Retrofit C++17 to large Visual Studio codebases

or how **Clang Power Tools** came to be

Gabriel Diaconița | gabriel.diaconita@caphyon.com

Caphyon, Romania

Abstract

You work hard to bring great products to market. Newly written C++17 code is easier to read, less prone to bugs, safer and more performant. However, not even great software products are perfect, they're being continuously developed. What about existing Visual C++ code in your product codebase?

Clang Power Tools is a free, open-source tool that brings C++17 to your existing Visual C++ codebase, effortlessly modernizing thousands of files and millions of lines of code.

Why?

- The LLVM Compiler Infrastructure offers excellent tools for automatically modernizing large C++ codebases, bringing shiny modern features to production code.
- Visual Studio developers on Windows are not given easy access to these tools.
- Clang Power Tools brings clang-format, clang-compile and clang-tidy directly into Visual Studio.

What do I need?

1. Download and install Clang for Windows (LLVM pre-built binary, v4.0-7.0)
2. Clang Power Tools directly from the Visual Studio Marketplace.

Got a build automation server?
Clang Power Tools integrates seamlessly with popular automation servers.
Keep builds fast by compiling only translation units affected by the modified source / header files.

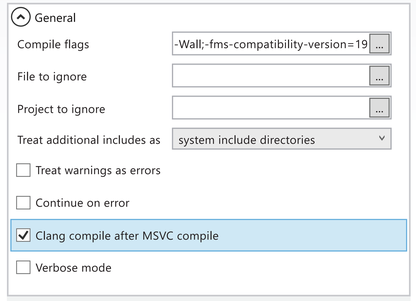
Configure once, run everywhere.
cpt.config versioned configuration files bring tailored Clang Power Tools settings to your entire team.

Clang Compile

Clang compile is the first step you take towards modernization. It will gradually lead you to make the code standard-compliant and weed out potential issues, such as double implicit conversions, integer promotions, and more. After your code compiles successfully with Clang you can then move to more advanced tools, like Tidy, LibTooling and Static Analyzer.

It also serves as gateway towards compliance with the latest Visual C++ compilation mode `/permissive-`.

Clang Power Tools can automatically clang compile every source file directly after compiling with MSVC.



Clang compile after MSVC compile
Automatically run Clang compile on the current source file, after successful MSVC compilation.

Clang Tidy

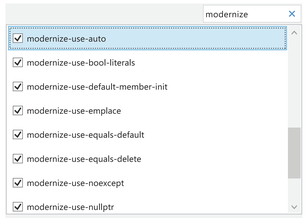
After getting your codebase to compile with clang, you can use clang-tidy to perform large scale refactorings, called modernizers. This will retrofit important C++17 features to your old code, increasing ease of reading, safety, and code performance.

Examples of modernizers:

- `modernize-use-auto` : inserts the `auto` type specifier for variable declarations to improve code readability and maintainability

```
std::vector<int>::iterator I = my_container.begin();
// transforms to:
auto I = my_container.begin();
```

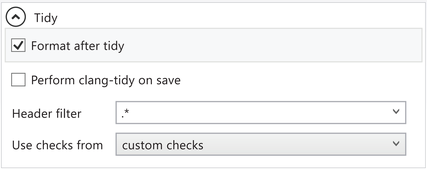
- `modernize-make-shared/unique`: inserts `make_shared` and `make_unique` for smart pointer exception safety and memory optimization
- `modernize-use-override` : inserts `override` keyword to virtual functions overrides
- `modernize-loop-convert` : converts raw loops to range-based for loops



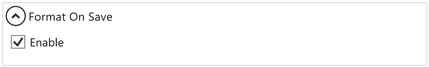
modernize-use-auto
This check is responsible for using the auto type specifier for variable declarations to improve code readability and maintainability. For example:

Clang Format

Even after modernizing the code, keeping it consistent with a particular formatting style is a serious challenge for any team. Eliminate the styling-related part of code reviews by having your source files automatically formatted using clang-format.



After selecting a formatting style (LLVM, Google, Mozilla and more), you can have source files auto-formatted at each Save operation.



Eager to see Clang Power Tools in action? Make sure you check out the following links:

www.clangpowertools.com
The project's official site

Get Clang Power Tools now
from Visual Studio Marketplace

github.com/Caphyon
/clang-power-tools

Clang Power Tools was created with ♥ by some very awesome people at CAPHYON

https://caphyon.com