

Announcing full support for a C/C++ conformant preprocessor in MSVC



Elnar Dakeshov

March 27th, 2020 | 20 | 0

Update June 8, 2021: The [C11 and C17 announcement post](#) has instructions on how to get the latest Windows SDK that works with the conformant preprocessor.

We are excited to announce full support for a conformant preprocessor in the MSVC toolset starting with Visual Studio 2019 version 16.6 Preview 2.

Since the original [blog post](#) announcing preprocessor conformance changes, we've come a long way and are now ready to announce the completion of the C/C++ conformant preprocessor and its move to a non-experimental, fully supported state via the `/Zc:preprocessor` switch. Alongside standard conformance, the preprocessor also supports C++20's `__VA_OPT__` and is also available in the C language mode.

What's new:

To reach conformance, a couple of additional features have been added to the preprocessor and MSVC compiler, including a variety of bug fixes, `__VA_OPT__`, and `_Pragma` support.

Bugfixes involving various parts of the preprocessor, from [parameter expansion](#) and special macro names like `__FUNCSIG__` to reporting [arity errors](#) and line number fixes. Special thanks to Edward Diener for providing a lot of valuable feedback!

`__VA_OPT__`

`__VA_OPT__` is a new feature of variadic macros in C++20. It lets you optionally insert tokens depending on if a variadic macro is invoked with additional arguments. An example usage is comma elision in a standardized manner.

```
#define M(X, ...) X __VA_OPT__(,) __VA_ARGS__
M(3) // expands to 3
M(3, 4) // expands to 3, 4
```

`_Pragma`

The `_Pragma` operator has been one of the long-standing deficiencies of the preprocessor, and a blocker in being standard conformant in C++ and C99. Though MSVC had the non-conformant `__pragma`, the semantics differ in that `_Pragma` takes a string literal as its parameter instead of a series of preprocessor tokens. This feature is now implemented.

```
_Pragma("once")
#define GUARD _Pragma("once")
```

Miscellaneous

There were some [contextual keyword changes relating to modules](#). These changes unblock further C++20 modules work.

Preprocessor-only output (via /E and /P) is now prettier, reducing the amount of line directives and fixing some formatting [issues](#).

Enabling the conformant preprocessor in your environment

All that is needed to use the conformant preprocessor is to add `/Zc:preprocessor` to your compilation flags. The flag is available in C and C++ language modes. It works with any language level, but we plan to enable it for `/std:c++latest` in a future release.

Language mode	/Zc:preprocessor
/std:c++14	Not implied
/std:c++17	Not implied
/std:c++latest	Implied in a future update (Not implied in VS 2019 v16.6)

The conformant preprocessor can be tested by checking if the macro `_MSVC_TRADITIONAL` is defined and set to 0.

```
#if defined(_MSVC_TRADITIONAL) && _MSVC_TRADITIONAL
// old behavior
#else
// new behavior
#endif
```

The legacy preprocessor is not going anywhere, it will continue to serve as a compatibility layer for old code, but it will only be serviced with the intention of keeping old code working. Additionally, the `/experimental:preprocessor` switch is still available and will activate `/Zc:preprocessor` in VS 2019 v16.6 but will be removed in a future release. Any projects that have been configured to use the experimental switch should migrate to the supported version.

What's next:

Improved diagnostics are in the works, which will provide a better expansion context for macro invocation and errors.

This feature is not currently implied by any other flags, but we are planning to include it in `/std:c++latest` once we stabilize the public SDK headers from using non-conformant macros. Using the latest Windows SDK is advised, as many of the noisy warnings are fixed in later SDK versions.

Call to Action

Let us know how the conformant preprocessor works for you! Get it in Visual Studio 2019 version 16.6 Preview 2 (please see <https://visualstudio.microsoft.com/vs/preview/> for download links) and try it out.

As always, we welcome your feedback. We can be reached via the comments below or via email (visualcpp@microsoft.com). If you encounter problems with Visual Studio or MSVC, or have a suggestion for us, please let us know through Help > Send Feedback > Report A Problem / Provide a Suggestion in the product, or via [Developer Community](#). You can also find us on Twitter ([@VisualC](#)).



Elnar Dakeshov Software Engineer 2, Visual C++ Front End
Follow

Read next

Porting a C++/CLI Project to .NET Core

One of the new features of Visual Studio 2019 (beginning with version 16.4) and .NET Core 3.1 is the ability to build C++/CLI projects targeting .NET Core. This can be ...

 **Mike Rousos**

March 30, 2020

 11 comments

New templates for debugging CMake projects on remote systems and WSL in Visual Studio 2019

We heard your feedback that it can be difficult to configure debugging sessions on remote Linux systems or the Windows Subsystem for Linux (WSL). In Visual Studio 2019 ...

 **Erika Sweet**

April 3, 2020

 3 comments



 **comments**

 ments are closed. [Login to edit/delete your existing comments](#)

 **shbouwhuis@gmail.com** March 27, 2020 4:25 pm  0

I've been looking for the state of C++20 support in Visual Studio 2019, but can't seem to find a page with what is and what isn't implemented yet.

Is it language-wise feature complete? Is STL feature complete? Is it production ready? Is it in alpha state?

Could you please inform us of the progress? I saw the Visual Studio roadmap, but couldn't get a clear picture.

 **Me Gusta** March 27, 2020 4:52 pm  0



 

While my comment with direct links is awaiting moderation, if you go to the cppreference website, you should find a page there named C++ compiler support.

 **shbouwhuis@gmail.com** March 30, 2020 6:32 pm  0

I know that page, but I thought that page was always fairly far behind (old information). But I think that page together with the one Laurent Lessieux suggested will give me a good idea. Thanks!

 **Me Gusta** March 31, 2020 10:37 pm  0

That page is very up to date. It has everything up to the current preview. To give an idea, the numbers in the VC column refers to the compiler toolset version. You can get this from running cl.exe on the command line. As an example, the current release version gives:

Microsoft (R) C/C++ Optimizing Compiler Version 19.25.28612 for x64

So the current release is 19.25. The current preview is 19.26. If you look through the VC column you will find features listed with 19.25 and 19.26, and if you look for the C99 preprocessor under the C++11 features then you will see that it is marked as fully implemented as of version 19.26.

Heck, it even has a couple of 19.27 library features listed, and that version isn't




even in preview yet. They are probably getting that by monitoring the STL repository on GitHub.
So it is quite accurate.

 **laurent lessieux** March 27, 2020 11:53 pm  0  

Are you looking for something like that?

[Visual Studio C++ Conformance](#)

It is the best I could find. Very interesting to see the progress being done and realizing how much is left. I can't wait to get the final C++20 support.

 **shbouwhuis@gmail.com** March 30, 2020 6:33 pm  0  

Yesss, that page is actually pretty good! I searched on the Microsoft site instead of Github.
Thanks for the info!

f

🐦

in

 **Alastair Taylor** March 27, 2020 5:27 pm  0  

```
#if defined(_MSVC_TRADITIONAL) && _MSVC_TRADITIONAL
// new behavior
#else
// old behavior
#endif
```

Is that correct? As <https://devblogs.microsoft.com/cppblog/msvc-preprocessor-progress-towards-conformance/> has it as

```
#if defined(_MSVC_TRADITIONAL) && _MSVC_TRADITIONAL
// Logic using the traditional preprocessor
#else
// Logic using cross-platform compatible preprocessor
#endif
```

 **Elnar Dakeshov**  March 27, 2020 8:28 pm  0  

Fixed, thanks Alastair.

 **Rick de Water** March 27, 2020 6:16 pm  0  

"but we plan to enable it explicitly for /std:c++latest in a future release."

Don't you mean implicitly?

 **Elnar Dakeshov**  March 27, 2020 8:52 pm  0  

We mean that /Zc:preprocessor will be implied by /std:c++latest in a future release. Updated to clear up wording, thanks!

 **Paul Topping** March 27, 2020 10:52 pm  0  

Feedback

One of the things I have always wanted is a way to generate the preprocessed intermediate file (.i) for a given .cpp with just a menu command. This would be very handy for looking into the occasional macro blunder. I'll admit that I haven't looked for it in years but, AFAIK, we still have to change the project to do this via compiler arguments and then change it back afterwards. It's not super hard but it also can't be hard to just provide a Preprocess Only menu command to produce the file without changing the project.



Paul Topping March 27, 2020 10:56 pm 0



I looked again, expecting a Preprocess command within the Build category of commands but no such luck. In my configuration, I would put it right next to Compile in the Build menu.



Elnar Dakeshov March 29, 2020 1:43 am 0



Hi Paul, I've gone ahead and made a suggestion ticket for this feature, so that it's on our radar and can be properly assigned to the right team:

<https://developercommunity.visualstudio.com/idea/967496/there-should-be-a-way-to-generate-a-preprocessed-f.html>



Alexander Sklar March 31, 2020 12:09 am 0



+100 to Paul's point. It is frankly inconceivable that there isn't a straightforward way to preprocess a cpp file from the UI without having to build your own msbuild task or some other way in order to pass the preprocess to file flag



shbouwhuis@gmail.com March 30, 2020 6:36 pm 0



The bigger problem is that changing the project settings means that EVERYTHING has to be recompiled again. So, a very base project in your project chain potentially means hours of wasted time.



Victor Derks March 28, 2020 4:19 pm 0



Enabling this option generates:

CppUnitTest.h(120,114): warning C5104: found 'L#methodName' in macro replacement list,

for the Visual Studio C++ unit test headers that are shipped with Visual Studio 2019 16.6. Preview 2.0

It seems that using the new preprocessor is not possible for VC++ unit test projects as TEST_METHOD macro cannot be expanded correctly.



Elnar Dakeshov March 29, 2020 1:41 am 0



Hi Victor, I've made a feedback ticket for this so that it's on our radar:

<https://developercommunity.visualstudio.com/content/problem/967495/cppunittest-does-not-work-with-the-conformant-prep.html>



e4lam March 28, 2020 11:45 pm 0



Can the latest public Windows SDK headers compile with this new preprocessor?



Elnar Dakeshov March 29, 2020 1:39 am 0





NN March 31, 2020 3:25 am 0



With /Wv:18 SDK headers compile.

But WDK headers are not.

There Are only few macros affected in the WDK headers.

Is it possible to fix them in the upcoming 20H1 WDK release ?

If it can speed up, I can submit a patch 😊

Relevant Links

[Getting Started with C++ in VS](#)
[Bring Your Existing C++ Code to VS](#)
[C++ Code Editing & Navigation](#)
[C++ Unit Testing](#)
[C++ Debugging & Diagnostics](#)
[Collaborating with Your Team in VS](#)
[C++ Windows Development](#)
[C++ Linux Development](#)
[C++ Android & iOS Development](#)
[C++ Game Development](#)

Topics

[C++](#)
[Announcement](#)
[CMake](#)
[New Feature](#)
[Linux](#)
[Visual Studio Code](#)
[Diagnostics](#)
[General C++ Series](#)
[performance](#)
[Vcpkg](#)
[Writing Code](#)

Archive

[December 2022](#)
[November 2022](#)
[October 2022](#)
[September 2022](#)
[August 2022](#)
[July 2022](#)

June 2022
May 2022
April 2022
March 2022
February 2022



Stay informed



	What's new	Microsoft Store	Education	Business	Developer & IT	Company
f Twitter in	Surface Pro 9	Account profile	Microsoft in education	Microsoft Cloud	Azure	Careers
	Surface Laptop 5	Download Center	Devices for education	Microsoft Security	Developer Center	About Microsoft
	Surface Studio 2+	Microsoft Store support	Microsoft Teams for Education	Dynamics 365	Documentation	Company news
	Surface Laptop Go 2	Returns	Microsoft 365 Education	Microsoft 365	Microsoft Learn	Privacy at Microsoft
	Surface Laptop Studio	Order tracking	Education consultation appointment	Microsoft Power Platform	Microsoft Tech Community	Investors
	Surface Duo 2	Personal shopping appointments	Educator training and development	Microsoft Teams	Azure Marketplace	Diversity and inclusion
	Microsoft 365	Microsoft Store Promise	Deals for students and parents	Microsoft Industry	AppSource	Accessibility
	Windows 11 apps	Flexible Payments	Azure for students	Small Business	Visual Studio	Sustainability