Setting Up VS Code for Unreal Engine

How to set up Visual Studio Code as your IDE for use with Unreal Engine projects.

While **Unreal Engine (UE)** supports **Microsoft Visual Studio** as its default IDE for C++ projects in Windows, it also supports **Visual Studio Code (VS Code)** as a more lightweight, free, open-source alternative. Although VS Code does not have the same capabilities as Visual Studio out of the box, it is highly extensible, and is available for Windows, MacOS, and Linux, providing a common ground for developers working in multiple platforms.

However, VS Code requires extra manual setup to give it equivalent functionality to Visual Studio for Windows. This guide will walk you through these extra steps so that your VS Code environment is fully equipped for UE development.

> You do not need a full Visual Studio installation to use VS Code.

Required Setup

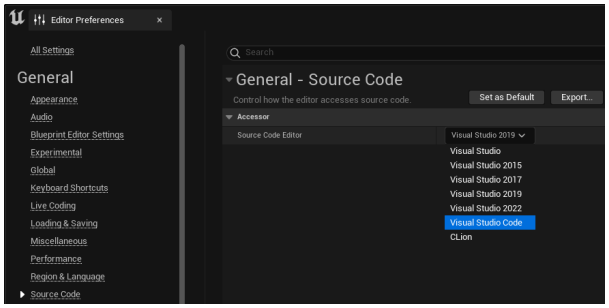
This guide assumes that you have installed Unreal Engine and created a C++ project with it.

Installing VS Code and Required Extensions for Your OS

1. Download and install VS Code as well as the official C/C++ extension pack and C# extension for VS Code. These are required for reading the source code for both Unreal Engine and its **Build Tools**.
2. Download and install the compiler toolset for your OS.
    - **Windows:** The Microsoft Visual C++ (MSVC) compiler toolset.
    - **Mac:** LLVM/Clang.
    - **Linux:** LLVM/Clang.

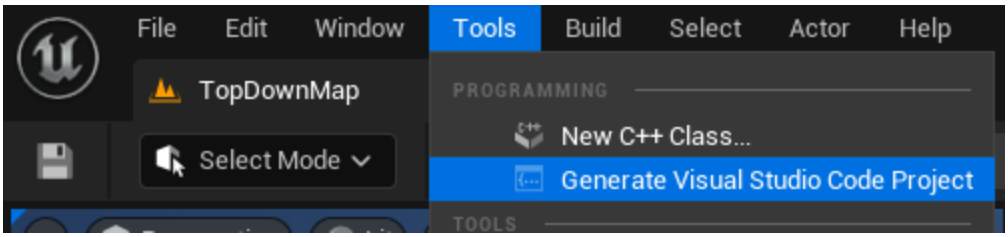
For details about how to set up these components, refer to Installing the Compiler Toolset.

1. If you are debugging on Mac or Linux, download and install the LLDB extension.
2. If you need to set VS Code as your default IDE, open **Unreal Editor** and go to **Edit** > **Editor Preferences** > **General** > **Source Code**, then set your **Source Code Editor** to **Visual Studio Code**. Restart the editor for the change to take effect. This is not necessary to generate a VS Code solution (see step 5c), but it becomes the default instead of Visual Studio.

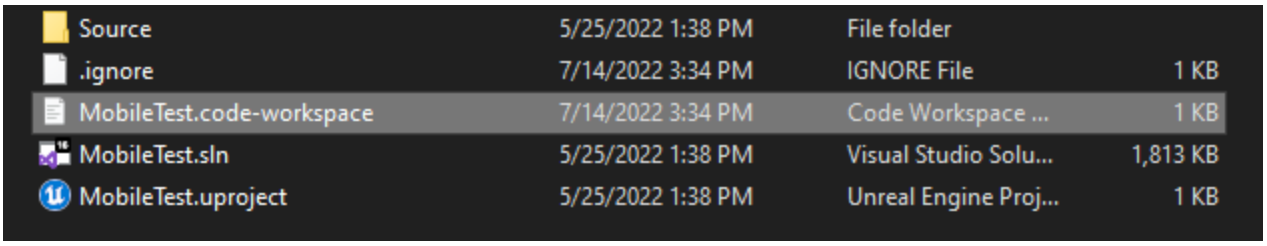


Click image to enlarge.

3. Generate your VS Code workspace. There are three ways you can do this:
    a. Open **Unreal Editor** and click **Tools** > **Refresh Visual Studio Code Project**.



    b. On Windows and Mac, right-click your project's `.uproject` file and click **Generate Project Files**. When complete, you should see a `.code-workspace` file in your project's folder.



c. In a command line, run `GenerateProjectFiles.bat -vscode`. Adding the `-vscode` parameter will make a `.vscode` workspace instead of a Visual Studio `.sln`. If you use this method, you do not need to change your default source code editor.

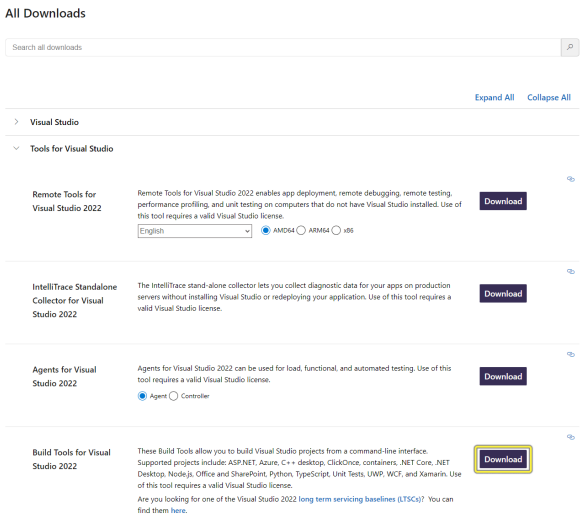
Installing the Compiler Toolset

Each desktop OS uses a different compiler toolset to compile projects in VS Code. The installation process for each one is straightforward, but each requires you to look in a different place to initiate setup.

Compiler Toolset for Windows

VS Code uses the Microsoft Visual C++ compiler (MSVC) toolset on Windows.

1. Open the Microsoft Visual Studio Downloads page.
2. Scroll down to the **All Downloads** section, then expand **Tools for Visual Studio**.
3. Click the **Download** button next to **Build Tools for Visual Studio 2022** and install it.



Click image to enlarge.

Compiler Toolset for MacOS

Unreal Engine uses LLVM/Clang as its compiler toolset on MacOS. Refer to Microsoft's documentation on Using Clang for Visual Studio Code for full instructions on how to install and enable it.

Compiler Toolset for Linux

Unreal Engine uses the LLVM/Clang toolset for Linux. To set it up, follow these steps:

1. Open a Terminal and run the following command:

```
sudo apt install clang
```

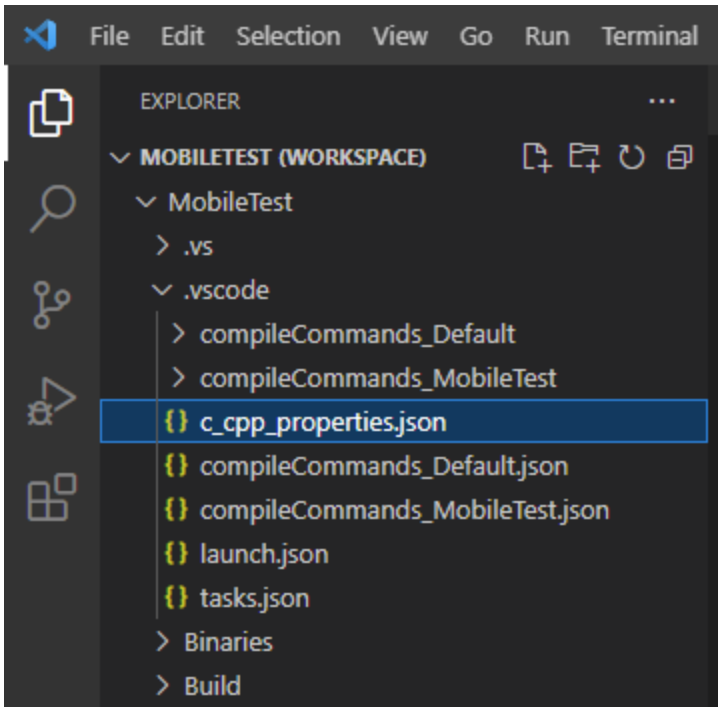
Copy full snippet

2. Run `SetupToolchain.sh`. This is located in your Engine's install directory under `Build/BatchFiles/Linux`.
3. Run `GenerateProjectFiles.sh` to build your VS Code workspace.

Setting Up IntelliSense for VS Code

VS Code can use **IntelliSense** for code hinting, but using it for Unreal Engine requires extra setup steps to expose your project's code to it.

1. In the **Explorer**, open `.vscode/c_cpp_properties.json`.



2. Add the `includePath` property as follows:

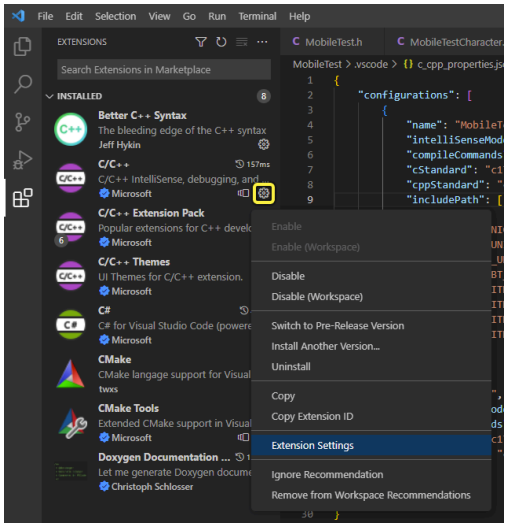
```
"includePath": [
"${workspaceFolder}\\Intermediate\\**",
"${workspaceFolder}\\Plugins\\**",
"${workspaceFolder}\\Source\\**"
],
```

Copy full snippet

This exposes these paths in your project to IntelliSense so it can discover your project's code.

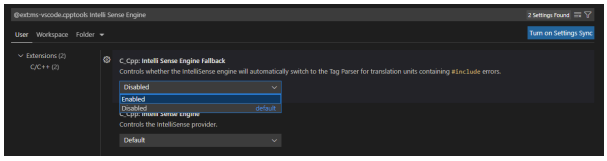
> `${workspaceFolder}` is not placeholder text. It is a shortcut indicating your workspace's current directory, which removes the need for absolute paths when editing these files.

3. Open the **Extensions** panel. Click the gear icon for the **C/C++** extension, then click **Extension Settings**.



Click image to enlarge.

4. Locate the entry for **C_Cpp: IntelliSense Engine Fallback**. Click the dropdown and set it to **Enabled**.



Click image to enlarge.

5. Set your **configuration** in the status tray to match the name of your configuration in `c_cpp_properties.json`.
6. You now see a small database icon in the VSCode status tray (blue bar) located in the lower-right side of the VS Code window. Mouse over this icon to see IntelliSense's progress parsing your project.

Example C/C++ Configurations File

The following is an example of a working `c_cpp_properties.json` file:

```
{
"configurations": [
{
"name": "CustomGame Editor Win64 Development",
"intelliSenseMode": "msvc-x64",
```

```
"compileCommands": "",
"cStandard": "c17",
"cppStandard": "c++17",
"includePath": [
"${workspaceFolder}\\Intermediate\\**",
"${workspaceFolder}\\Plugins\\**",
"${workspaceFolder}\\Source\\**"
],
"defines": [
"UNICODE",
"_UNICODE",
"__UNREAL__",
"UBT_COMPILED_PLATFORM=Windows",
"WITH_ENGINE=1",
"WITH_UNREAL_DEVELOPER_TOOLS=1",
"WITH_APPLICATION_CORE=1",
"WITH_COREUOBJECT=1"
]
}
],
"version":
}
```

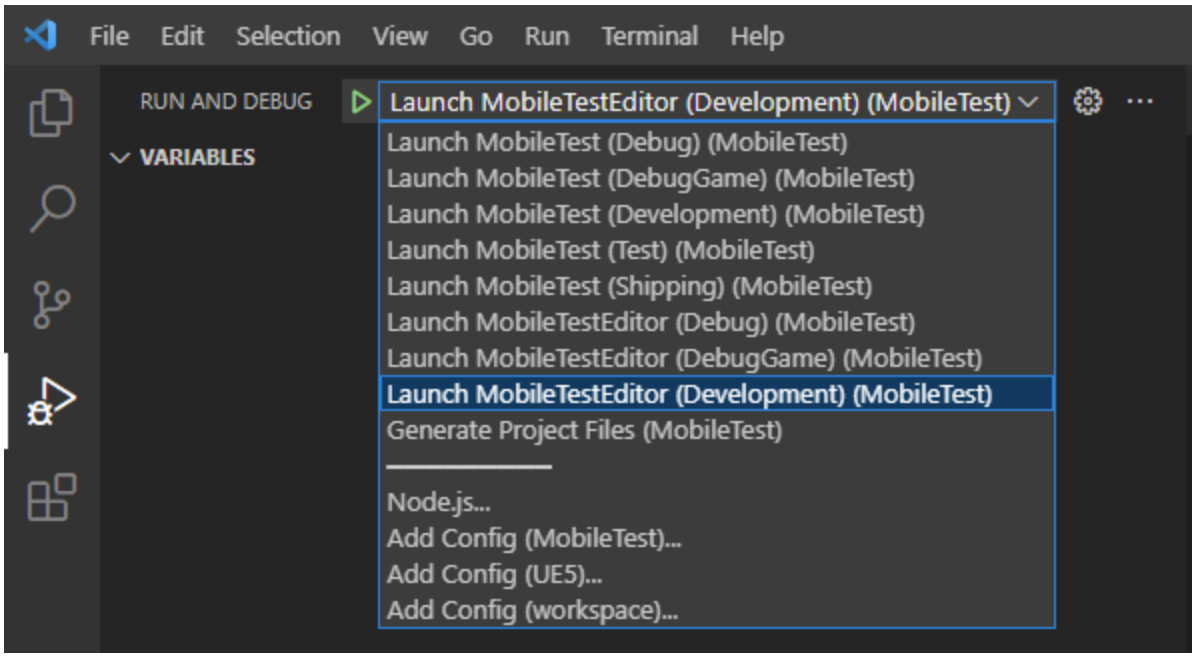
Copy full snippet

Once you are finished, IntelliSense provides code hinting for your project, including context-sensitive auto-complete functionality.

Building and Launching Projects in VS Code

Before you can build your project, make sure VS Code is set to the correct **build configuration**, otherwise it will attempt to build and run a standalone or shipping build of your game instead of the editor.

1. Open the **Run and Debug** panel by clicking the play button tab on the left side of the window, or by pressing **CTRL+SHIFT+D**.
2. Click the dropdown next to **RUN AND DEBUG** at the top of the panel. Choose the Development Editor configuration for your project. For example, if your project is named TestGame, you would choose **Launch TestGameEditor (Development) (TestGame)**.



Refer to the Build Configuration Reference page for more information about these options.

3. Click the **Play** button or press **F5** to start building your project in Editor mode. The project compiles, and when it finishes it opens Unreal Editor. Make sure you do not already have Unreal Editor open.

> After you choose your Build Configuration, you can press F5 or click **Run** > **Start Debugging** in the toolbar to build and debug your project regardless of whether you are using the Run and Debug panel.