



# **Cities: Skylines II Modding Instructions**

**June 9, 2025**

# Table of Contents

Introduction.....	1
Assumptions.....	1
Desktop Shortcut.....	1
Accounts.....	3
Paradox Account.....	3
GitHub Account.....	4
Modding Toolchain.....	5
Basic Mod Project.....	5
Choose Assembly Name.....	5
Create Mod Project.....	6
Configure Mod Project.....	6
Build Mod Project.....	9
Test Mod.....	9
Create Your Mod.....	10
User Interface.....	10
Harmony.....	12
Language Support.....	12
Developer Modes.....	13
Continue Last Save.....	14
Logging.....	14
Game Console.....	15
Open Mod Log File in VS or Notepad.....	15
Open Mod Log File in VS Code.....	15
Open Mod Log File in Notepad++.....	15
Log File Locations.....	16
Visual Studio Debugging.....	16
View Game Source Code.....	18
Get Game Source Code.....	18
Get UI Source Code.....	19
C# Language Version.....	20
Compiler Errors.....	20
Publish New Mod To Paradox Mods.....	20
Remove Unneeded Code.....	20
Add License File.....	21
Create Paradox Forum Thread.....	21
Update Project Properties.....	22
Prepare Publishing Files.....	22
Thumbnail Image File.....	22
Screenshot Files.....	22
Read Me File.....	23
Change Log File.....	24
Publish Configuration File.....	24
Prepare Solution.....	25
Publish To Paradox Mods.....	25
Verify Published Mod.....	26
Update Mod ID.....	27
Save to GitHub.....	27
Mod Is Published.....	29
Update Your Mod.....	29

Publish New Version To Paradox Mods.....	29
Update Project Properties.....	29
Update Publishing Files.....	30
Review Changes.....	30
Prepare, Publish, Verify Mod.....	30
Update GitHub.....	30
Update Paradox Forum Thread.....	31
Version Is Published.....	32
Update Published Configuration To Paradox Mods.....	32
Update Publishing Files.....	32
Review Changes.....	32
Prepare, Publish, Verify Mod.....	32
Update GitHub.....	32
Configuration Is Published.....	33
Game Version Updates.....	33
Get New Source Code.....	33
Update Modding Toolchain.....	33
Update User Interface.....	33
Delete Your Mod.....	34
Delete Mod From Paradox Mods.....	34
Hide GitHub Repository.....	34
Delete GitHub Repository.....	35
Update Paradox Forum Thread.....	35
Working With Markdown.....	35

# Introduction

This document contains very detailed step-by-step instructions for creating a Cities: Skylines II (CS2) code modification. A code modification is known as a “mod”. A code mod adds on to or modifies existing game play. This document is not for creating a map or other asset. Hereinafter, “mod” refers to the CS2 code mod being created.

There are ways to organize your code and do things other than the methods detailed in this document. Other mod authors will have different preferences on how to do things. You probably have your own preferences. But following these instructions will result in a basic working mod to serve as a starting point for adding your own functionality and making changes to suit your own style and preferences.

For more information on CS2 itself, see:

[https://cs2.paradoxwikis.com/Cities\\_Skylines\\_II\\_Wiki](https://cs2.paradoxwikis.com/Cities_Skylines_II_Wiki)

For more information on CS2 modding, see:

<https://cs2.paradoxwikis.com/Modding>

[https://cs2.paradoxwikis.com/Community-made\\_guides](https://cs2.paradoxwikis.com/Community-made_guides)

For help with CS2 modding, see the following, especially the section on Code Mods:

<https://discord.com/channels/1169011184557637825>

## Assumptions

This document assumes the following regarding your environment. If your environment is different, then you will need to make adjustments to the instructions in this document for your environment differences.

- CS2 was installed from Steam on a Windows PC. This implies you already have a Steam account.
- Development will be with Microsoft Visual Studio on a Windows PC. Author is using “Microsoft Visual Studio Community 2022 (64-bit)”, which is totally free.
- Mod source code will be stored on GitHub.
- No other mods are active. In the game, create and activate a new empty playset. This author uses Scene Explorer mod during development. But for your first mod, start with no mods active.

This document assumes you are familiar with the following. You do not need to be an expert. But at least some familiarity is very helpful. If you are unfamiliar with any of these, search the internet for more information.

- Using Microsoft Visual Studio.
- Programming in C#.
- Entity Component System (ECS) software architecture. For more information on ECS, see: [https://cs2.paradoxwikis.com/ECS - Entity Component System](https://cs2.paradoxwikis.com/ECS_-_Entity_Component_System) and <https://docs.unity3d.com/Manual/index.html> (search manual for “ECS”, in the search results click on “Entities”, then click on the latest documentation location for the entities package)
- For creating a user interface:
  - HyperText Markup Language (HTML).
  - Programming in TypeScript. TypeScript is a strongly typed programming language that builds on JavaScript. For more information on TypeScript, see: <https://www.typescriptlang.org/>

## Desktop Shortcut

For creating a basic mod as described in this document and during further mod development, you will be starting the game often. If the game installation did not create a shortcut on the desktop to start the game, you

can create your own shortcut. A desktop shortcut allows you to start the game without first starting Steam or the Paradox Launcher. This might be more convenient while developing your mod.

To create a shortcut on the desktop to start the game:

1. Right-click on the Windows desktop, click “New”, and click “Shortcut”. The “Create Shortcut” screen is displayed.
2. On the “Create Shortcut” screen:
  - a) For the location of the item, enter:  
`steam://rungameid/949230`
  - b) Click “Next”.
  - c) For the name of the shortcut, enter the following (no colon):  
`Cities Skylines II`
  - d) Click “Finish”.
  - e) The “Create Shortcut” screen is closed and the shortcut is created on the desktop.
3. Right-click on the shortcut and click “Properties”. The shortcut properties screen is displayed.
4. On the shortcut properties screen:
  - a) Click “Change Icon...”. The “Change Icon” screen is displayed.
  - b) On the “Change Icon” screen:
    - i. For the icon location file, enter or browse to:  
`C:\Program Files (x86)\Steam\steamapps\common\Cities Skylines II\Cities2.exe`
    - ii. The standard CS2 icon will be displayed. Click “OK” to use that icon.
    - iii. The “Change Icon” screen is closed and the shortcut properties screen shows the new icon.
  - c) Click “OK” to save the changed icon.
  - d) The shortcut properties screen is closed.
5. Test the shortcut by double-clicking on it. The Paradox Launcher is displayed for CS2 asking you whether to “Resume” or “Play”.
6. Close the Paradox Launcher screen.

You can use the shortcut this way and always choose whether to Resume (see also section “Continue Last Save”) or Play. To skip the Paradox Launcher and go directly into the game from your desktop shortcut, adjust the game’s launch options:

1. Open the Steam app.
2. Select Cities: Skylines II game.
3. Click on “Manage” (gear) and then “Properties”. The game’s properties are displayed, defaulted to the “General” properties.
4. On the “General” tab:
  - a) If the executable is not already present in the “Launch Options”, add it now, including the quotes:  
`"C:\Program Files (x86)\Steam\steamapps\common\Cities Skylines II\Cities2.exe"`
  - b) Update “Launch Options” to include the following launch option after the executable and after any other launch options (this launch option must always be last):  
`%command%`
  - c) If you add any other launch option, make sure it is between the executable and the “%command%” launch option.
5. Close the properties.
6. Test the shortcut by double-clicking on it. The game should start without first displaying the Paradox Launcher. Note that this and any other launch options will also be applied when starting the game from the Steam app.
7. In the game’s “Launch Options”, you can also set the “Selected Launch Option” to “Play Cities: Skylines II” to go directly into the game.

# Accounts

You will need a Paradox account and a GitHub account. These are described in the following sections. Even if you already have these accounts, read thru the sections anyway to complete any steps beyond just creating the accounts.

## Paradox Account

Paradox Mods is the only official place for CS2 mods. You should obtain CS2 mods only from Paradox Mods. You can browse for mods on Paradox Mods without an account. But a Paradox account is required to use a mod and to publish a mod.

The Paradox Mods website for CS2 mods is:

[https://mods.paradoxplaza.com/games/cities\\_skylines\\_2](https://mods.paradoxplaza.com/games/cities_skylines_2)

If you don't already have a Paradox account, create a new account now, which is totally free:

1. Start the game to the main menu.
2. On the right side of the screen, click on "Login". A login screen is displayed.
3. On the login screen, click on "Create Account". The "Create Paradox Account" screen is displayed.
4. On the "Create Paradox Account" screen, enter the requested information and then click on "Create Account". After the account is created, a confirmation message is displayed.
5. Click "OK" on the confirmation message. The game main menu is displayed.
6. Go to your email:
  - a) Find the message from Paradox Interactive. Be sure to check your spam for the message.
  - b) In the message, click on "Confirm Account" to confirm your account. A browser is opened showing that your account is now confirmed.
  - c) Close the browser.
7. On the main menu of the game, click on "Paradox Mods". It is okay if the button shows "Beta". Modding is considered to be in "Beta" testing status. The "Paradox Mods" screen is displayed.
8. On the "Paradox Mods" screen:
  - a) Click on "Me". A message will be displayed that a username is required.
  - b) Under the message, click on "Edit profile". A message is displayed asking permission to open a web browser. Click on "Yes" to allow the game to open the displayed link in a browser.
  - c) In the browser:
    - i. In the upper right, click on "Log in". A popup is displayed.
    - ii. In the popup, click on "Log In". The log in screen is displayed.
    - iii. Log in to your Paradox account with the email and password used to create your account. The "Choose username" screen is displayed.
    - iv. On the "Choose username" screen, enter a username. Choose your username wisely because it can be changed only by deleting and recreating your account. This author uses the same username as Steam and GitHub accounts. You might want to do the same (see section "GitHub Account").
    - v. Click on "Set username". The screen shows your mods, which will be none.
    - vi. If desired, click on "Edit profile" to change your avatar, bio, or external links.
    - vii. You may optionally link your Paradox account to other accounts like Steam, PlayStation, and Microsoft. In the upper right, click on your username and select "Account settings". The "Account settings" screen is displayed in a new browser. Under "Account Linking", click on "Link account" for the accounts you want to link.
    - viii. Log out of your Paradox account and close the browser.
  - d) In the game, close the "Paradox Mods" screen.
9. Click on "Paradox Mods" again. The "Paradox Mods" screen is displayed again.

10. On the “Paradox Mods” screen:
  - a) Verify that the “Me” tab now shows your profile information, including your username.
  - b) You now have full access to all the activities on the Paradox Mods screen. But importantly for this document, you will be able to publish your mod later.
  - c) Close the “Paradox Mods” screen.
11. Exit the game to desktop.

Log into your Paradox account in the game:

1. Start the game to the main menu.
2. If not already logged in, log in to your Paradox account in the game:
  - a) Under “Paradox Account”, click on “Login”. A log in screen is displayed.
  - b) On the log in screen, enter your Paradox account email address (not username) and password.
  - c) Click on “Login”. You are returned to the game main menu.
3. On the main menu under “Paradox Account”, verify your Paradox account username is displayed. If your username is not displayed, see the steps above for creating your username.
4. Once logged in, there is a “Logout” button to log out of your Paradox account. It is recommended to stay logged in to Paradox in the game. If you logout and log in again, all your subscribed mods will be downloaded and installed again.
5. Exit the game to desktop.

Create a Paradox account data file that will be used later when publishing your mod:

1. Use a text editor (e.g. Notepad) to create a new text file with 2 lines:
  - a) Line 1 contains your Paradox account username (not email address).
  - b) Line 2 contains your Paradox account password.
2. Choose a folder to store the file.
  - a) Do NOT save the file in your Visual Studio project folder. Place the file in a folder separate from your project.
    - i. This helps prevent the file from accidentally being included in the project and copied to GitHub for all to see.
    - ii. This makes the file easily reusable for multiple mods.
  - b) Possible file locations are your desktop or your Documents folder.
  - c) This author stores the file one folder above all the CS2 mod project folders.
3. Save the file with the name “pdx\_account.txt”.
4. Close the Paradox account data file.

## GitHub Account

Your source code must be available for others to review. GitHub seems to be preferred by most mod authors and is easily linked from Paradox Mods.

If you don’t already have a GitHub account, create a new account now, which is totally free:

1. In a browser, go to the GitHub web site:  
<https://github.com/>
2. Click on “Sign up”. The “Welcome” screen is displayed.
3. On the “Welcome” screen, follow the prompts to create a new account.
4. For the username, you might want to use the same username as your Steam and/or Paradox accounts.
5. Log out of GitHub.

In Visual Studio, set up access to GitHub:

1. Visual Studio includes Git support built directly into the IDE.
2. Start Visual Studio without code.
3. Verify that there is a “Git” top-level menu item.

4. Click on “Git” top-level menu item and select “Settings”. The “Options” screen is displayed defaulted to “Source Control” -> “Git Global Settings”.
5. In the “Git Global Settings”:
  - a) Set the “User name” and “Email” properties for your GitHub account.
  - b) Turn on “Commit changes after merge by default”.
6. Click “OK” to save the changes. The “Options” screen is closed.
7. Close Visual Studio.

## Modding Toolchain

CS2 uses the Modding Toolchain to enable the creation of mods. The Modding Toolchain is required to create a mod. Well, technically not *required*, but there is no reason to try to create a mod from scratch instead.

For more information on the Modding Toolchain, see:

[https://cs2.paradoxwikis.com/Modding\\_Toolchain](https://cs2.paradoxwikis.com/Modding_Toolchain)

Install the Modding Toolchain:

1. Start the game to the main menu.
2. From the game’s main menu, click on “Options”. The “Options” screen is displayed.
3. On the “Options” screen, on the left side, click on “Modding”. It is okay if the button shows “Beta”. The modding tools are considered to be in “Beta” testing status. The “Toolchain state” and “Dependencies” are displayed.
4. All of the listed “Dependencies” must be satisfied to create and publish a mod.
  - a) Satisfied dependencies are indicated as “Installed”, “Activated”, or “Detected” with a green check mark.
  - b) For each dependency not satisfied, click on the dependency to expand it and click on “Install” or “Update”. Follow any in-game instructions for the installation or update.
  - c) Some dependencies have a specific or minimum required version. If given a choice, be sure to install a compatible version.
  - d) For the Unity Editor installation, the default installation folder is “C:\Program Files\Unity”, but Unity will actually be installed in “C:\Program Files\Unity <version>” where “<version>” is the Unity version. This allows you to have more than one Unity version installed at the same time. Use Windows Apps management to uninstall previous Unity versions that are no longer needed.
  - e) For the IDE, your existing installation of Visual Studio should be found and marked as “Installed”.
5. Verify that the “Toolchain state” indicates “Installed” with a green check mark.
6. Close the “Options” screen and exit the game to desktop.

## Basic Mod Project

The following sections describe the steps for creating a basic Visual Studio project for your mod. Dependencies from the Modding Toolchain will be used, so be sure they are all satisfied before proceeding.

### Choose Assembly Name

The assembly name will be used for many things in the mod including: default namespace, installation folder name, DLL file name, Product name in the DLL file properties, etc.

Choose an assembly name for your mod:

1. Choose an assembly name that is one or a few words representing your mod. Keep it short.
2. Choose an assembly name that will not be too much like any other existing CS2 mod.
  - a) In a browser, go to the Paradox Mods web site for CS2:  
[https://mods.paradoxplaza.com/games/cities\\_skylines\\_2](https://mods.paradoxplaza.com/games/cities_skylines_2)



- b) Search for mods with similar names or features that are tagged “Code Mod”.
- 3. Choose the assembly name wisely.
  - a) After the mod project is created (see section “Create Mod Project”), it is possible but more difficult to change the assembly name.
  - b) The assembly name will likely be the basis for your mod title that users will see (see section “Configure Mod Project”).
- 4. Include only letters and digits.
- 5. Include no spaces, underscores, or special characters.
- 6. Capitalize only the first character of each word in the assembly name (i.e. Pascal Case).
- 7. Hereinafter, wherever “AssemblyName” is referenced, it means this chosen assembly name.

## Create Mod Project

Create the mod project in Visual Studio using the mod template:

1. Create a new project:
  - a) Start Visual Studio.
  - b) If the startup screen is displayed, click on “Create a new project”.
  - c) Otherwise, in the menu, click on “File” -> “New Project...”.
  - d) Either way, the “Create a new project” screen is displayed.
2. On the “Create a new project” screen:
  - a) The Modding Toolchain installs a template that makes it much easier to start a new mod. Search the templates for “cities skylines”.
  - b) Click on the template for “Cities Skylines II mod (Colossal Order Ltd)” to highlight it.
  - c) Click on “Next”. The “Configure your new project” screen is displayed.
3. On the “Configure your new project” screen:
  - a) For “Project name”, enter the AssemblyName chosen earlier.
  - b) For “Location”, set as needed. This author uses:  
C:\Users\<name>\Documents\Visual Studio Projects\Cities Skylines 2 Mods\My Mods
  - c) For “Solution” (if present), select “Create new solution”.
  - d) For “Place solution and project in the same directory”, set to checked or unchecked according to your preference. This author prefers solution and project in the same directory (i.e. checked).
  - e) Click on “Next”. The “Additional information” screen is displayed.
4. On the “Additional information” screen:
  - a) For “Include mod settings”:
    - i. It is easier to include settings code now and delete the settings code later if not needed than it is to include settings code later from scratch.
    - ii. If your mod will or might have settings in the game’s “Options” menu, set to checked.
    - iii. If your mod will not have settings in the game’s “Options” menu, set to unchecked.
  - b) For “Include key bindings”:
    - i. It is easier to include key bindings code now and delete the key bindings code later if not needed than it is to include key bindings code later from scratch.
    - ii. If your mod will or might have a key binding that you want the user to be able to set in the game’s “Options” menu, set to checked.
    - iii. Otherwise, set to unchecked.
  - c) For “Short description” and “Long description”, leave the default text. Descriptions will be updated later.
  - d) Click on “Create”. The mod solution and project are created and opened in Visual Studio.

## Configure Mod Project

Update the project properties:

1. In the Visual Studio menu, click on “Project” -> “AssemblyName Properties”. The project properties are opened.
2. In section “Build” -> “Errors and warnings”:
  - a) Under “Treat warnings as errors”, check “Instruct the compiler to treat warnings as errors”. This is an author preference, but is recommended.
3. In section “Package” -> “General”:
  - a) Set the “Title” to a human-friendly title of the mod. This usually is the AssemblyName with spaces between the words. The title will be seen by users. Do not include any double quotes (") in the title.
  - b) Set the “Package Version” to “0.1.0” for initial development.
  - c) Set the “Description” to a short one sentence description of the mod. This can be changed later. Do not include any double quotes (") in the description.
  - d) Set the “Copyright” to “Copyright © <year>” where <year> is the current year.
  - e) The “Assembly version” and “File version” both default to blank. Leave these blank so they automatically derive from the “Package Version” property, as desired.
  - f) The “Title”, “Description”, and “Version” will be used later for publishing the mod to Paradox Mods.
4. Save and close the project properties.

Make assembly info easily accessible as constant strings in the code:

1. Under the project, add a new folder named “ModAssemblyInfo”.
2. Copy file “ModAssemblyInfo.csproj” from GitHub:  
<https://github.com/rcav8tr/CS2-Modding-Instructions>
3. In Visual Studio, paste the file to the “ModAssemblyInfo” folder.
4. Click on the pasted “ModAssemblyInfo.csproj” file to view its properties.
5. In the properties for the “ModAssemblyInfo.csproj” file, verify “Build Action” is “None” and “Copy to Output Directory” is “Do not copy”. Set if needed.

Update the project file:

1. In the Visual Studio menu, click on “Project” -> “Edit Project File”. The project file is opened.
2. In the main “PropertyGroup”:
  - a) Verify that the “Title”, “Version”, “Description”, and “Copyright” are correct.
3. Replace two “TreatWarningsAsErrors” entries with one entry:
  - a) Find the two “PropertyGroup” entries for “TreatWarningsAsErrors”. There will be one for Debug and one for Release.
  - b) Copy the “TreatWarningsAsErrors” tag from one of the two property groups and paste it at the end of the main “PropertyGroup”. This makes it apply to both Debug and Release.
  - c) Delete both the Debug and Release property groups for “TreatWarningsAsErrors”.
4. Define PDX account data file path:
  - a) At the end of the main “PropertyGroup”, add an entry to define the PDX account data file that was created earlier. For example, if the file was saved on your desktop, the entry would be:  
`<PDXAccountDataPath>$(USERPROFILE)\Desktop\pdx_account.txt</PDXAccountDataPath>`
  - b) The value of “\$(USERPROFILE)” is typically:  
`C:\Users\<name>`  
 so if your file is stored elsewhere, adjust the whole path or just the path after “\$(USERPROFILE)”.
5. Automatically create the “ModAssemblyInfo.cs” file:
  - a) In the “ItemGroup” at the end of the file, delete the entry for the “ModAssemblyInfo.csproj” file that was automatically added by Visual Studio:  
`<None Include="ModAssemblyInfo\ModAssemblyInfo.csproj" />`
  - b) Add the following before the “</Project>” tag:  
`<!-- Automatically build ModAssemblyInfo.cs file. -->  
 <Import Project="ModAssemblyInfo\ModAssemblyInfo.csproj" />  
 <ItemGroup>`

```
<None Include="ModAssemblyInfo\ModAssemblyInfo.csproj" />
</ItemGroup>
```

- c) Save the project file. As soon as the project file is saved, Visual Studio immediately builds the “ModAssemblyInfo.cs” file in the “ModAssemblyInfo” folder.
  - i. Open the “ModAssemblyInfo.cs” file and verify its contents.
  - ii. In the file’s properties, verify the “Build Action” is “C# Compiler”. Set if needed.
  - iii. Close the “ModAssemblyInfo.cs” file.
6. Close the project file.

If settings code was included, configure it now:

1. Some of these things seem to be sort of a standard used by some other mod authors. The exact naming and structure may differ from one author to another.
2. Under the project, add a new folder named “ModSettings”.
3. Move the “Setting.cs” file from under the project to the new “ModSettings” folder.
  - a) If asked to confirm the move, click on “OK”.
  - b) If asked to “Adjust namespaces for moved files”:
    - i. This author prefers all code to be in one namespace, select “No”.
    - ii. If you prefer your namespaces to match your folders, select “Yes”.
  - c) If not asked to “Adjust namespaces for moved files”, you may need to adjust the namespaces manually to add or remove the “.ModSettings” suffix according to your preference.
4. Rename file “Setting.cs” to “ModSettings.cs”.
  - a) If asked “Would you also like to perform a rename in this project ...”, select “Yes”.
  - b) Otherwise, find all occurrences of the “Setting” class in the solution and change to “ModSettings”.
  - c) In the “ModSettings.cs” file, the default class properties provide examples for creating various types of settings (e.g. button, slider, checkbox, drop down, keyboard binding, etc.). For now, leave all the default class properties. Later, you can make changes for whatever settings your mod requires.
5. Separate the settings logic from the locale logic:
  - a) In the “ModSettings” folder, add a new C# class file named “LocaleEN.cs”. The file is opened.
  - b) Remove all the “using” statements. They should all be grayed out to indicate unused.
  - c) This author prefers all code to be in one namespace, so remove “.ModSettings” from the end of the namespace.
  - d) If you prefer your namespaces to match your folders, then leave the namespace unchanged.
  - e) Cut the “LocaleEN” class from the “ModSettings.cs” file and paste it to the new “LocaleEN.cs” file, replacing the default “LocaleEN” class.
  - f) In “ModSettings.cs”, remove unused “using” statements.
  - g) Save and close the “ModSettings.cs” file.
6. In the “LocaleEN.cs” file:
  - a) Add needed “using” statements to fix errors.
  - b) In the “ReadEntries” method:
    - i. In the entry for “m\_Setting.GetSettingsLocaleID()”, replace the quoted string of your AssemblyName (including the quotes) with “ModAssemblyInfo.Title” (without the quotes).
    - ii. If key bindings were included, in the entry for “m\_Setting.GetBindingMapLocaleID()”, replace the quoted string “Mod settings sample” with “ModAssemblyInfo.Title” (without the quotes).
    - iii. Note that if your mod title is too long, then it will be cut off when displayed on the game’s Options screen. If that happens, replace “ModAssemblyInfo.Title” with some abbreviated version of your mod title enclosed in quotes.
    - iv. If your mod will later support more than one language, then later you might want to use locale specific text instead of “ModAssemblyInfo.Title”.
  - c) Save and close the “LocaleEN.cs” file.
7. For more information on mod settings, see:  
[https://cs2.paradoxwikis.com/Options\\_UI](https://cs2.paradoxwikis.com/Options_UI)

# Build Mod Project

Verify solution builds successfully:

1. Save the entire solution.
2. Rebuild the entire solution.
3. Verify the build succeeded with no errors.

In Windows, verify the mod's files were created:

1. Go to the game's local mods folder:  
C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II\Mods
2. Verify there is a folder for the AssemblyName.
3. In the AssemblyName folder, verify the following files are present:  
AssemblyName.dll  
AssemblyName.pdb  
AssemblyName\_linux\_x86\_64.so  
AssemblyName\_mac\_x86\_64.bundle  
AssemblyName\_win\_x86\_64.dll  
AssemblyName\_win\_x86\_64.pdb
4. Right-click on the "AssemblyName.dll" file and select "Properties". The "Properties" screen is displayed.
5. On the "Properties" screen, click on the "Details" tab. The file's details are displayed.
6. On the "Details" tab, verify the project configuration settings are correctly represented in the following DLL details:
  - a) "File description" should be the AssemblyName.
  - b) "File version" should be "0.1.0.0".
  - c) "Product name" should be the AssemblyName.
  - d) "Product version" should "0.1.0".
  - e) "Copyright" should be the copyright text set earlier.
7. Close the "Properties" screen.

## Test Mod

If your mod includes settings, test your mod's settings:

1. Start the game to the main menu.
2. On the main menu, click on "Options". The "Options" screen is displayed.
3. On the "Options" screen:
  - a) On the left, verify there is an entry with your assembly title (not assembly name).
  - b) Click on the entry for your mod.
  - c) On the right:
    - i. Verify your mod's default options are displayed.
    - ii. If your mod includes key bindings, verify the mod's default key bindings are displayed.
    - iii. Your mod's options were generated by the logic in the "ModSettings" folder of your project.
  - d) Change the value of the Int slider.
  - e) In Windows:
    - i. Verify the mod settings file is present at:  
C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II\AssemblyName.coc
    - ii. Open the file and verify the int slider value is correct in the file.
    - iii. The settings file includes only settings that are different than the defaults. So if the file is absent or empty, it means all the settings have their default value.
    - iv. The folder also contains settings files for the game and any other mods.
4. Close the "Options" screen.

In Windows, verify your mod's log file:

1. If not already in the game, start the game to the main menu. This creates your mod's log file.
2. Exit the game to desktop.
3. Go to the game's logs folder:  
C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II\Logs
4. Verify there is a log file for your mod named "AssemblyName.log".
5. Open your mod's log file.
  - a) Verify there is an entry for "OnLoad" at the start.
  - b) Verify there is an entry for "OnDispose" at the end.
  - c) Close the log file.

**Congratulations!** Your mod was successfully loaded by the game and your mod's basic functionality is working!

## Create Your Mod

**Now go make your mod do what your mod must do!** To this basic mod project, add the specific functionality your mod will provide in the game. The following sections may be helpful during the development of your mod.

When you are ready to publish your new mod to Paradox Mods, continue with section "Publish New Mod To Paradox Mods".

## User Interface

Most mods will have some type of user interface (UI) to allow the user to interact with your mod and/or to display information to the user. An exception might be a mod that only changes game behavior and perhaps has only settings to change the mod's behavior. The Modding Toolchain includes a template for adding UI to a mod project.

For more information on UI modding, see:

[https://cs2.paradoxwikis.com/UI\\_Modding](https://cs2.paradoxwikis.com/UI_Modding)

To include a UI for your mod from the UI template:

1. Open your project in Visual Studio.
2. On the Visual Studio menus, select "Tools" -> "Command Line" -> "Developer Command Prompt". A new command prompt window is opened.
3. In the command prompt window:
  - a) Verify the current directory is set to your project folder. If not, change directory to your project folder.
  - b) Run the following command to add the UI logic to your mod project from the UI template:  
`npx create-csii-ui-mod`
  - c) If you get an error running the "npx" command, try uninstalling and reinstalling the "UI Mod Project Template" in the Modding Toolchain.
4. The npx command will prompt for some details:
  - a) For "Project name", enter "UI". This seems to be the standard used by other mod authors.
  - b) For "Author", enter your Paradox account username (not email).
  - c) Wait for the process to complete. It might take about 15-20 seconds.
  - d) Verify the command window shows "Project 'UI' created successfully".
  - e) Close the command prompt window.
5. Verify there is now a "UI" folder in the project. The folder name is the "Project name" entered above.
6. Exclude "node\_modules" folder:

- a) Expand the “UI” folder.
  - b) Notice there is a “node\_modules” folder that was created by the npx command. There are thousands of files in the “node\_modules” folder. This makes solution-wide source code searches take a long time because all these files are searched. The following steps will exclude the “node\_modules” folder.
  - c) In the Visual Studio menu, click on “Project” -> “Edit Project File”. The project file is opened.
  - d) At the end of the main “PropertyGroup”, add the following entry:  
`<DefaultItemExcludes>UI\node_modules\**;$ (DefaultItemExcludes)</DefaultItemExcludes>`
  - e) Save the project file.
  - f) Under the UI folder, notice that the “node\_modules” folder is no longer listed or if you have the “Show All Files” option turned on then the folder is shown as excluded. It might take a few seconds for the change to be shown.
7. Automatically build the “mod.json” file and automatically build the UI with the project:
- a) Expand the “UI” folder.
  - b) Open the “mod.json” file. In the “mod.json” file:
    - i. Notice that the file contains the “id” and “author” you entered earlier.
    - ii. Notice that the file defaults to version “1.0.0” which is different than your mod’s version of “0.1.0”.
  - c) Rather than having to remember to update this file every time some of the data is changed (especially the version), the file will be automatically rebuilt.
  - d) Copy the “mod.json.csproj” file from GitHub  
<https://github.com/rcav8tr/CS2-Modding-Instructions>
  - e) In Visual Studio, paste the file under the “UI” folder.
  - f) Return to the project file.
  - g) In the “ItemGroup” at the end of the file, delete the entry for the “mod.json.csproj” file that was automatically added by Visual Studio:  
`<None Include="UI\mod.json.csproj" />`
  - h) Add the following before the “</Project>” tag:  

```

<!-- Automatically build mod.json file. -->
<Import Project="UI\mod.json.csproj" />
<ItemGroup>
  <None Include="UI\mod.json.csproj" />
</ItemGroup>

<!-- Automatically build the UI with the project. -->
<Target Name="BuildUI" AfterTargets="AfterBuild">
  <Exec Command="npm run build" WorkingDirectory="$(ProjectDir)/UI" />
</Target>

```
  - i) Save the project file. As soon as the project file is saved, Visual Studio immediately builds the “mod.json” file, overwriting the existing file.
    - i. Return to the “mod.json” file.
    - ii. Verify:
      - “id” is AssemblyName;
      - “author” is your Paradox account username taken from your Paradox account data file;
      - “version” is “0.1.0” taken from the assembly version;
      - “dependencies” is “[]” (i.e. empty square brackets).
    - iii. Close the “mod.json” file.
  - j) Close the project file.
8. Rebuild the entire solution.
9. Go to the game’s local mods folder:  
 C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II\Mods
10. In the AssemblyName folder, verify the “AssemblyName.mjs” file is present.
11. To test that the UI in your mod is working, in section “Developer Modes”, see the steps to test UI developer mode.

# Harmony

Harmony is a library by Andreas Pardeike for patching, replacing, and decorating application methods during runtime. You can use Harmony to alter the functionality of all the available assemblies of that application.

Harmony gives you:

- A way to keep the original method intact
- Execute your code before and/or after the original method
- Modify the original with IL code processors
- Multiple Harmony patches co-exist and don't conflict with each other
- Works at runtime and does not touch any files

For more information on Harmony and how to use it, see:

<https://harmony.pardeike.net/>

<https://github.com/pardeike/Harmony>

Note that much of what Harmony was used for on Cities Skylines 1 can be accomplished with ECS on CS2.

To install the Harmony package in your project:

1. In Visual Studio menus, select “Tools” -> “NuGet Package Manager” -> “Manage NuGet Packages for Solution...”. The “Manage Packages for Solution” screen is displayed.
2. On the “Manage Packages for Solution” screen:
  - a) Click on “Browse”. A list of packages is displayed.
  - b) In the search box, enter “lib.harmony”. Matching packages are listed.
  - c) Click on the “Lib.Harmony” package to highlight it (not “Lib.Harmony.Ref” or “Lib.Harmony.Thin” or “CitiesHarmony.API”). The package’s information is displayed on the right side.
  - d) Place a check mark next to your project name.
  - e) For “Version”, select “2.2.2”. This seems to be the de facto standard version used by other CS2 mod authors, even though a later version of Harmony is available.
  - f) Verify the Author is “Andreas Pardeike”.
  - g) Click on “Install”.
  - h) If the “Preview Changes” screen is displayed, click on “Apply”.
  - i) Wait for the installation to complete.
  - j) Close the “Manage Packages for Solution” screen.
3. Under the project, expand “Dependencies”. Verify there is an entry for “Packages”.
4. Expand the “Packages” entry. Verify “Lib.Harmony (2.2.2)” is listed.
5. Rebuild the entire solution.
6. Go to the game’s local mods folder:  
C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II\Mods
7. In the AssemblyName folder, verify the “0Harmony.dll” file is present.
8. For the Harmony ID, this author uses the Paradox account username, followed by a period, followed by the AssemblyName. This should be enough to make the Harmony ID unique among all CS2 mods.

## Language Support

CS2 comes with multiple languages built in. If desired, your mod can also support multiple languages. If your mod supports any additional languages, it is recommended that your mod support all of the languages built into the base game.

The detailed steps for including language support are too much to include here. See my [Performance Monitor mod on GitHub](#) for an example of how to include language support in your mod with NO dependencies on other mods. See especially the “Localization” folder, the “UITranslationKey” class (two places), and the UI components in the “UI\src\components” folder where the translations are actually used.



Some authors get translations thru Crowdin. This author uses [Google translate](#) to translate from English to the other languages. For each text that needs to be translated:

1. Translate from English to the other language.
2. Use the “Swap languages” button (two arrows) to translate the other language back to English.
3. Verify that the same text as the original English is obtained.
4. If the translation back to English is different and unacceptable, try using synonyms or different English wording/grammar to get good translations into other languages.
5. When the translation is good, copy the translation from the other language and paste it into the “Localization\Translation.csv” file.

The mod title and description should be obtained from your translations for each language, not from the mod assembly info which does not change with the user’s language selection.

1. Open the “ModAssemblyInfo.csproj” file.
2. Remove the lines for “Title” and “Description”. Title and Description will no longer be obtained from the mod’s assembly info.
3. Save and close the “ModAssemblyInfo.csproj” file. Visual Studio automatically updates the “ModAssemblyInfo.cs” file.
4. In the “ModAssemblyInfo.cs” file, verify the title and description entries are removed.
5. Everywhere in your project that the mod assembly title or description are used, update to instead use your translations. For example:

```
string modTitle      = Translation.instance.Get(UITranslationKey.Title);  
string modDescription = Translation.instance.Get(UITranslationKey.Description);
```

## Developer Modes

The game includes two developer modes that may be useful while developing mods. Add the following to the game’s launch options to activate the two developer modes (see section “Desktop Shortcut” for how to adjust the game’s launch options):

```
-developerMode -uiDeveloperMode
```

The game’s developer mode screen is activated by the “-developerMode” launch option. Test developer mode:

1. Start the game to the main menu.
2. Press the “Tab” key. Verify that the developer mode screen is displayed.
3. The developer mode screen provides access to some of the game’s internal workings.
4. Press “Tab” again to close the developer mode screen.
5. For more information on developer mode, see:  
[https://cs2.paradoxwikis.com/Developer\\_mode](https://cs2.paradoxwikis.com/Developer_mode)

The game’s UI developer mode is activated by the “-uiDeveloperMode” launch option. Test UI developer mode:

1. With the game started, in a browser, go to the following address to access the game’s UI developer mode page:  
<http://localhost:9444>  
The “Inspectable pages” screen is displayed.
2. On the “Inspectable pages” screen, click on the “Coherent Labs” icon. The “DevTools” page is displayed. The “DevTools” page is available regardless of whether or not your mod has a UI.
3. Click on “Elements”.
  - a) Drill down into the game’s UI structure. This may help you determine where your mod may need to make changes to the game’s UI.
  - b) Hover your cursor over an element to highlight the element in the game. This is helpful to make sure you are finding exactly the right element.



- c) You can change an existing element's attributes to see in real time what the change looks like in the game. For example, change an element's size, position, color, or other attributes. This is helpful for fine tuning the UI for your own mod instead of continually making changes to the UI code, recompiling, and restarting the game to see the effect.
4. If your mod includes a UI:
  - a) Click on "Console".
  - b) Verify your mod's "Hello UI!" message is displayed. This console message was generated from your mod's default UI code at "UI\src\index.tsx", which includes "HelloWorldComponent".
  - c) You can write debug messages to the UI console from your UI code (using console.log method) and see the results on the "DevTools" page. Remember to remove debug logging later.
5. The "DevTools" page can be left active in your browser while exiting and restarting the game during development:
  - a) After exiting the game, the "DevTools" page will show a warning that the connection was closed. Leave the warning message displayed!
  - b) After restarting the game, click on the "Reconnect DevTools" button to reconnect to the game. Reconnecting also clears the console.

## Continue Last Save

While developing your mod, you will likely be exiting and restarting the game often. Each time you start the game to test your mod, you will likely need to start a new game or load a saved game. To save yourself the step of loading a saved game, add the following game launch option to instruct the game to automatically load the last saved game (see section "Desktop Shortcut" for how to change the game's launch options):

`-continuelastsave`

For unknown reasons, the continue last save feature does not always work. In that case, just load a saved game manually. Author also found that performing a manual save or letting the game perform an auto save helps to continue last save next time.

## Logging

Log messages can be a tool for debugging your mod (see also section "Visual Studio Debugging"). Logging can also be helpful if a user has an issue with your mod to see what your mod was doing or if your mod logged an error. See section "Log File Locations" for the locations of various log files.

Logging capabilities for your mod are included in the basic mod by the Modding Toolchain template. You write a message to your mod's log file by executing the statement:

`Mod.log.<method>`

where "<method>" is one of the following depending on the severity level of the message you want to write from least severe (lowest level) to most severe (highest level):

`Verbose, Trace, Debug, Info, Warn, Error, Critical, Fatal, Emergency`

Info is usually used for ordinary messages. Each method is overloaded to accept a variety of parameters. The overloads you will probably use most often accept a string or an exception.

You can control the minimum severity level of all messages that the game will write to all log files by adding the following launch option (see section "Desktop Shortcut" for how to change the game's launch options):

`-logsEffectiveness=<option>`

where <option> is one of the severity levels from above or "Disabled" or "All". The game will log all messages at the specified severity level or higher.

By default, the game will write all messages of severity level "Info" or higher if the launch option is not specified. One technique is to include "Debug" level messages at critical places in your code. Then if a user is having an issue, you can ask the user to include the "logsEffectiveness" launch option set to "Debug", run the

game to reproduce the issue, and then provide the mod log file to you. In this way, you can get additional information in the mod log file from that user without logging more than necessary for every user.

By default, the game will include a stack trace with all messages of severity level “Warn” or higher. You can adjust the severity level at which a stack trace is included by executing the following in your mod:

```
log.SetShowsStackTraceAboveLevels(Level.<level>);
```

where <level> is the minimum severity level for which you want a stack trace. This author prefers a stack trace for minimum severity level “Error”.

By default, the game will display no popup in the UI for severity level “Error” or higher messages because of the “.SetShowsErrorsInUI(false)” on the creation of the logger. To display a popup message in the UI for severity level “Error” or higher messages, change the parameter to true (this author’s preference).

The following sections describe some of the methods that you can use during development of your mod to view logged messages or your mod’s log file while the game is running. There are likely other methods used by other authors.

## Game Console

The game has a built-in console that can display log messages. Add the following launch option to enable the game’s console (see section “Desktop Shortcut” for how to change the game’s launch options):

```
-captureStdout=console
```

Test the launch option is working by starting the game. A “Cities: Skylines II” console window should be displayed and filled with log messages from the game and your mod.

This author does not use this method because it is too difficult to locate the mod’s messages among all the other messages.

## Open Mod Log File in VS or Notepad

You can open your mod log file in Visual Studio or Notepad. When VS or Notepad has the focus, and only when they have the focus, updates to your mod log file will be displayed.

This author does not use this method because it too inconvenient to have to give another window the focus to see updates to the log file.

## Open Mod Log File in VS Code

You can install VS Code from the following location and use it to view your mod log file:

<https://code.visualstudio.com/>

VS Code will automatically update its display even if it does not have the focus and VS Code nicely formats the log file. However, VS Code will not automatically scroll down to the last entry in the log file (TBD Perhaps there is a setting in VS Code to do this?) and VS Code uses a lot of memory to just display a text file.

This author does not use this method because it too inconvenient to manually scroll down to the last entry.

## Open Mod Log File in Notepad++

You can install Notepad++ from here:

<https://notepad-plus-plus.org/>

After opening your mod log file in Notepad++, click on the “Monitoring (tail -f)” (eyeball) button in the tool bar to turn on automatic updating of the display of your mod log file and automatic scrolling to the last entry. Notepad++ uses about 5% as much memory as VS Code.

This is the author's preferred method of viewing the mod log file while developing while the game is running.

## Log File Locations

Log files may be useful to help find where errors are occurring in your mod or in the game. Many log files are created. In the following, "<CS2DataFolder>" means:

C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II

Log file locations:

1. Game main log files (current and previous):  
    <CS2DataFolder>\Player.log  
    <CS2DataFolder>\Player-prev.log
2. Game other log files:  
    <CS2DataFolder>\Logs\Discord.log  
    <CS2DataFolder>\Logs\FileSystem.log  
    <CS2DataFolder>\Logs\InputManager.log  
    <CS2DataFolder>\Logs\Localization.log  
    <CS2DataFolder>\Logs\Modding.log  
    <CS2DataFolder>\Logs\PdxSdk.log  
    <CS2DataFolder>\Logs\Radio.log  
    <CS2DataFolder>\Logs\SceneFlow.log  
    <CS2DataFolder>\Logs\UI.log  
    (may be more from the game)
3. Your mod log file:  
    <CS2DataFolder>\Logs\AssemblyName.log  
    (may be more from other mods)

## Visual Studio Debugging

Much debugging and analysis can be accomplished by just logging messages from your mod. This section describes the steps to use Visual Studio for more in depth debugging. These steps are adapted from Paradox Wiki:

<https://cs2.paradoxwikis.com/Debugging>

Add the Unity workload to Visual Studio:

1. Close Visual Studio if it is open.
2. In Windows, use "Apps" to modify your Visual Studio installation. The "Visual Studio Installer" screen is displayed and within that screen the "Modifying" window is displayed defaulted to the "Workloads" tab.
3. On the "Workloads" tab, under the "Gaming" heading, find the entry for "Game development with Unity". If it is already checked, you already have the Unity workload installed. If it is not checked:
  - a) Check "Game development with Unity". The installation details on the right will be updated.
  - b) Click on "Modify" to install the workload.
  - c) Wait for the installation to complete.
4. Close the "Modifying" window and the "Visual Studio Installer" screen.

Configure Unity debugging:

1. In Windows, use File Explorer go to the Unity editor directory. If you installed Unity in a different directory, adjust as needed:  
    C:\Program Files\Unity <version>\Editor\Data\PlaybackEngines\windowsstandalonesupport\Variations\win64\_player\_development\_mono  
    where <version> is the Unity version.
2. Copy the "UnityPlayer.dll" file.

3. Go to the CS2 game directory:  
C:\Program Files (x86)\Steam\steamapps\common\Cities Skylines II
4. Rename the existing “UnityPlayer.dll” file to “UnityPlayerOriginal.dll”. This is so you can easily restore the original file later.
5. Paste the “UnityPlayer.dll” file in the CS2 game directory.
6. Go to the CS2 data directory:  
C:\Program Files (x86)\Steam\steamapps\common\Cities Skylines II\Cities2\_Data
7. Edit the “boot.config” file in a text editor (e.g. Notepad).
8. Add the following line to the end of the file:  
player-connection-debug=1
9. Save and close the “boot.config” file.

#### Attach Unity debugger in Visual Studio:

1. Start the game normally via Steam. The Paradox Wiki notes that starting from other than Steam might prevent the debugger from working.
2. Verify that the game screen shows “Development Build” in the lower right corner. This is due to the “UnityPlayer.dll” file that was copied.
3. Open your project in Visual Studio.
4. On the Visual Studio menus, select “Debug” -> “Attach Unity Debugger”. The “Select Unity Instance” window is displayed.
5. On the “Select Unity Instance” window, click on the “Cities: Skylines II” entry to highlight it (hopefully it is the only entry). Click on “OK”.
6. You can now set breakpoints and do other debugging tasks on your mod in Visual Studio.
7. To stop debugging, click on “Stop debugging” on the toolbar or in the Visual Studio menus click on “Debug” -> “Stop debugging”.
8. If you exit the game to desktop with the debugger attached, the debugger will stop automatically.

#### To see how a breakpoint works with the debugger:

1. Exit the game to desktop.
2. In your project in Visual Studio, edit “Mod.cs” to add the following class after the “Mod” class:  

```
partial class TestSystem : Game.UI.UISystemBase
{
    protected override void OnUpdate()
    {
        base.OnUpdate();
    }
}
```
3. Set a breakpoint on the “base.OnUpdate();” line.
4. In the “Mod” class, at the end of the “OnLoad” method, add the following:  

```
updateSystem.UpdateAt<TestSystem>(SystemUpdatePhase.UIUpdate);
```
5. Build the solution.
6. Start the game to the main menu.
7. In Visual Studio, attach the Unity debugger as described above.
8. Verify that the breakpoint is hit. The mod’s TestSystem.OnUpdate runs even in the game’s main menu.
9. In the game, notice that the UI is now frozen because of the breakpoint.
10. In Visual Studio, click on “Continue”. Notice that the breakpoint is immediately hit again because the OnUpdate method gets called every frame.
11. Remove the breakpoint and click on “Continue”.
12. In the game, notice that the UI is now responsive again.
13. Stop the debugger and exit the game to desktop.
14. Remove the code from “Mod.cs” that was added for testing breakpoints.

If a new version of Unity is installed later, remember to get the corresponding “UnityPlayer.dll” for that version.

## View Game Source Code

When creating a mod, it is often useful to view the game's source code to see what the game is doing, where your mod needs to make changes, or to find some example code to adapt for use in your own mod. There are a number of intermediate language decompilers available. This author uses ILSpy. ILSpy is a free, open-source .NET assembly browser and decompiler that allows you to view the structure of .NET assemblies and decompile them into C# code.

To install the ILSpy extension in Visual Studio:

1. In Visual Studio, in the menus, click on "Extensions" -> "Manage Extensions...". The "Extension Manager" screen is displayed.
2. On the "Extension Manager" screen, search for "ilspy". The search results are displayed.
3. In the search results, click on "ILSpy 2022" or the latest version and click on "Install". The installation is scheduled to run later.
4. Close the "Extension Manager" screen.
5. Close Visual Studio. The "VSIX Installer" screen is displayed.
6. On the "VSIX Installer" screen, click on "Modify" to install ILSpy. The extension gets installed.
7. Wait for the modifications to be complete. The "Modifications Complete" screen is displayed.
8. Close the "Modifications Complete" screen.

To view the game's source code in ILSpy:

1. Open your mod project in Visual Studio.
2. Under the project, expand "Dependencies" -> "Assemblies".
3. Right-click on "Game" and select "Open in ILSpy". ILSpy opens pointing to the "Game.dll" file.
4. Use ILSpy to expand the Game dll or use the search capabilities in ILSpy.
5. Close ILSpy.

## Get Game Source Code

Sometimes it is easier to view or search for source code if the game's source code is in Visual Studio.

To get the source code for the "Game.dll" file (assumes ILSpy extension is installed, see section "View Game Source Code"):

1. In Windows, either:
  - a) Delete all existing folders and files from the folder where the files will be stored.
  - b) Create a new folder to hold the files. Author uses:  
C:\Users\<name>\Documents\Visual Studio Projects\Cities Skylines 2 Mods\Base Game <version>  
where <version> is the game version, something like "1.2.3f1".
2. View the game source code in ILSpy as described in section "View Game Source Code".
3. In ILSpy:
  - a) Right-click on the "Game" assembly and select "Save Code...". A "Save As" dialog is opened.
  - b) Select the folder where the files should be stored.
  - c) Click on "Save".
  - d) Wait for the export to complete. The chosen folder should now have a "Game.csproj" file along with folders and source code files that were created from the "Game.dll" file.
  - e) Close ILSpy.

Configure the game source code:

1. Open the "Game.csproj" file in Visual Studio.
2. In the project explorer, expand "Dependencies" -> "Assemblies". If "netstandard" is not listed:
  - a) Right-click on "Assemblies" and select "Add Assembly Reference...". The "Reference Manager" dialog is opened.

- b) On the “Reference Manager” dialog:
  - i. On the left side, click on “Browse”.
  - ii. If “netstandard.dll” is displayed, click on it to place a check mark next to it.
  - iii. If “netstandard.dll” is not displayed, click on the “Browse...” button. A standard Windows file dialog is opened. On the Windows file dialog, find and select the “netstandard.dll” file and click on “Add”. The file is normally at:  
C:\Program Files (x86)\Steam\steamapps\common\Cities Skylines II\Cities2\_Data\Managed
  - iv. Click on the “OK” button. The dialog is closed.
- c) Verify the “netstandard” assembly is referenced in the “Assemblies” list.
- d) Collapse “Dependencies”.
3. The “MethodImplOptions.AggressiveInlining” attribute present on hundreds of methods cannot be resolved. To fix this and significantly reduce the number of errors:
  - a) Open any source code file in the project.
  - b) Enter Control+H or on the menu click “Edit” -> “Find and Replace” -> “Quick Replace”. The Quick Replace dialog is displayed.
  - c) On the Quick Replace dialog:
    - i. For the find text, enter “MethodImplOptions.AggressiveInlining”.
    - ii. For the replace text, enter “(MethodImplOptions)0x100 /\*AggressiveInlining\*/”.
    - iii. For the scope, select “Entire Solution”.
    - iv. Enter Control+A or click the “Replace all” button. Wait for the replaces to complete.
  - d) Close the source code file.
4. In the Visual Studio menu, click “File” -> “Save All”. The “File Save As” dialog is displayed to save the “Game.sln” file.
5. In the “File Save As” dialog, click “Save”. The solution file is saved.

The game source code is now ready to browse and search. For legal purposes, do not try to compile the game source code or distribute it.

## Get UI Source Code

It is easier to view or search the game’s UI source code if the UI source code is in a file.

To get the game’s UI source code for the “index.js” file:

1. Make sure the UI developer mode is activated as described in section “Developer Modes”.
2. Start the game to the main menu.
3. In a browser, go to the following address to access the game’s UI developer mode page:  
<http://localhost:9444>  
The “Inspectable pages” screen is displayed.
4. On the “Inspectable pages” screen:
  - a) Click on the “Coherent Labs” icon. The “DevTools” page is displayed.
  - b) On the “DevTools” page:
    - i. At the top, click on “Sources”. The sources are displayed.
    - ii. On the left in the list of sources, click on “index.js”. The source code for “index.js” is displayed.
    - iii. Notice that the source code is in “minified” format all on one line. Minifying takes the original source code and removes white space, comments, and other non-essential characters and replaces long variable names with arbitrary short names. This significantly reduces the file size which improves JavaScript performance.
    - iv. At the top, click on the “Pretty-print” button. The formatted source code for “index.js” is displayed. The variable names are still replaced, but now the code is spread over multiple lines and indented for easier viewing.
    - v. Copy and paste all of the formatted source code to a file. This author names the file “game index <version>.js” where <version> is the game version (e.g. “1.2.3f1”).

- vi. During development, this author keeps the file in the mod project for quick access.
5. Close the browser tab with the “Inspectable pages”.
6. Exit the game to desktop.

## C# Language Version

The basic mod project created from the template uses target framework “net472” which has a corresponding default C# language version of “7.3”. You might write code where C# generates an error indicating that some feature “is not available in C# 7.3” and that you should use a greater language version. According to another mod author, target framework “net472” and C# language version “9.0” should be used because those are the latest supported by Unity/ECS and higher versions definitely break things.

To use C# language version “9.0”:

1. Open your project in Visual Studio.
2. In the Visual Studio menu, click on “Project” -> “Edit Project File”. The project file is opened.
3. In the main “PropertyGroup”, below the “TargetFramework” entry, add the following new entry:  
`<LangVersion>9.0</LangVersion>`
4. Save and close the project file.
5. Verify your error is resolved.

## Compiler Errors

The following error (or similar) may be generated when compiling certain code:

Reference to type 'Span<>' claims it is defined in 'mscorlib', but it could not be found.

This can occur because by default the compiler will try to use the vanilla “mscorlib.dll” but the Unity version is slightly different. To resolve this error:

1. Open your project in Visual Studio.
2. In the Visual Studio menu, click on “Project” -> “Edit Project File”. The project file is opened.
3. In the “ItemGroup” block with the other references, add the following entry:  
`<Reference Include="mscorlib">  
 <Private>false</Private>  
 <HintPath>$(ManagedPath)\mscorlib.dll</HintPath>  
</Reference>`
4. Save and close the project file.
5. Compile and verify the error is resolved.

## Publish New Mod To Paradox Mods

After you make your mod do what your mod must do, you probably will want to publish your mod to Paradox Mods so that other users can subscribe to your awesome mod. The following sections describe the steps to publish your new mod to Paradox Mods for the first time.

### Remove Unneeded Code

If settings code (with or without key bindings) was initially included but your mod does not need settings now, remove the settings code:

1. Delete the “ModSettings” folder from the project.
2. Edit the “Mod.cs” file:
  - a) Remove all code related to mod settings.
  - b) Remove unused “using” statements.
  - c) Save and close the “Mod.cs” file.
3. Remove any other code related to settings from everywhere in your project.

If UI code was included but your mod does not need a UI now, remove the UI code:

1. In the Visual Studio menu, click on “Project” -> “Edit Project File”. The project file is opened.
2. At the end of the first “PropertyGroup”, delete the following entry that was previously added. If you specified other items to exclude, then delete only the “UI\node\_modules\\*\*\;” portion and leave the rest.  
`<DefaultItemExcludes>UI\node_modules\**\;$(DefaultItemExcludes)</DefaultItemExcludes>`
3. At the bottom of the file, delete the two sets of entries that were previously added for automatically building the UI with the project and for automatically building the “mod.json” file.
4. Delete the “UI” folder from the project.
5. Remove any other code related to UI from everywhere in your project.

If Harmony was included but your mod does not need Harmony now, remove Harmony:

1. In Visual Studio under the project, expand “Dependencies” and “Packages”.
2. Right-click on the “Lib.Harmony (2.2.2)” package and click on “Remove”.
3. Remove any other code related to Harmony from everywhere in your project.

## Add License File

A license tells others what they can and cannot do with your code. A license might also disclaim responsibility for any problem caused by your code. A license is optional, but recommended.

Determine which license type you want for your mod’s code. The “MIT License” is relatively simple and is used by this and some other CS2 mod authors. For more information on licenses, see:

<https://choosealicense.com/>

To add the MIT license:

1. Copy the “LICENSE.txt” file from GitHub  
<https://github.com/rcav8tr/CS2-Modding-Instructions>  
This is the file GitHub would add to your project later when creating a repository if your project does not already have a license file and you choose the MIT License.
2. In Visual Studio, paste the file directly under the project.
3. Click on the “LICENSE.txt” file to select it. Verify the “Build Action” is “None” and “Copy to Output Directory” is “Do not copy”. Adjust if needed.
4. Edit the “LICENSE.txt” file just pasted:
  - a) Change “[year]” to the current year.
  - b) Change “[fullname]”. Some mod authors seem to use their real name. This author uses the Paradox username. Any legal ramifications of using a Paradox username instead of a real name are unknown.
  - c) Save and close the “LICENSE.txt” file.

To add a different type of license, follow similar steps to add a “LICENSE.txt” file for that license type.

## Create Paradox Forum Thread

A Paradox Forum thread provides subscribers a place to ask questions, report problems, make suggestions, and discuss your mod. A Paradox Forum thread is optional, but recommended.

To create a Paradox Forum thread for your mod:

1. In a browser, go to the Paradox Forum web site for CS2:  
<https://forum.paradoxplaza.com/forum/forums/cities-skylines-2.1147/>
2. Log in to your Paradox account.
3. Click “Cities Skylines 2: User Mods”. The User Mods threads are displayed.
4. In the text box for “Post a new thread in this forum”, enter your mod’s title.
5. In the text box for “Message...”, enter something like “Discussion thread for the <your mod title> mod.” where “<your mod title>” is the title of your mod.



6. Click on “POST THREAD”. Wait for the thread to be created. The thread list is displayed.
7. Find your thread in the list and click on it. Verify your thread is displayed with your initial message.
8. Log out of your Paradox account.

## Update Project Properties

Update project properties:

1. In the Visual Studio menu, click on “Project” -> “AssemblyName Properties”. The project properties are opened.
2. Under “Package” -> “General”, set “Package Version” for your initial release:
  - a) This author prefers “1.0.0” for the initial release.
  - b) Some other mod authors use version “0.m.p” (where “m” is minor version number and “p” is patch number) for their initial release and subsequent releases until they determine that the mod is no longer in testing or release candidate status.
3. Make sure the “Title”, “Description”, and “Copyright” properties are correct.
4. Save and close the project properties.

## Prepare Publishing Files

The section describes the steps to prepare the publishing files.

### Thumbnail Image File

The mod template includes a default thumbnail image in the “Properties” folder in the “Thumbnail.png” file. This default image should be replaced so your mod has its own unique thumbnail. Create/update a new thumbnail image:

1. Use your own image creation/editing tools to create a new thumbnail image.
2. Your thumbnail image should be 500 x 500 pixels even though the default thumbnail image is 950 x 500 pixels. A bit larger or smaller is okay, but it should be square.
3. If the game is played at 1920 x 1080 (i.e. the most common display resolution), then on the “Paradox Mods” screen of the game, the thumbnail will be scaled down to about 220 x 220 (i.e. about 44%) on the “Browse” tab and about 128 x 128 (i.e. about 26%) when viewing the individual mod. The thumbnail will be larger or smaller if the game is played at a higher or lower resolution. As you are creating your thumbnail image, you can try viewing it at these smaller sizes to verify that it looks okay.
4. Keep the same file name and image type “Thumbnail.png”.
5. Replace the default file in the “Properties” folder with your own file.

### Screenshot Files

Screenshots are optional but recommended to show potential users what your mod will look like in the game. A picture of your mod in action is worth a lot of words to describe it. To add screenshots to your published mod:

1. If you set up Visual Studio debugging, restore the original “UnityPlayer.dll” file so that the “Development Build” text is not captured in your screenshot:
  - a) Exit the game to desktop.
  - b) Go to the CS2 game directory:  
C:\Program Files (x86)\Steam\steamapps\common\Cities Skylines II
  - c) Rename “UnityPlayer.dll” to “UnityPlayerDevelopment.dll”.
  - d) Rename “UnityPlayerOriginal.dll” to “UnityPlayer.dll”.
  - e) You can switch back and forth between original and development by renaming the files.
2. Under the project, in the “Properties” folder, create a new folder named “Screenshots”.
3. This author prefers to include the thumbnail image as the first screenshot because the thumbnail is not otherwise displayed on the [Paradox Mods web site](#) for the mod.
  - a) Copy the “Thumbnail.png” file from the Properties folder and paste it to the Screenshots folder.

- b) Rename the file in the Screenshots folder according to the file naming convention below. This author uses file name “10Thumbnail.png”.
4. Start the game to capture screenshots of your mod in action.
5. For each screenshot:
  - a) The Windows Snipping tool can be used to take a screenshot. The Snipping tool can be activated with WindowsKey+Shift+S. The Snipping tool places the snip in the Windows clipboard.
  - b) To make changes to the snip, use the Snipping tool’s Snip & Sketch tool or paste the snip into an image editor of your choice.
  - c) On Paradox Mods (both in game and web site), the screen shots are displayed in an area that has a width to height ratio of 1.778 (aka 16 x 9).
    - i. If you want your screenshot to fill that area without any blank space on the top/bottom or left/right sides, then your screenshot must also have this ratio.
    - ii. It is okay if your screenshot is not this ratio. Paradox Mods will automatically scale your image to fit in the area, but with blank space where needed.
    - iii. Your screenshot does not need to be the whole game screen. Your screenshot can be only part of the game screen. If you want your partial screenshot to fill the display area on Paradox Mods, then it still needs to be this ratio.
    - iv. Here some standard screen sizes that have this ratio:  
 2560 x 1440  
 1920 x 1080 (most common)  
 1600 x 900  
 1280 x 720  
 1024 x 576  
 800 x 450
  - d) Save the screen shot to the new “Screenshots” folder:
    - i. The file name should be a 2-digit number representing the screen shot order you want on Paradox Mods (e.g. “10”, “20”, etc.) followed by a descriptive name of the screenshot so you know what it is. Include no spaces in the file name. Counting by 10’s now allows you to later easily add new screenshots in between without needing to rename existing files (e.g. “05”, “15”, “25”, etc.).
    - ii. The file format should be png.
    - iii. The file size should be less than 2.1MB. If a file is too large, resize it to a lower resolution or capture only part of the screen.
  - e) In the properties for the screenshot file, verify “Build Action” is “None” and “Copy to Output Directory” is “Do not copy”. Set if needed.
6. Exit the game to desktop.
7. In the Visual Studio menu, click on “Project” -> “Edit Project File”. The project file is opened.
8. In the project file:
  - a) Delete the “ItemGroup” for the “Screenshots” folder, if present:
 

```
<ItemGroup>
  <Folder Include="Properties\Screenshots\" />
</ItemGroup>
```
  - b) Save and close the project file.

## Read Me File

The Read Me file will be used as the long description on Paradox Mods and will be displayed by default in GitHub. Create a Read Me file:

1. Under the project, add a new text file named “ReadMe.md”. The file is created and opened.
2. The file uses Markdown for formatting. See section “Working With Markdown”.
3. The standard based on looking at other mods seems to be to start with the mod title as a heading followed by the short description. Add the following to the “ReadMe.md” file:  
 # <title>

<description>

where <title> and <description> are the assembly “Title” and “Description”.

4. Below that, add your full description of the mod.
5. In the menus, select: “Edit” -> “Advanced” -> “Set End of Line Sequence”. If “CRLF” is not checked, click on it now.
6. Save and close the “ReadMe.md” file.
7. In the properties for the “ReadMe.md” file, verify “Build Action” is “None” and “Copy to Output Directory” is “Do not copy”. Set if needed.

## Change Log File

The Change Log file will be used as the change log on Paradox Mods. Create a Change Log file:

1. Under the project, add a new text file named “ChangeLog.md”. The file is created and opened.
2. The file uses Markdown for formatting. See section “Working With Markdown”.
3. Add one unordered list item to the file with the following text:  
- Initial release.
4. Paradox Mods will automatically display the change log with a heading that includes the mod version and release date, so there is no need to include those in your change log.
5. In the menus, select: “Edit” -> “Advanced” -> “Set End of Line Sequence”. If “CRLF” is not checked, click on it now.
6. Save and close the “ChangeLog.md” file.
7. In the properties for the “ChangeLog.md” file, verify “Build Action” is “None” and “Copy to Output Directory” is “Do not copy”. Set if needed.

## Publish Configuration File

The “PublishConfiguration.xml” file provides information about your mod to Paradox Mods. Automatically create the “PublishConfiguration.xml” file from assembly information:

1. Copy the “PublishConfiguration.csproj” file from GitHub  
<https://github.com/rcav8tr/CS2-Modding-Instructions>
2. In Visual Studio, paste the file to the “Properties” folder.
3. Edit the “PublishConfiguration.csproj” file just added:
  - a) Find the “ModId” entry in the CDATA section:
    - i. If publishing a new mod, the “ModId” value must be “0”, which is the default in the template.
    - ii. If publishing a new version or updating the published configuration, the “ModId” value must be the mod’s Paradox ID. Update if needed.
  - b) If you created a Paradox Forum thread:
    - i. Find the “ForumLink” entry in the CDATA section.
    - ii. In a browser, go to the Paradox Forum web site for CS2:  
<https://forum.paradoxplaza.com/forum/forums/cities-skylines-2.1147/>
    - iii. Log into your Paradox account.
    - iv. Find and open the thread for this mod.
    - v. Copy the address for your thread.
    - vi. Paste the address inside the quotes for the Value of the “ForumLink” entry.
    - vii. Log out of your Paradox account.
  - c) If your mod requires a specific game version:
    - i. Find the “GameVersion” entry.
    - ii. Update the Value of the entry to your required game version. Use an asterisk (\*) to accept any version in the position (e.g. “1.\*”, “1.1.\*”, “1.2.\*”, etc).
  - d) If your mod is dependent on any other mods (i.e. other mods that must be subscribed for your mod to work correctly):
    - i. Find the “Dependency” entry.
    - ii. Update the Id of the entry with the ID of the mod upon which your mod depends.

- iii. Repeat the “Dependency” entry for each dependent mod.
- e) If your mod requires a Down Loadable Content (DLC) package to be present:
  - i. Find the “RequiredDLC” entry.
  - ii. Change the Value of the entry to one of the supported values.
  - iii. Repeat the “RequiredDLC” entry for each DLC that is required.
- f) In the “ExternalLink” entry for GitHub:
  - i. Replace “<GitHubUsername>” with your GitHub username.
  - ii. The entry assumes your GitHub repository name will be your AssemblyName with a “CS2Mod-” prefix. This is the author’s preference. Update as needed.
- g) Add any other entries for external links. See the project file for the list of supported types.
- h) Save and close the “PublishConfiguration.csproj” file.
- 4. In the Visual Studio menu, click on “Project” -> “Edit Project File”. The project file is opened.
- 5. Find and delete the “ItemGroup” entry for the “PublishConfiguration.csproj” file that was automatically added by Visual Studio:
 

```
<None Include="Properties\PublishConfiguration.csproj" />
```
- 6. Add the following before the “</Project>” tag:
 

```
<!-- Automatically build PublishConfiguration.xml file. -->
<Import Project="Properties\PublishConfiguration.csproj" />
<ItemGroup>
  <None Include="Properties\PublishConfiguration.csproj" />
</ItemGroup>
```
- 7. Save the project file. As soon as the project file is saved, Visual Studio immediately builds the “PublishConfiguration.xml” file, overwriting the existing file.
- 8. Verify the “PublishConfiguration.xml” file:
  - a) Go to the “Properties” folder of the project:
  - b) Open the “PublishConfiguration.xml” file and verify its contents.
  - c) If you have screenshots, verify the screenshot file names are correct and in the correct order.
  - d) Close the file.

## Prepare Solution

Rebuild the solution:

- 1. Change the build configuration from “Debug” to “Release”.
- 2. Rebuild the entire solution by clicking on menu “Build” -> “Rebuild Solution”.
- 3. Test the mod one last time before publishing.
- 4. Exit the game to desktop.

## Publish To Paradox Mods

Publish the mod to Paradox Mods:

- 1. Mods are published from Visual Studio, not from within the game.
- 2. In Visual Studio, right-click on the project and select “Publish...”. The “Publish” screen is displayed.
- 3. On the “Publish” screen:
  - a) In the drop-down at the top next to the folder icon:
    - i. If publishing a new mod, select “PublishNewMod.pubxml”.
    - ii. If publishing a new version, select “PublishNewVersion.pubxml”.
    - iii. If updating published configuration, select “UpdatePublishedConfiguration.pubxml”, but first see the IMPORTANT note in section Update Published Configuration To Paradox Mods.
  - b) Click on “Publish”. Publishing is started, including a rebuild of the solution. Wait for publishing to complete.
  - c) Verify the “Publish: 1 succeeded, 0 failed, 0 skipped” message is displayed.
  - d) For any error, see Visual Studio’s Output log, correct the error, and try the publish again.

- i. If publishing a new version or updating the published configuration, check that the “ModId” value is correct in the “PublishConfiguration.csproj” file.
- e) Close the “Publish” screen.

## Verify Published Mod

When the solution was rebuilt and again when the mod was published, a local copy of the mod was created. Remove the local copy of the mod so it does not interfere with the published mod:

1. In Windows, go to the game’s local mods folder:  
C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II\Mods
2. Delete the folder for your mod.

A work in progress copy of your mod was created. Remove the work in progress copy of the mod so it does not interfere with the published mod:

1. In Windows, go to the game’s work in progress folder:  
C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II\cache\Mods\mods\_workInProgress
2. Open the contents of each folder to identify the folder for your mod. There may be only one folder.
3. Delete the folder for your mod.

Verify the mod on Paradox Mods:

1. Start the game to the main menu.
2. On the main menu, click on “Paradox Mods”. The “Paradox Mods” screen is displayed.
3. On the “Paradox Mods” screen:
  - a) If publishing a new mod:
    - i. Click on “Playsets” at the top. The “Playsets” screen is displayed.
    - ii. On the “Playsets” screen, create a new Playset or activate an existing Playset to which your mod will be added for testing.
    - iii. Click on “Browse” at the top. The “Browse” screen is displayed.
    - iv. Under “Filter” and “Tags”, check the “Code Mod” tag. This helps to avoid finding maps and other assets.
    - v. In the “Search mods...” box, search for your mod by name. A list of matching mods is displayed.
    - vi. Locate your mod and click on it to display its properties. Your mod’s properties are displayed.
    - vii. Click on “Add to Active Playset”. Wait for the installation to complete.
  - b) If publishing a new version:
    - i. Click on “Me” at the top. Your published mods are displayed.
    - ii. Locate your mod and click on it. Your mod’s properties are displayed.
    - iii. In the “Mod Info” section, verify the version number just published is displayed. For the new version’s mod properties to be displayed, you may need to add the mod to a playset and/or exit the game to desktop and restart the game a few times.
    - iv. In the “Mod Info” section, verify the “Last Updated” date/time is the current date/time.
    - v. Verify a “ChangeLog” section is present with an entry for the version number just published and with the formatted text from your “ChangeLog.md” file.
4. On the screen for your mod’s properties:
  - a) Verify all the screenshots are available and in the correct order.
  - b) Verify the long description is correct and is formatted as desired.
  - c) Verify the thumbnail image is correct.
  - d) Verify the mod title next to the thumbnail image is correct.
  - e) Verify the author is correct.
  - f) Verify the Tags are correct. At a minimum, the “Code Mod” tag should be present.

- g) Under the “More About This Mod” section, verify the external links (except GitHub) are correct by clicking on each one. If publishing a new mod, the GitHub link won’t work until the solution is saved to GitHub below.
- h) If publishing a new mod, there will be no “ChangeLog” section to verify.
- i) If a Paradox Forum thread was created and referenced in the publish configuration file, verify there is a “Discussion” section after the mod description with your initial thread text. Click on “Open Full Discussion” and verify your Paradox Forum thread is displayed in a browser.
- j) Under the “Mod Info” section, make a note of the mod’s ID number. It will be used in steps below.
5. If any of the above verifications fail, then after completing these steps, see section “Update Published Configuration To Paradox Mods”.
6. Close the Paradox Mods screen.
7. Start a new game or load a saved game.
8. Verify the subscribed mod works as expected.
9. If the mod does not work as expected:
  - a) Try exit to desktop and restart the game once more for the latest mod version to be used, especially after publishing a new version.
  - b) If you need to update the mod to make it work, then after completing these steps for a new mod, see section “Publish New Version To Paradox Mods”.
10. Exit the game to desktop.
11. Verify the mod is present in the game’s folder of subscribed mods:  
 C:\Users\<name>\AppData\LocalLow\Colossal Order\Cities Skylines II\cache\Mods\mods\_subscribed\<ModID>  
 where <ModID> is the mod’s ID obtained above followed by an underscore and another number. This second number seems to be the number of times the mod was updated on Paradox Mods.

## Update Mod ID

The mod ID is required to publish any future updates of the mod to Paradox Mods. Update the mod ID in the publish configuration:

1. Edit the “PublishConfiguration.csproj” file.
2. Replace the default “ModId” value of “0” with your mod’s ID value inside the quotes.
3. Save and close the project file. As soon as the project file is saved, Visual Studio immediately builds the “PublishConfiguration.xml” file, overwriting the existing file.
4. In the PublishConfiguration.xml file, verify the mod ID value is correct.

## Save to GitHub

For more information on creating a Git repository from Visual Studio, see:

<https://learn.microsoft.com/en-us/visualstudio/version-control/git-create-repository?view=vs-2022>

Save the solution to GitHub:

1. On the Visual Studio menu, click on “Git” -> “Create Git Repository...”. The “Create a Git repository” screen is displayed.
2. On the “Create a Git repository” screen:
  - a) Under “Push to a new remote”, click on “GitHub” to select it.
  - b) Under “Initialize a local Git repository”:
    - i. The “Local path” defaults to your local project folder. Leave as is.
    - ii. The “.gitignore template” defaults to “Default (VisualStudio)”. Leave as is.
    - iii. If you already added a license file, there will be no entry for “License template”.
    - iv. If you did not already add a license file, the “License template” defaults to “None”. If you don’t want a license file, leave the default of “None”. You can add a license file here by selecting a template from the “License template” list. Be aware that if you add the license file here, then after the repository is created in GitHub, you may need to update the file for the copyright year and/or your name. Then that file update will need to be pushed to GitHub. It is better to create

and update the license file before creating the GitHub repository. Click “Cancel” then see section “Add License File”.

- c) Under “Create a new GitHub repository”:
  - i. The “Account” should default to your GitHub account. If not, click “Sign in” and select “GitHub Account”. A GitHub authorization web page is displayed. Click “Authorize github”. After the authorization completes, return to Visual Studio.
  - ii. The “Owner” should default to your GitHub username. Leave as is.
  - iii. The “Repository name” should default to your AssemblyName. Prefix your AssemblyName with “CS2Mod-” (author preference). This prefix quickly and easily identifies this repository as a CS2 mod among any other repositories you might have in your GitHub account.
  - iv. The “Description” defaults to blank. Set this to “Cities Skylines 2 mod - ” followed by your mod’s assembly description. You may need to open your project file in a text editor (e.g. Notepad) to get the description.
  - v. The “visibility” defaults to “Private”. Set to “Public”.
- d) Under “Push your code to GitHub”, verify the address is correct.
- e) Click on “Create and Push”.
  - i. If notified that “Your solution contains files outside the solution folder ...”, click “Yes” to continue anyway.
  - ii. Wait for the repository to be created. It will take just a few seconds.
3. Return to the Solution Explorer and verify that all your source folders and files indicate locked (i.e. not changed).

Update the GitHub repository:

1. In a browser, go to the GitHub web site:  
<https://github.com/>
2. Sign in to your GitHub account.
3. Verify the new repository for your mod is present.
4. Click on the repository for your mod. The repository is displayed defaulted to “<>Code”.
5. On the right side, to the right of “About”, click on the gear icon. The “Edit repository details” screen is displayed.
6. On the “Edit repository details” screen:
  - a) Verify your mod description is correct.
  - b) Under “Topics”, add all the following entries:  
“cities-skylines2”  
“cities-skylines-2”  
“cities-skylines-2-mod”
  - c) Under “Include in the home page”, leave all settings checked (i.e. the default).
  - d) Click “Save changes”. GitHub returns to the “<>Code” page.
7. Verify all your source files are present.
8. Verify the contents of the “ReadMe.md” file are displayed below your source files with Markdown formatting applied.
9. Under “master”, click on “Tags” and then “View All Tags”. An empty tags list is displayed.
10. Click on “Create a new release”. The “Create a new release” screen is displayed.
11. On the “Create a new release” screen:
  - a) Click on “Choose a tag”. In the drop down, enter “v” followed by “M.m.p” where “M” is major version number, “m” is minor version number, and “p” is patch number (e.g. “v1.0.0”). Click “Create new tag: ... on publish”.
  - b) For “Release title”, enter “Original” (author preference).
  - c) Enter a description of the release if desired. This author leaves description blank because the title alone is usually sufficient to describe the release.
  - d) There are no binaries to attach.

- e) Uncheck “Set as a pre-release”.
  - f) Click on “Publish release”. The release and tag are created.
12. Verify the release and tag were created.
  13. Sign out of your GitHub account.

Verify the GitHub link on Paradox Mods:

1. Start the game to the main menu.
2. Click on “Paradox Mods”. The “Paradox Mods” screen is displayed.
3. Locate your mod and click on it to display its properties. Your mod’s properties are displayed.
4. Under the “More About This Mod” section, click on the GitHub link.
5. When asked to open a link in a web browser, click “Yes”.
6. Verify the link opens your GitHub repository in a browser.
7. Close the “Paradox Mods” screen.
8. Exit the game to desktop.

## Mod Is Published

**Your new mod is now published to Paradox Mods!** Wait for CS2 users to find your awesome mod!

Be sure to monitor your mod’s discussion and your Paradox Forum thread for problems, errors, suggestions, etc. If you don’t want to monitor your mod’s discussion from within the game, you can also access the Paradox Mods web site for CS2:

[https://mods.paradoxplaza.com/games/cities\\_skylines\\_2](https://mods.paradoxplaza.com/games/cities_skylines_2)

## Update Your Mod

Before starting to update your mod with added/changed/deleted features or bug fixes:

1. Change the project build configuration from “Release” back to “Debug”.
2. If using Visual Studio debugging, change to the development “UnityPlayer.dll” file.
3. Unsubscribe from your mod to avoid interference with the local mod during development:
  - a) In the game, on the “Paradox Mods” screen, create a new empty Playset just for testing. You might name it something like “Empty” or “No Mods”. The new Playset is activated by default.
  - b) Now you can make changes to your mod locally without the subscribed instance of your mod interfering.

**Now go update your mod to make your mod do what your mod must do!** When you are ready to publish your updated mod to Paradox Mods, continue with section “Publish New Version To Paradox Mods”.

## Publish New Version To Paradox Mods

This section describes the steps to publish your mod to Paradox Mods after it’s functionality is changed. If only the publish configuration and/or code comments are changed, then see section “Update Published Configuration To Paradox Mods”.

## Update Project Properties

Update project properties.

1. In the Visual Studio menu, click on “Project” -> “AssemblyName Properties”. The project properties are opened.
2. Under “Package” -> “General”, update “Package Version” as needed. Paradox Mods expects a new version number. Click on the question mark to get Microsoft best practices for “Package Version”.
3. Update the “Title”, “Description”, and “Copyright” properties if needed.
4. Save and close the project properties.



## Update Publishing Files

Update thumbnail file:

1. If needed, update the “Properties\Thumbnail.png” file.

Update screenshots:

1. Update existing screenshot files where needed.
2. Delete the files for any screenshots no longer needed from the “Properties/Screenshots” folder.
3. Add any new screenshots using the procedure described above for a new mod.

Update the “ReadMe.md” file.

1. Edit the “ReadMe.md” file.
2. Verify the title and description at the top of the “ReadMe.md” file matches the title and description from your assembly info.
3. Make any other changes needed to the file.
4. Save and close the “ReadMe.md” file.

Update the “ChangeLog.md” file:

1. Edit the “ChangeLog.md” file.
2. The file lists only the changes since the previous version (i.e. not a running log of all changes):
  - a) Delete the entire contents of the file.
  - b) Add a bulleted list of changes where each bullet (markdown: -) is a short description of the change.
  - c) Paradox Mods will automatically display the change log with a heading that includes the mod version and release date, so there is no need to include those in your change log.
3. Save and close the “ChangeLog.md” file.

Update the “PublishConfiguration.xml” file:

1. Delete the “PublishConfiguration.xml” file. Visual Studio will automatically rebuild the file.
2. Review the “PublishConfiguration.xml” file.
3. If any changes are needed for entries that are not automatically populated, edit the “PublishConfiguration.csproj” file and make the changes there.

## Review Changes

Review all changes before publishing:

1. In Visual Studio, on the menu, click “View” -> “Team Explorer”. The “Team Explorer” tab or screen is displayed.
2. In “Team Explorer”, click on “Git Changes”. The “Git Changes” tab or screen is displayed.
3. On “Git Changes”, review every change in every changed file to make sure nothing unexpected was changed.
4. Close “Team Explorer”.

## Prepare, Publish, Verify Mod

Prepare, publish, and verify the mod using the same steps as in sections “Prepare Solution”, “Publish To Paradox Mods”, and “Verify Published Mod” for a new mod. Follow the instructions for a new version, not a new mod, where the instructions are different.

## Update GitHub

Update the solution on GitHub:

1. In Visual Studio, return to “Git Changes”:

- a) In the message box, enter “Version” followed by space and “M.m.p” where “M” is major version number, “m” is minor version number, and “p” is patch number (e.g. “Version 1.2.3”).
- b) Click on the drop down arrow next to “Commit All” and select “Commit All and Push”. Sign in to GitHub if needed. Wait for the commit to complete.
2. Return to the solution explorer and verify that all the files indicate locked (i.e. not changed).
3. In a browser, go to the GitHub web site:  
<https://github.com/>
  - a) Sign in into your GitHub account.
  - b) Click on the repository for your mod. The repository is displayed defaulted to “<>Code”. Verify your changed source files are updated.
  - c) Click on “master”, click on “Tags” and then “View All Tags”. The list of tags is displayed.
  - d) Click on “Releases”. The list of releases is displayed.
  - e) Click on “Draft a new release”. The “Draft a new release” screen is displayed.
  - f) On the “Draft a new release” screen:
    - i. Click on “Choose a tag”. In the drop down, enter “v” followed by the same “M.m.p” as above for Version (e.g. “v1.2.3”). Click “Create new tag: <tag> on publish”.
    - ii. For “Release title”, enter a short description of the changes made to the mod in this version.
    - iii. Enter a description of the release if desired. This author leaves description blank because the title alone is usually sufficient to describe the release.
    - iv. There are no binaries to attach.
    - v. Uncheck “Set as a pre-release”.
    - vi. Check “Set as the latest release”.
    - vii. Click on “Publish release”. The release and tag are created.
  - g) Click on “Releases”. Verify the release was created.
  - h) Click on “Tags”. Verify the tag was created.
  - i) Sign out of your GitHub account.

## Update Paradox Forum Thread

If you created a Paradox Forum thread, update it now either in a browser or in the game (this author finds it easier in a browser than in the game):

1. In a browser:
  - a) Go to the Paradox Forum web site for CS2:  
<https://forum.paradoxplaza.com/forum/forums/cities-skylines-2.1147/>
  - b) Log in to your Paradox account.
  - c) In the upper right, click on your username and select “Watched threads”. You ARE watching the thread for your mod, right? Your watched threads are displayed.
  - d) Find and open the thread for this mod.
  - e) Click on “JUMP TO LATEST”. Create a new post with the mod version number in bold and a short description of the changes. You might want to use the same text as your “ChangeLog.md” file.
  - f) Do not include any formatting that is not supported when the posting will be viewed in the game. For example, links do not work when the posting is viewed in the game.
  - g) Log out of your Paradox account.
2. OR In the game:
  - a) Start the game to the main menu.
  - b) Click on “Paradox Mods”. The “Paradox Mods” screen is displayed.
  - c) Click on the “Me” button. All of your mods are displayed.
  - d) Locate this mod and click on it to display its properties.
  - e) Scroll down past your description to the “Discussion” section.
  - f) Create a new post with the mod version number in bold and a short description of the changes. You might want to use the same text as your “ChangeLog.md” file.
  - g) Close the “Paradox Mods” screen.

- h) Exit the game to desktop.

## Version Is Published

**Congratulations! Your new version is now published to Paradox Mods!** Continue to monitor your mod's discussion and your Paradox Forum thread.

## Update Published Configuration To Paradox Mods

This section describes the steps when only your mod configuration on Paradox Mods needs to be updated and/or only comments were updated in your source code. Sometimes it is desirable to update only comments in the source code and not wait until the next functional release before updating GitHub. If any functionality or assembly information (e.g. title, description, copyright, version, etc.) has changed, then see section “Publish New Version To Paradox Mods”.

While it is possible to update your published configuration on the Paradox Mods web site, it is recommended to make all your updates in your Visual Studio source files and then publish from Visual Studio. This keeps your Visual Studio project, your GitHub repository, and your Paradox Mods published configuration all in sync with each other.

**IMPORTANT:** For unknown reasons, publishing using “UpdatePublishedConfiguration.pubxml” described below does not work for this author. The Visual Studio Output window shows “Publish: 1 succeeded”, but the published configuration is not actually updated on Paradox Mods. Until this is resolved, instead of the steps below, use the steps in section “Publish New Version To Paradox Mods”. Change the version patch number so Paradox Mods detects that the version changed, which causes Paradox Mods to update the published configuration. Changing the version number has the unfortunate side effect that every subscriber will need to download the mod again when only the published configuration changed.

## Update Publishing Files

Update publishing files using the same steps as in section “Update Publishing Files” for a new version.

## Review Changes

Review all changes before publishing using the same steps as in section “Review Changes” for a new version. There should be changes only to publishing files and comments.

## Prepare, Publish, Verify Mod

Prepare, publish, and verify the mod using the same steps as in sections “Prepare Solution”, “Publish To Paradox Mods”, and “Verify Published Mod” for a new mod, except:

1. On the “Publish” screen, in the drop-down at the top, select “UpdatePublishedConfiguration.pubxml”.

## Update GitHub

Update the solution on GitHub using the same steps as in section “Update GitHub” for a new version, except:

1. Modify the “Version M.m.p” and “vM.m.p” release and tag labels (this is author's preference):
  - a) From GitHub for the repository, display all releases.
  - b) Note the latest release version and whether or not the version has a trailing letter. The latest version might not be first in the list.
  - c) At the end of the two version labels, append the next single letter a-z not previously used by the latest release on GitHub. For example, if the latest release is “v1.2.3”, then the two labels would be “Version 1.2.3a” and “v1.2.3a”. If the latest release is “v2.3.1c”, then the two labels would be “Version 2.3.1d” and “v2.3.1d”.

2. For “Release title” enter one of the following depending on what was changed:
  - a) “Update Only Published Configuration”
  - b) “Update Only Comments”
  - c) “Update Only Published Configuration and Comments”

## Configuration Is Published

**Congratulations! Your configuration is now published to Paradox Mods!** Continue to monitor your mod’s discussion and your Paradox Forum thread.

## Game Version Updates

This section describes things you may need to do when the game version is updated. The game version is displayed in the lower left corner of the screen when on the main menu. This is not an all inclusive list.

## Get New Source Code

Get new game source code as described in section “Get Game Source Code”.

Get new UI source code as described in section “Get UI Source Code”.

## Update Modding Toolchain

See section “Modding Toolchain” for details on installing and updating the Modding Toolchain.

1. For each dependency that is not up to date:
  - a) Click on the dependency to expand it.
  - b) Click on “Update”.
  - c) Follow any in-game instructions for the update.
2. Make sure all dependencies are updated before proceeding

## Update User Interface

If your mod includes a UI, the UI code may need to be updated as described in this section.

To update the “UI\types” folder:

1. Open your project in Visual Studio.
2. On the Visual Studio menus, select “Tools” -> “Command Line” -> “Developer Command Prompt”. A new command prompt window is opened.
3. In the command prompt window:
  - a) Verify the current directory is set to your project folder. If not, change directory to your project folder.
  - b) Change directory to the “UI” folder of your project.
  - c) Run the following command to update the “UI\types” folder to the latest for the game:  
`npx create-csii-ui-mod update`
  - d) Verify the command window shows “Project files updated successfully.”.
4. Close the command prompt window.

The “UI\node\_modules” folder in your project may have differences compared to a newly created mod project with UI added to it.

1. If your mod project compiles successfully and your mod’s UI works with the current code in the “UI\node\_modules” folder:
  - a) The differences can be ignored.
2. If your mod project fails to compile or your mod’s UI does not work with the current code in the “UI\node\_modules” folder:

- a) TBD Not sure if these steps will resolve the issue. This has not been tested. You might want to make a copy of your original folder and file that you can restore if these steps cause a problem.
- b) Create a new basic mod project as described in section “Basic Mod Project”.
- c) To that newly created mod project, add a UI as described in section “User Interface”.
- d) In your mod project, replace the “UI\node\_modules” folder and the “UI\package-lock.json” file with the same from the newly created mod project.

After completing the above:

1. Start the game to the main menu.
2. Click on “Paradox Mods”.
3. Activate a playset that does not include a published version of your mod. This author maintains an empty playset for this purpose.
4. Exit the game to desktop.
5. Rebuild your solution and ensure there are no errors.
6. Start the game to the main menu.
7. Load a saved game or start a new game.
8. Ensure your mod’s UI works as expected.
9. Publish your updated mod as described in section “Publish New Version To Paradox Mods”.

## Delete Your Mod

You may want to delete your mod at some time. For example, your mod is obsolete because a game update duplicates the functionality of your mod. Or a game update breaks your mod and you decide not to fix your mod (you really should try to fix your mod). Regardless of the reason, this section describes the steps to delete your mod.

## Delete Mod From Paradox Mods

To delete your mod from Paradox mods:

1. In a browser, go to the Paradox Mods web site:  
<https://mods.paradoxplaza.com>
2. Log in to your account.
3. Click on your username at the top right to display the drop down. Click on “My uploaded mods”. Your uploaded mods are displayed.
4. Locate the mod you want to delete.
5. Hover your cursor over the mod. Buttons will be displayed.
6. Hover your cursor over the edit/pencil button. Options will be displayed.
7. Click on the “Delete mod” option. A confirmation message will be displayed.
8. On the confirmation message, click “OK”. Your mod is deleted.

## Hide GitHub Repository

You might want to keep your GitHub repository to reference later, but hide the repository from public view. If instead you want to delete your GitHub repository, then see section “Delete GitHub Repository”.

To hide your repository on GitHub:

1. In a browser, go to the GitHub web site:  
<https://github.com/>
2. Sign in to your GitHub account.
3. Find and open the repository you want to hide.
4. Click on the “Settings” button for the repository (not the gear icon next to About). Your repository settings are displayed.
5. Scroll down to the bottom of the settings to the “Danger Zone”.

6. Click on “Change visibility” and then click on “Change to private”.
7. Perform all the steps to confirm the change to private.
8. Scroll down to the “Danger Zone” again and verify the repository is currently private.
9. Sign out of your GitHub account.

## Delete GitHub Repository

If you do not want to keep your GitHub repository, then you should delete it. If instead you want to keep your GitHub repository but hide the repository from public view, then see section “Hide GitHub Repository”.

To delete your repository from GitHub:

1. In a browser, go to the GitHub web site:  
<https://github.com/>
2. Sign in to your GitHub account.
3. Find and open the repository you want to delete.
4. Click on the “Settings” button for the repository (not the gear icon next to About). Your repository settings are displayed.
5. Scroll down to the bottom of the settings to the “Danger Zone”.
6. Click on “Delete this repository”.
7. Perform all the steps to confirm the deletion.
8. Verify the repository is deleted.
9. Sign out of your GitHub account.

## Update Paradox Forum Thread

It seems there is no way to delete a Paradox Forum thread.

If you created a Paradox Forum thread for your mod, update it now:

1. In a browser, go to the Paradox Forum web site for CS2:  
<https://forum.paradoxplaza.com/forum/forums/cities-skylines-2.1147/>
2. Log in to your Paradox account.
3. Find and open the thread for your mod.
4. Create a new post:
  - a) Explain why your mod was deleted.
  - b) Include any other information you want your previous subscribers to know.
5. Log out of your Paradox account.

## Working With Markdown

The ReadMe.md and ChangeLog.md files use Markdown (md) for text formatting.

A note on the [Modding Toolchain web site](#) indicates that only headings (#, ##, ###), bullet lists (-), and bold text (\*\*text\*\*) are supported. TBD But may want to try other markdown anyway.

Do not include any ampersands (&) in the markdown. Ampersands in the “PublishConfiguration.xml” file will interfere with publishing.

Markdown references:

- <https://www.markdownguide.org/cheat-sheet/>
- <https://commonmark.org/help/>

Markdown testing tool where you can paste your markdown and see what it will look like when formatted:

- <https://spec.commonmark.org/dingus/>

\*\*\*