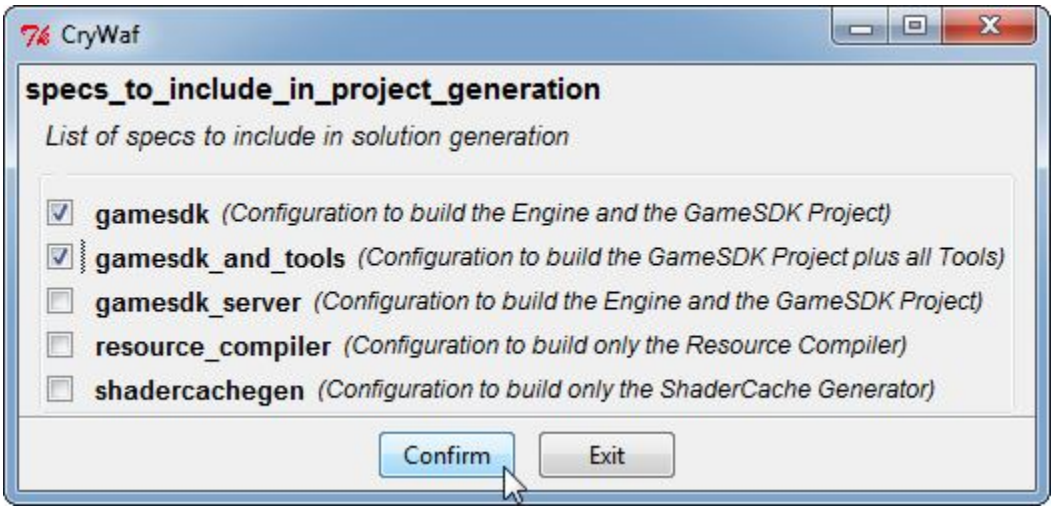
If enabled WAF is able to create a project for your favorite IDE.

The CRYENGINE is collection of many modules which interact with each other in various ways. The more you get to know the CRYENGINE , the more modules you will discover and potentially use.

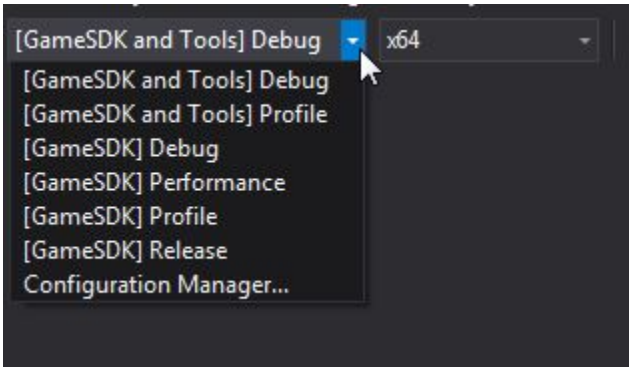
WAF allows you to categorize all modules within the CRYENGINE space into Specs. Depending on your Specs selection, you will be able to see the various dependent modules in the final generates solution.

There are two Specs which should be of initial interest to you:

- **GameSDK** - *allows you to build and run the CRYENGINE SDK sandbox game. In essence the sandbox game that is shipped as part of the C RYENGINE .*
- **GameSDK and Tools** *- allows you to build and run the CRYENGINE SDK sandbox game as well as tools, such as the CRYENGINE Editor.*



If you are working on the game as well as the editor. Select both Specs. This allows you to quickly switch between buildpipelines. After all if you are working on the game only, you are not interested on waiting for the Editor to re-compile.



Example result in Visual Studio

Compiling The Project

Once Cry WAF is configured you will be able to compile the various CRYENGINE components. This can be done via the WAF Command-Line Interface or from within the chosen IDE.

Option 1: Using the command line

Using the WAF Command-Line Interface is straight forward but requires some knowledge about how the WAF Command is put together.
In the previous section we already located the cry_waf entry point executable. This time you will instruct it to build the CryEngine component you are interested in.

A build command is made of the following component:

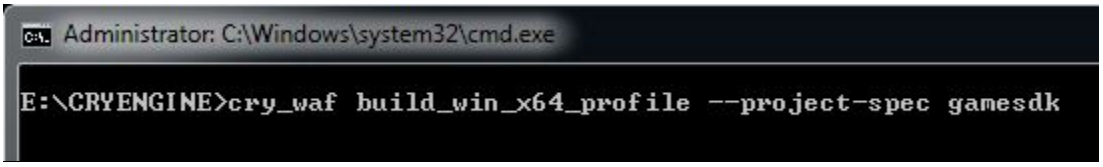
- cry_waf build_<platform>_<compiler>_<configuration>

where:

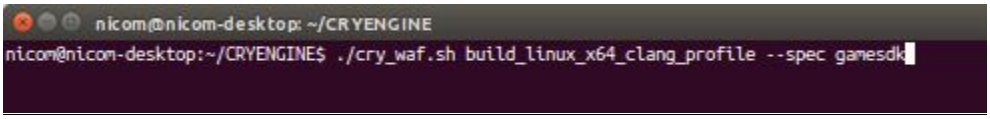
- <platform> stands for the target platform.
- <compiler> stand for the target compiler.
- <configuration> for the target configuration.

cry_waf has to be executed from the root of the CryEngine SDK folder!

Example Windows:

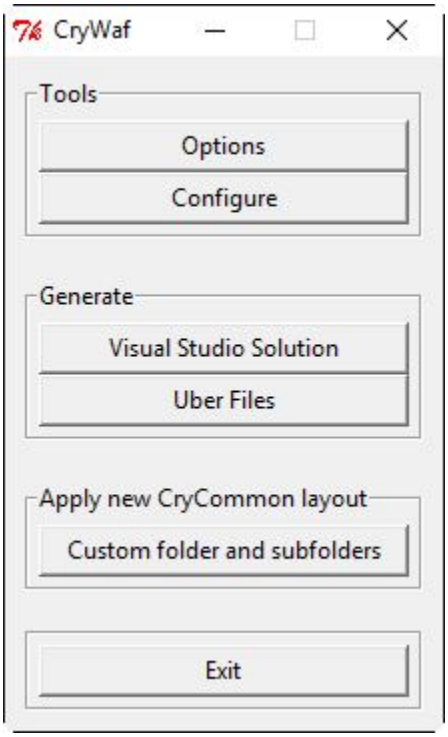


Example Linux:



Option 2: Using Visual Studio

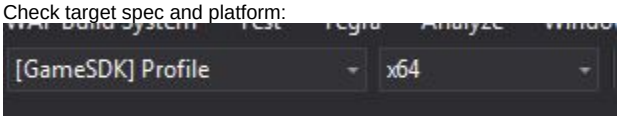
To generate a Visual Studio solution execute *<SDK_DIR>/cry_waf.exe* and select **Regenerate -> Visual Studio Solution** from the WAF Interface.

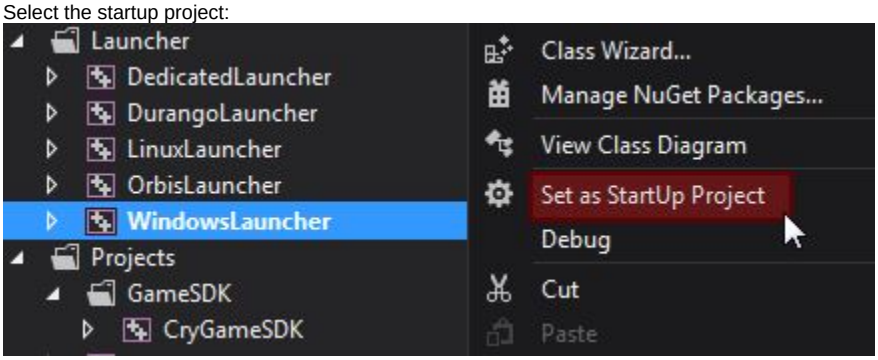


WAF will generate a Visual Studio Solution file for you which can be found in *<CRY_SDK>/Solutions* called *CRYENGINE.sln*.

Once you have opened the solution, ensure that you have the spec and platform selected that you want to build, as well as the correct startup project.

Example compile and launch GameSDK [Profile] - X64:

1. Check target spec and platform:


2. Build the game*:*
Press F7 or go *Build-> Build Solution*.
3. Select the startup project:


4. Launch the game:
Press *F5* or *Debug -> Start Debugging*

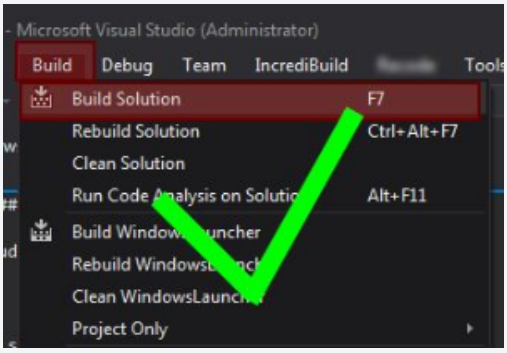
Black Screen

If you startup the game and you are faced with a black screen. Check that you've installed all needed assets into your source code root directory.

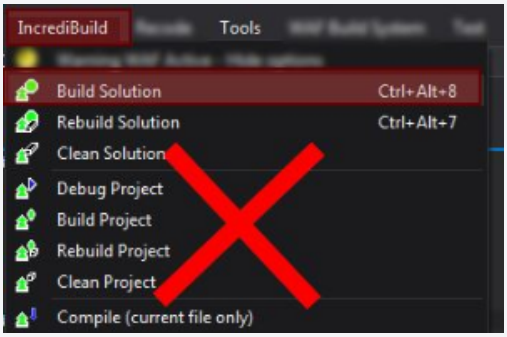
Incredibuild Users

Only use *Build-> Build Solution*

Do not use the *IncrediBuild* Build Interface.
IncrediBuild tasks are started from within CRY WAF itself.



OK



DON'T USE