**C11 and C17 Standard Support Arriving in MSVC**

Avatar

Elnar

September 14th, 2020

[C11 and C17 Standard Support Arriving in MSVC | C++ Team Blog (microsoft.com)](https://devblogs.microsoft.com/cppblog/c11-and-c17-standard-support-arriving-in-msvc/)

*Please see our*[*Visual Studio 2019 version 16.8 Preview 3 release notes*](https://devblogs.microsoft.com/cppblog/a-multitude-of-updates-in-visual-studio-2019-version-16-8-preview-3/)*for more of our latest features.*

Our team is happy to announce that C11 and C17 are becoming supported language versions in the MSVC compiler toolset starting with Visual Studio 2019 version 16.8 Preview 3!

For many years Visual Studio has only supported C to the extent of it being required for C++. Things are about to change now that a conformant token-based preprocessor has been added to the compiler. With the advent of two new compiler switches, /std:c11 and /std:c17, we are officially supporting the latest ISO C language standards.

**What’s in:**

All the required features of C11 and C17 are supported. This meant adding the following functionalities:

* \_Pragma
* restrict
* \_Noreturn and <stdnoreturn.h>
* \_Alignas, \_Alignof and <stdalign.h>
* \_Generic and <tgmath.h> support
* \_Static\_assert

IntelliSense works natively with these features as well, simply use a .c file extension for your source files or the /TC compiler switch to enable syntax highlighting for C code.

[https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image.png](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image.png)

Currently IntelliSense highlighting is only available for the keywords and not the macros introduced by the standard headers. This will be fixed in a later release.

Since C17 is essentially just a bug fix release of ISO C, with many defect reports being adopted, our support for C11 already includes all the relevant defect reports. At present, there are no differences between the C11 and C17 versions except for the \_\_STDC\_VERSION\_\_ macro, which expands to 201112L (for C11) and 201710L (for C17).

Here is an example that shows these features:

#include <assert.h>

#include <stdalign.h>

#include <stdlib.h>

#include <stdnoreturn.h>

#define NO\_WARN(X) \

\_Pragma("warning (push)") \_Pragma("warning (disable: 4146)") X; \

\_Pragma("warning (pop)")

// Pick stored or storei based on the type of the dst

#define store(x, y) \_Generic((x), \

data\*: stored, \

int\* : storei)(x, y)

typedef struct data {

\_Alignas(8) unsigned int i;

} data;

static\_assert(alignof(data) == 8, "data is not properly aligned");

void stored(data\* restrict dst, const data\* restrict src)

{

// Do not trigger warning 4245

dst->i = NO\_WARN(-(src->i));

}

void storei(int\* restrict dst, const int\* restrict src)

{

\*dst = \*src;

}

noreturn void my\_exit(int ret) {

exit(ret);

}

int main() {

data src, dst;

src.i = 5;

int i, j;

i = 10;

store(&src, &dst);

store(&i, &j);

my\_exit(0);

}

Since the inclusion of the token-based [conformant preprocessor](https://devblogs.microsoft.com/cppblog/announcing-full-support-for-a-c-c-conformant-preprocessor-in-msvc/), the two new C compiler switches imply /Zc:preprocessor. If you want to use the traditional, character-based preprocessor alongside C11 or C17, you will need to pass in the /Zc:preprocessor- compiler switch explicitly. We do encourage you to fix your source code where needed to not depend on the legacy, traditional preprocessor.

**What’s not:**

While there is currently no support for any C11 optional features, we are committed to providing the most impactful optional features in future releases. Atomic and threading support are on our roadmap. Support for Complex numbers is currently not planned and their absence is enforced with the proper feature test macros. Please go to [Visual Studio Developer Community](https://developercommunity.visualstudio.com/spaces/8/index.html) and voice your support for features you wish to be implemented. We are looking for your input that we are making the correct prioritizations.

Due to the nature of the Windows heap, aligned\_alloc support is missing. The alternative is to use [\_aligned\_malloc](https://docs.microsoft.com/en-us/cpp/c-runtime-library/reference/aligned-malloc?view=vs-2019).

Additionally, DR 400 support is currently unimplemented for realloc as this change would break ABI.

**Variable Length Arrays**

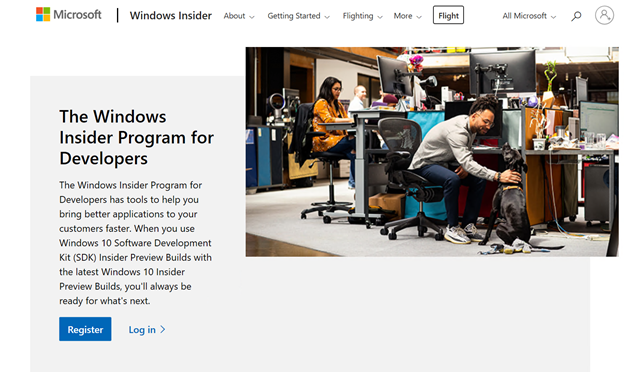
Astute readers will note that VLAs are also not supported. Variable length arrays are generally less efficient than comparable fixed sized arrays, and generally inefficient compared to equivalent malloc(), when safely and securely implemented. VLAs provide attack vectors comparable to those of the infamous gets() — deprecated and destined to removal — for opportunities of “shifting the stack” and other exploits. For these reasons we intend not to support VLAs as an optional feature in C11.

**C11 and C17: Getting Started**

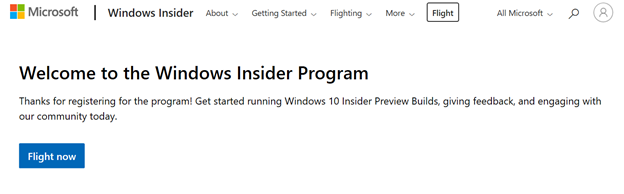
In order to use C11 or C17 in your programs, the latest Windows SDK updates are required to work properly with the conforming preprocessor (/Zc:preprocessor), and the new Universal C Runtime. Windows SDK releases correspond with Windows OS releases. Since there has been no Windows release with these changes, you will need an Insider Preview Windows SDK – a preview version of the Windows SDK that corresponds with Windows builds currently being flighted for Windows Insiders. Note that after installing the Insider Preview Windows 10 SDK, Visual Studio projects configured to use the latest Windows SDK will use the Insider Preview.

**Step 1: Log in with an Insider Microsoft Account**

Anyone can create a free Microsoft account (<https://signup.live.com/>) and then opt into the Insider program. Simply go to <https://insider.windows.com/en-us/for-developers> and click ‘Register’ and log in.

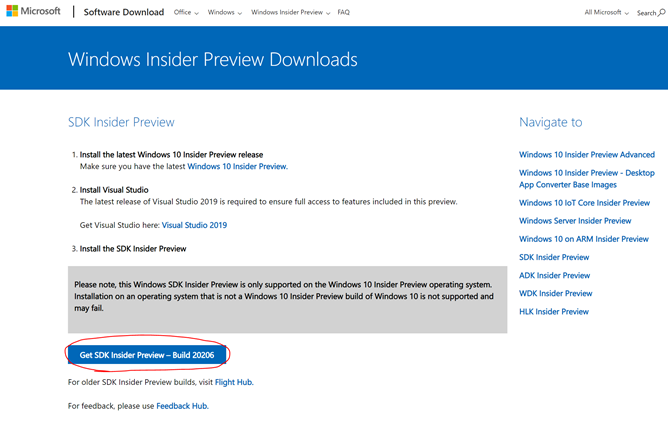
[](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-1.png)

After registering, you will be given the option to start flighting Insider versions of Windows, but this is not necessary to download and use the Insider Windows 10 SDK. Registering does not automatically opt in your machines to download new Windows flights.

[](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-2.png)

Once you hit this page, you do **not** need to click ‘Flight now’. Continue to the next step and download the Insider Preview Windows 10 SDK.

**Step 2: Download the Insider Preview Windows 10 SDK**

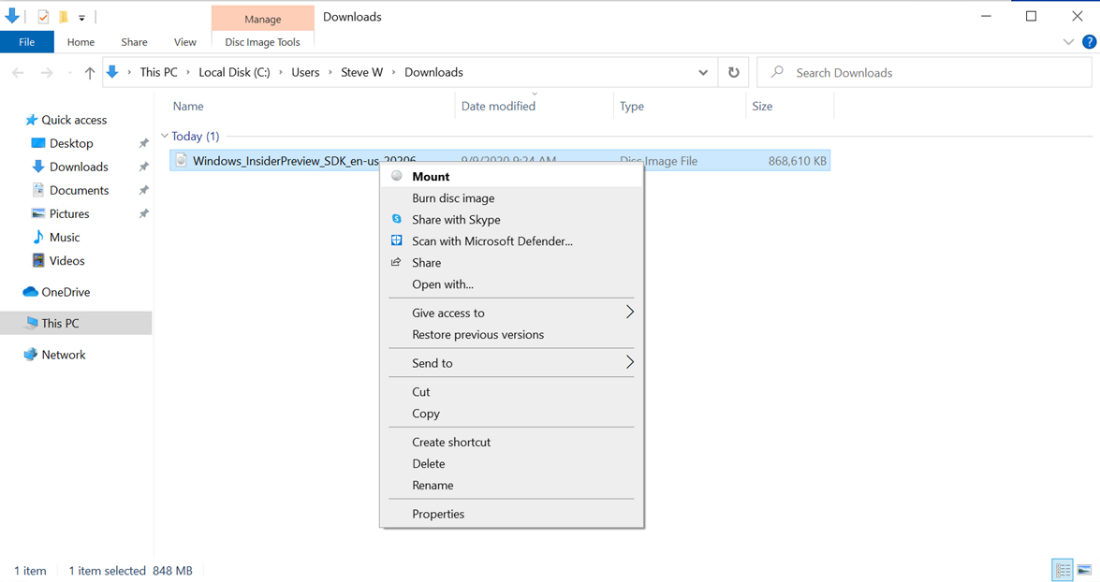
The Insider Preview Windows SDK can be installed from <https://www.microsoft.com/en-us/software-download/windowsinsiderpreviewSDK>. If this is not the page you receive, make sure you are logged in with your Microsoft account that has been opted into being an Insider. [](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-3.png)

The Insider page describes that using a Windows 10 Insider Preview OS is required. This is true for some of the content included in the Windows SDK, as they may depend on new APIs not present on older versions of Windows, but the Windows and Universal C Runtime headers will properly install and be usable without an Insider OS.

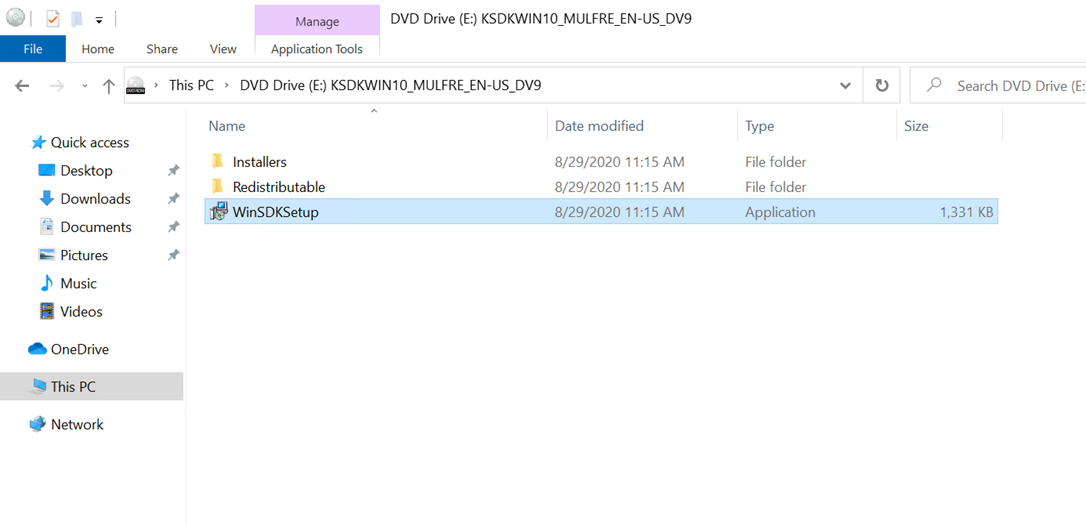
Click ‘Get SDK Insider Preview – Build 20206’ to begin download. Future versions of the Windows SDK will also work.

**Step 3: Install the Insider Preview Windows 10 SDK**

The Insider Preview Windows SDK will download as an .iso file.

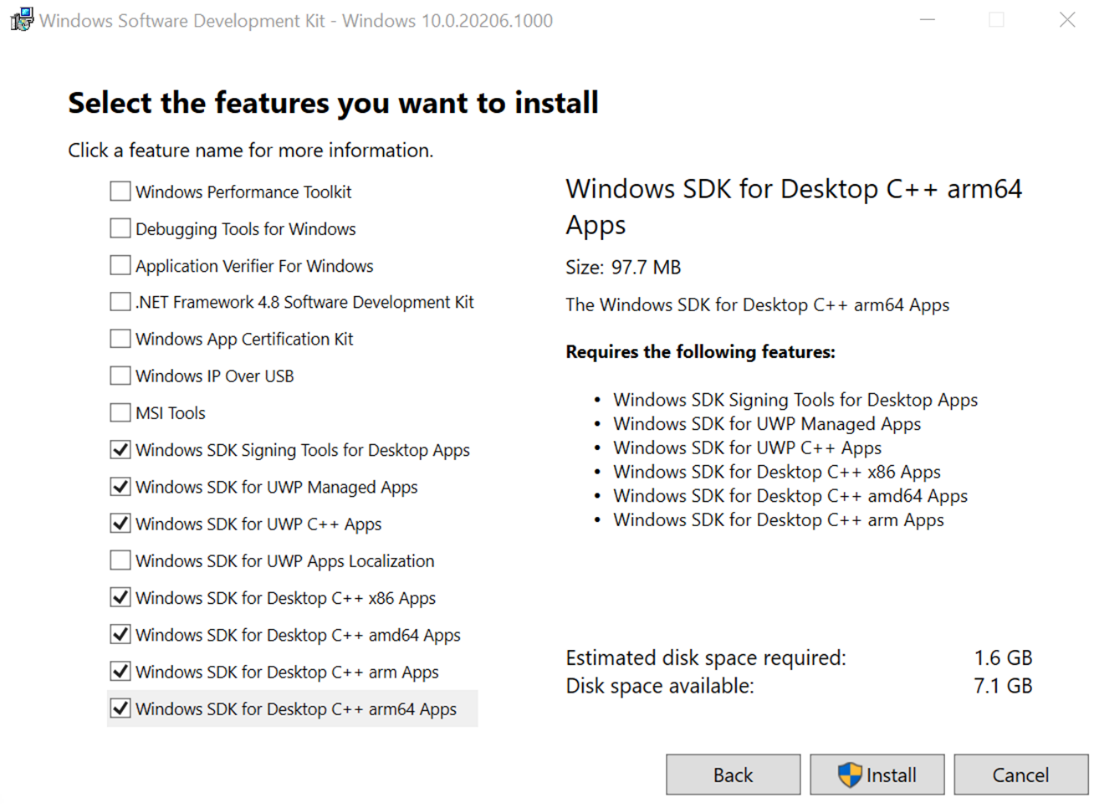
[](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-4.png)

Mount the .iso file and run WinSDKSetup.exe to start installation.

[](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-5.png)

Choose to install the Windows Software Development Kit on this computer and click next. You will have to choose whether to allow insights for Windows SDK usage and must accept the license agreement before you arrive at the feature installation page. The only features you should install (unless you are using an Insider Preview build of Windows 10) are:

* Windows SDK Signing Tools for Desktop Apps
* Windows SDK for UWP Managed Apps
* Windows SDK for UWP C++ Apps
* Windows SDK for Desktop C++ x86 Apps (if planning on building for x86)
* Windows SDK for Desktop C++ amd64 Apps (if planning on building for amd64)
* Windows SDK for Desktop C++ arm Apps (if planning on building for arm)
* Windows SDK for Desktop C++ arm64 Apps (if planning on building for arm64)

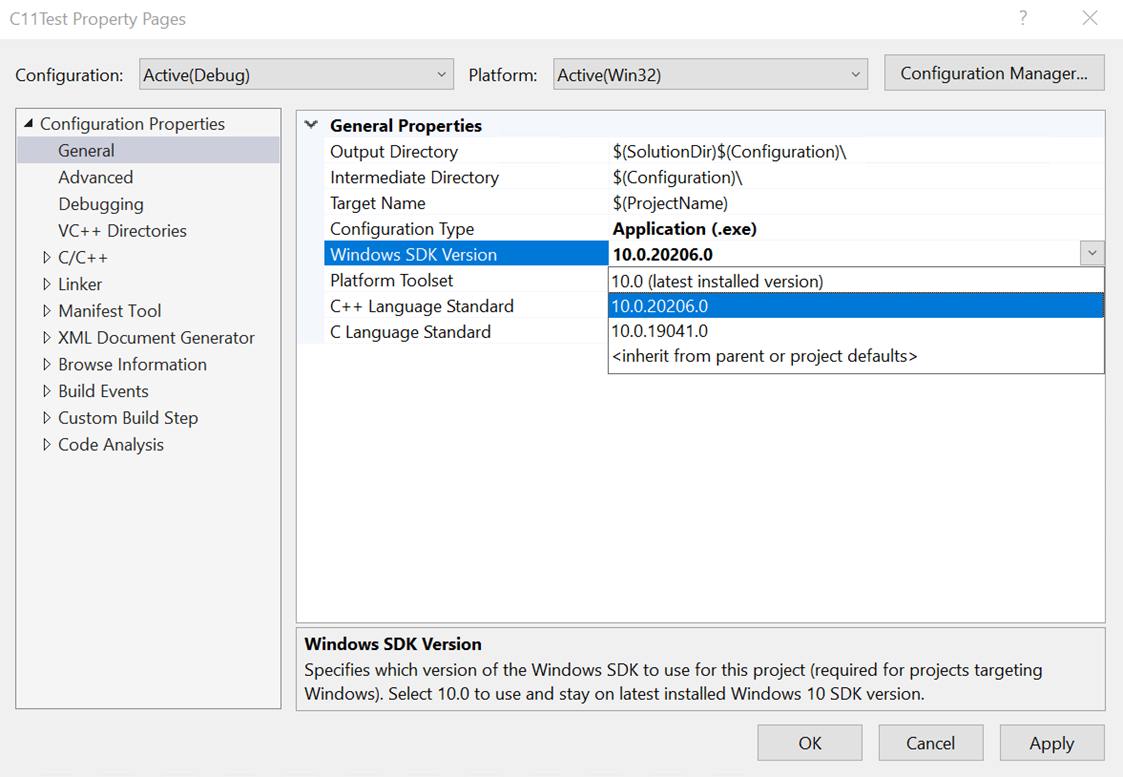
[](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-6.png)

The SDK will take a couple minutes to complete installation and then you can open Visual Studio 16.8 Preview 3.

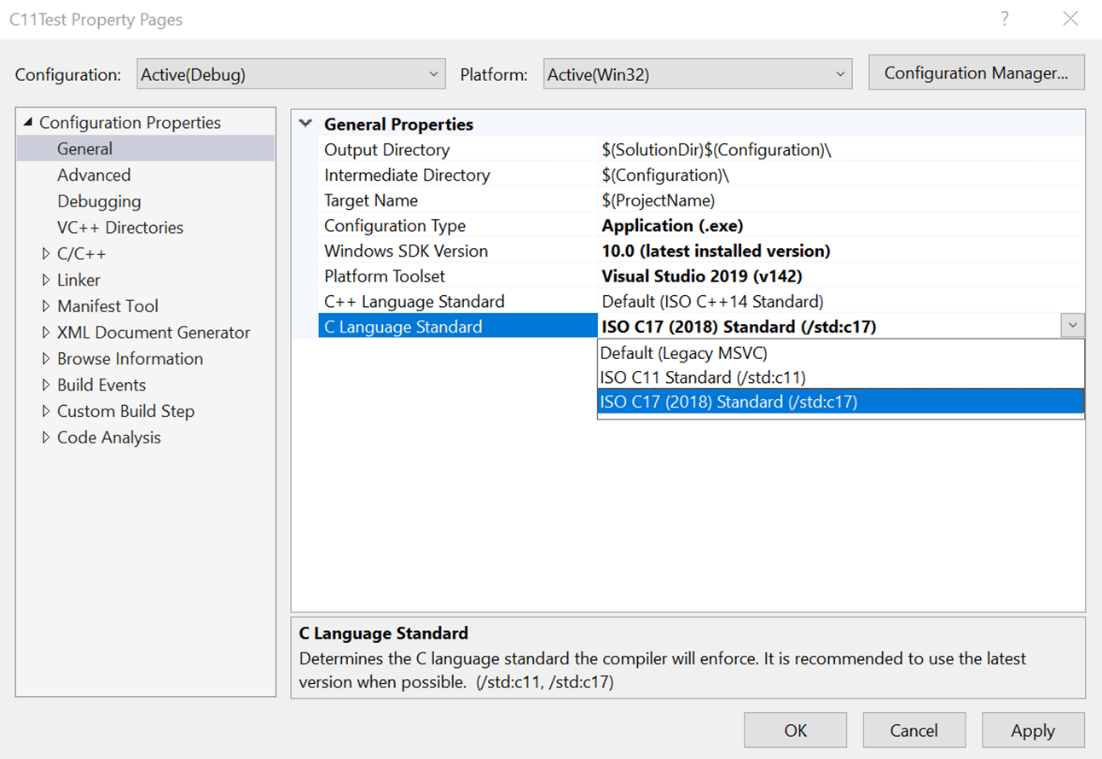
**Step 4: Configuring C11 or C17 mode in Visual Studio 16.8 Preview 3**

Support for C11 and C17 begins in Visual Studio 16.8 Preview 3. You can download the latest Preview via <https://visualstudio.microsoft.com/vs/preview/>.

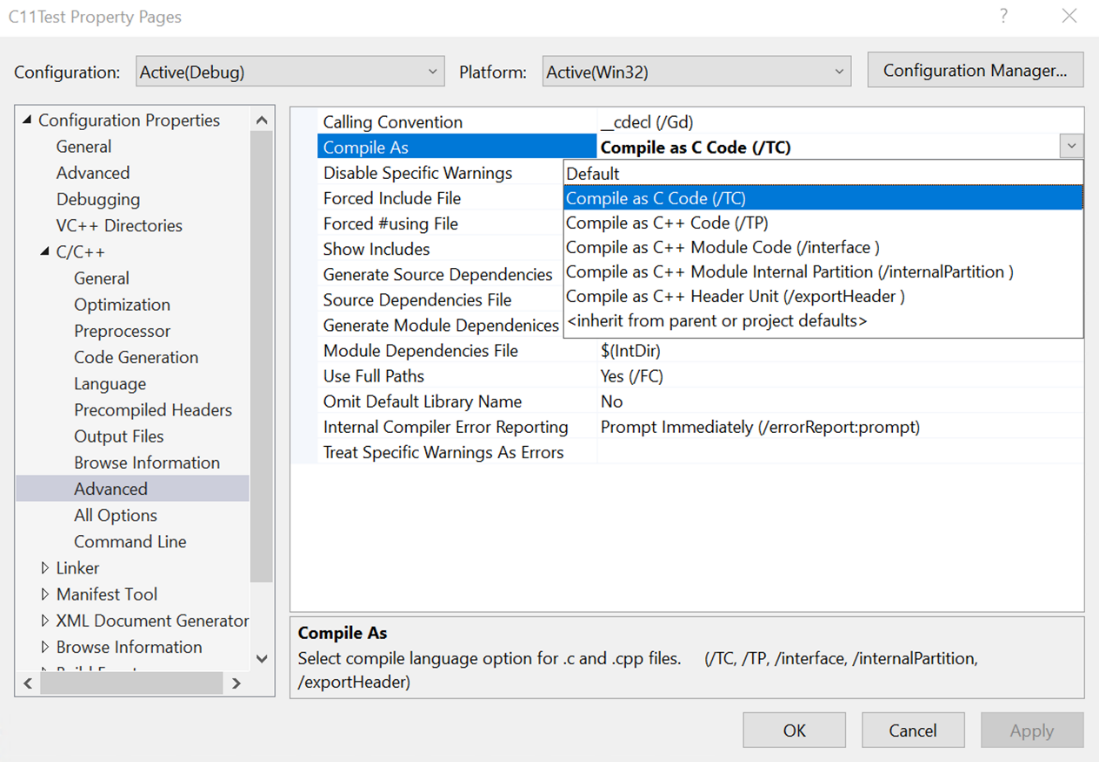
In your project, open the Properties page. First, we want to ensure that the project will use the Insiders Preview Window 10 SDK. Set Windows SDK Version to 10.0.20206.0 (or the latest Insider Preview Windows 10 SDK just installed).

[](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-7.png)

You will also see a new option: C Language Standard. Set this to ISO C11 Standard (/std:c11) or ISO C17 (2018) Standard (/std:c17).

[](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-8.png)

The C++ Language Standard is used when the language is C++ – this will be the default when the file extension is .cpp. The C Language Standard version is used when the language is C – this will be the default when the file extension is .c. To ensure the project is building with C11 or C17, you must ensure it is a .c file, or set the code to compile as C in the Properties tab.

[](https://devblogs.microsoft.com/cppblog/wp-content/uploads/sites/9/2020/09/word-image-9.png)

After these project changes, you’re ready to test out C11 and C17!

**In Conclusion**

We are very excited to finally support C11 and C17! As the highest voted Developer Community suggestion, we’re sure you’re excited too. If you want support for the optional features of C11, please let us know on [Developer Community](https://developercommunity.visualstudio.com/).

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](https://developercommunity.visualstudio.com/). You can also find us on Twitter ([@VisualC](https://twitter.com/visualc)).