

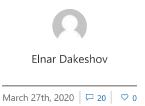
in

☆ Theme

Q

Login

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



Update June 8, 2021: The <u>C11 and C17 announcement post</u> 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 <u>blog post</u> 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 <u>parameter expansion</u> and special macro names like <u>FUNCSIG</u> to reporting <u>arity errors</u> 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 <u>contextual keyword changes relating to modules</u>. 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 <u>issues</u>.

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 <code>/experimental:preprocessor</code> switch is still available and will activate <code>/Zc:preprocessor</code> 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:

in

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

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

f

comments

in aments are closed. Login to edit/delete your existing comments



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

√ ⊗

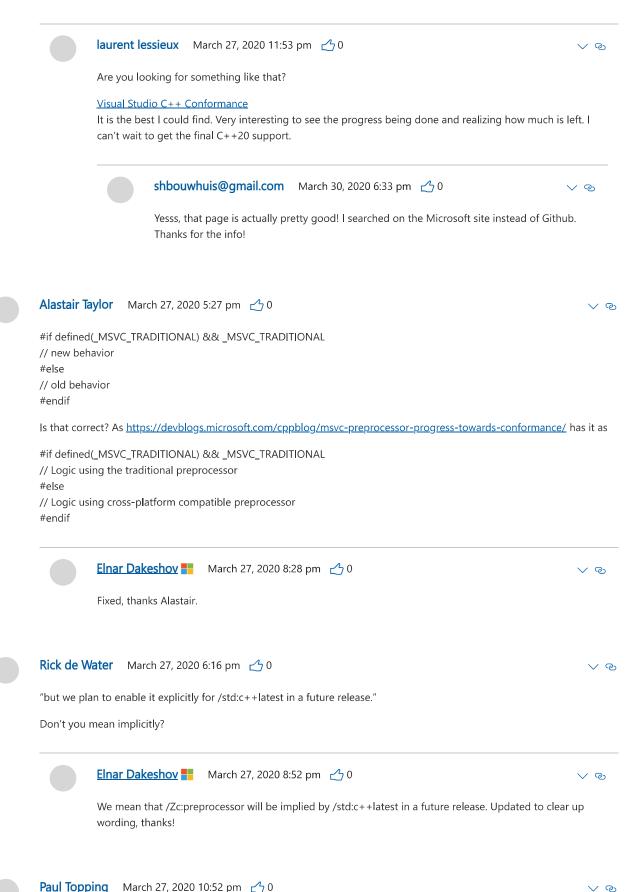
∨ ⊗

V @

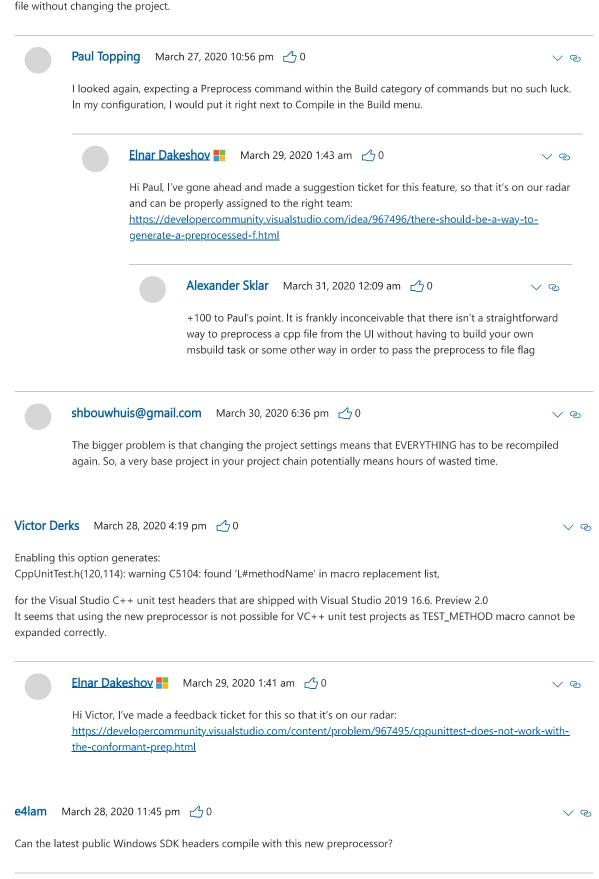
So it is quite accurate.

f

in



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.

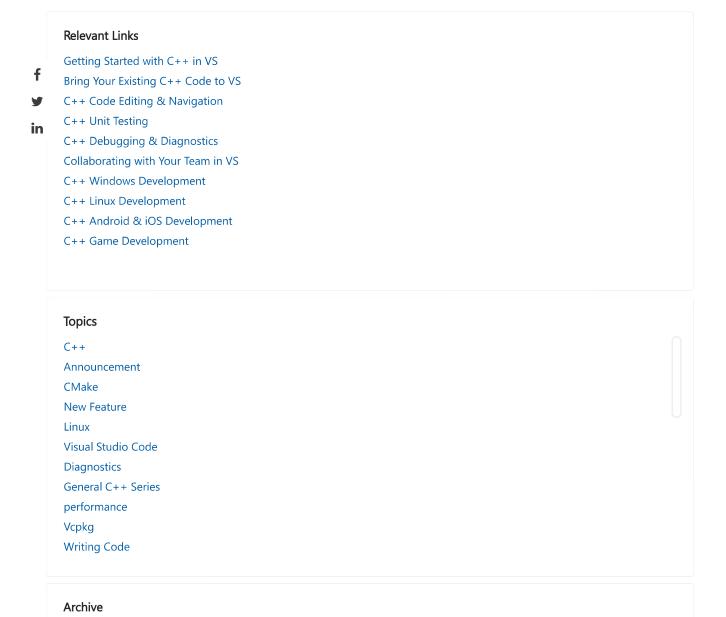


Elnar Dakeshov March 29, 2020 1:39 am 🖒 0

in

✓ ⊗





December 2022 November 2022 October 2022 September 2022 August 2022 July 2022 June 2022 May 2022 April 2022 March 2022

1- 2021

Stay informed



	What's new	Microsoft Store	Education	Business	Developer & IT	Company
f	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
in	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	Microsoft Power Platform	Microsoft Tech Community	Investors
	Surface Duo 2	Personal shopping appointments	appointment	Microsoft Teams	Azure Marketplace	Diversity and inclusion
	Microsoft 365	Microsoft Store Promise	Educator training and development	Microsoft Industry	AppSource	Accessibility
	Windows 11 apps	Flexible Payments	Deals for students and parents	Small Business	Visua l Studio	Sustainability
			Azure for students			
		Sitemap Co	ntact Microsoft Privacy	Terms of use Trademarks	Safety & eco About our	ads © Microsoft 2022