Official Modding Documentation for Warhammer 40,000: Space Marine 2

Intro

Space Marine 2 supports mods for PC. Players can use our modding editor to create custom content for the game.

The modding documentation is a work in progress. See also additional sources of information.

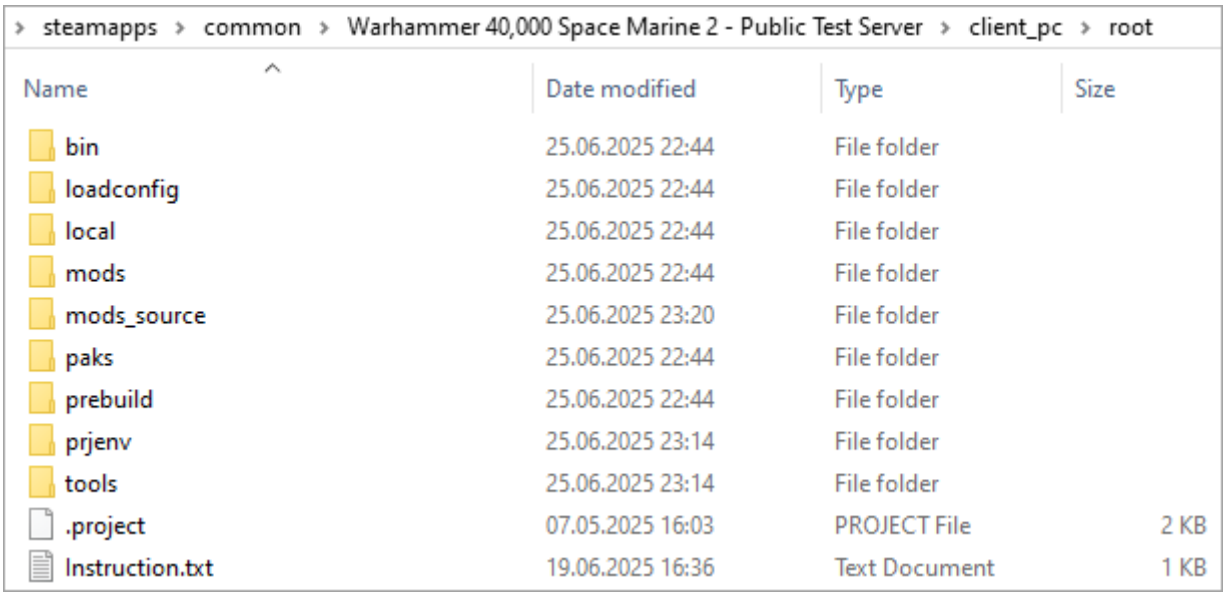
Make sure to use the Mod Editor version that matches your game instance. Backward compatibility is not guaranteed.

Installation

Install Mod Editor

1. Copy the contents of the `Mod Editor` folder to `<path_to_game>\client_pc\root`, where `<path_to_game>` is the full path to your local `Warhammer 40,000 Space Marine 2` folder.
2. Extract the client and server source files from `.pak` archives specified below to `<path_to_game>\client_pc\root\mods_source\`. If prompted, replace or skip the duplicate files. (!) Any alterations to this procedure may break resource generation.
   - Client files: `<path_to_game>\client_pc\root\paks\client\default\default_other.pak`
   - Server files: `<path_to_game>\server_pc\root\paks\server\default\default_other.pak`

The expected content of the `root` folder:



Install Extra Tools

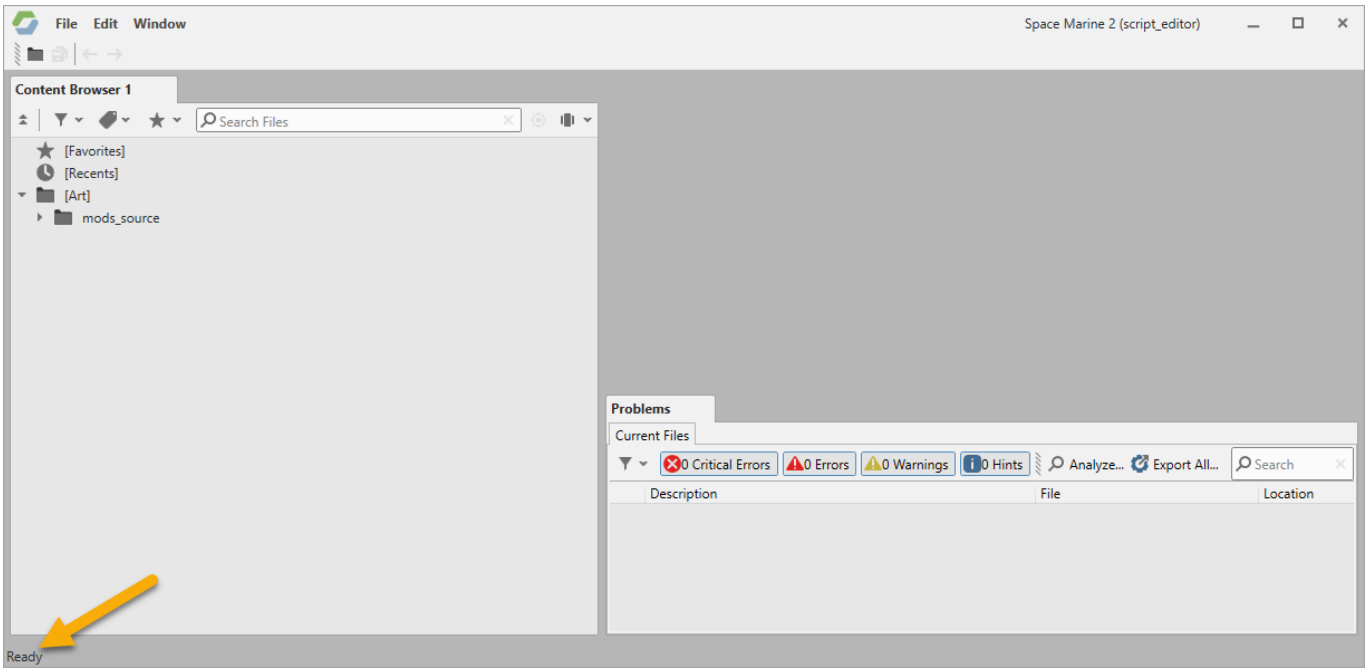
The `mod_tools` archive also contains the `Tools` folder with handy tools you can use to make mod creation easier. Follow the instructions in this folder, if available (e.g. Model Converter Reference in the `Tutorials` subfolder).

For advanced crash reporting, copy all files from the `Toolset\CrashReporter` folder to `<path_to_game>\client_pc\root\bin\pc` (when prompted, replace the default files). Whenever your game crashes during debugging, you'll get more insights into potential causes of the problem. You can use this information for troubleshooting or when requesting support via the unofficial Modding Hub on Discord.

Launch

The Mod Editor's executable is named **IntegrationStudio** – launch it via `<path_to_game>\client_pc\root\tools\ModEditor\IntegrationStudio.exe`. If prompted, allow the program to access the network.

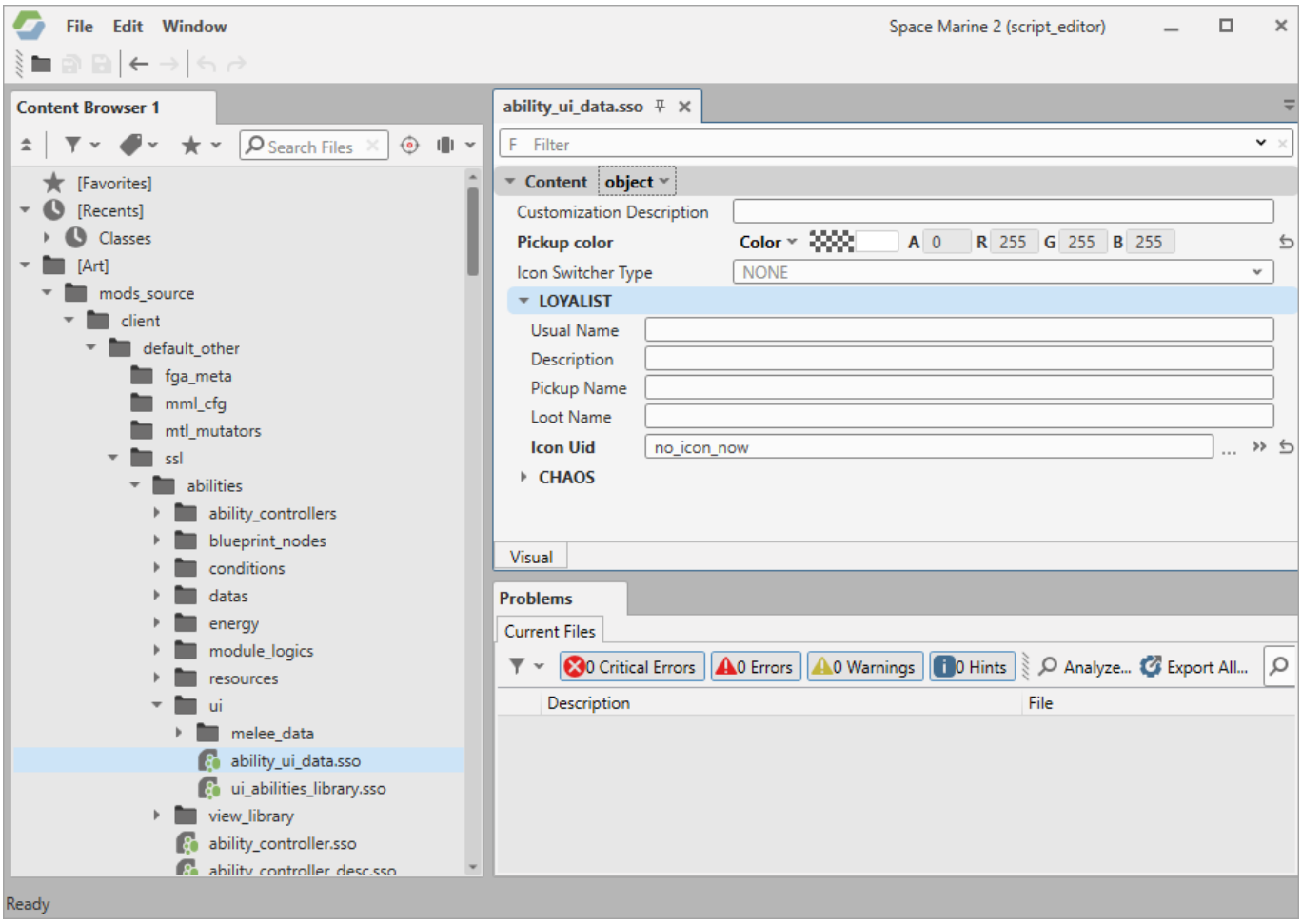
Wait until all required sources are loaded: the progress bar is in the bottom left corner.



Interface

The main editor window contains the following elements:

- **Top menu**:
  - **File** – main operations with files. Use **File > Open file in project** to browse all project source files available for editing. (!) Make sure to open only files from the `mods_source` folder; opening any external files may cause various problems, such as the failure to save your changes.
  - **Edit** – basic editing operations and editor preferences.
  - **Window** – open available editor tabs or navigate between them.
- **Content Browser** – an interactive tree of all project files. Search and tag files, add them to favorites, and more. Hover over buttons on the top panel to see tooltips.
- **File editor** – settings of files you open from the Content Browser.
- **Problems** – an interactive log of current issues.
- **Progress bar** – shows the current status of an operation performed by IntegrationStudio.



Quick Start: Your First Mod

Let's dive in and change something in the game:

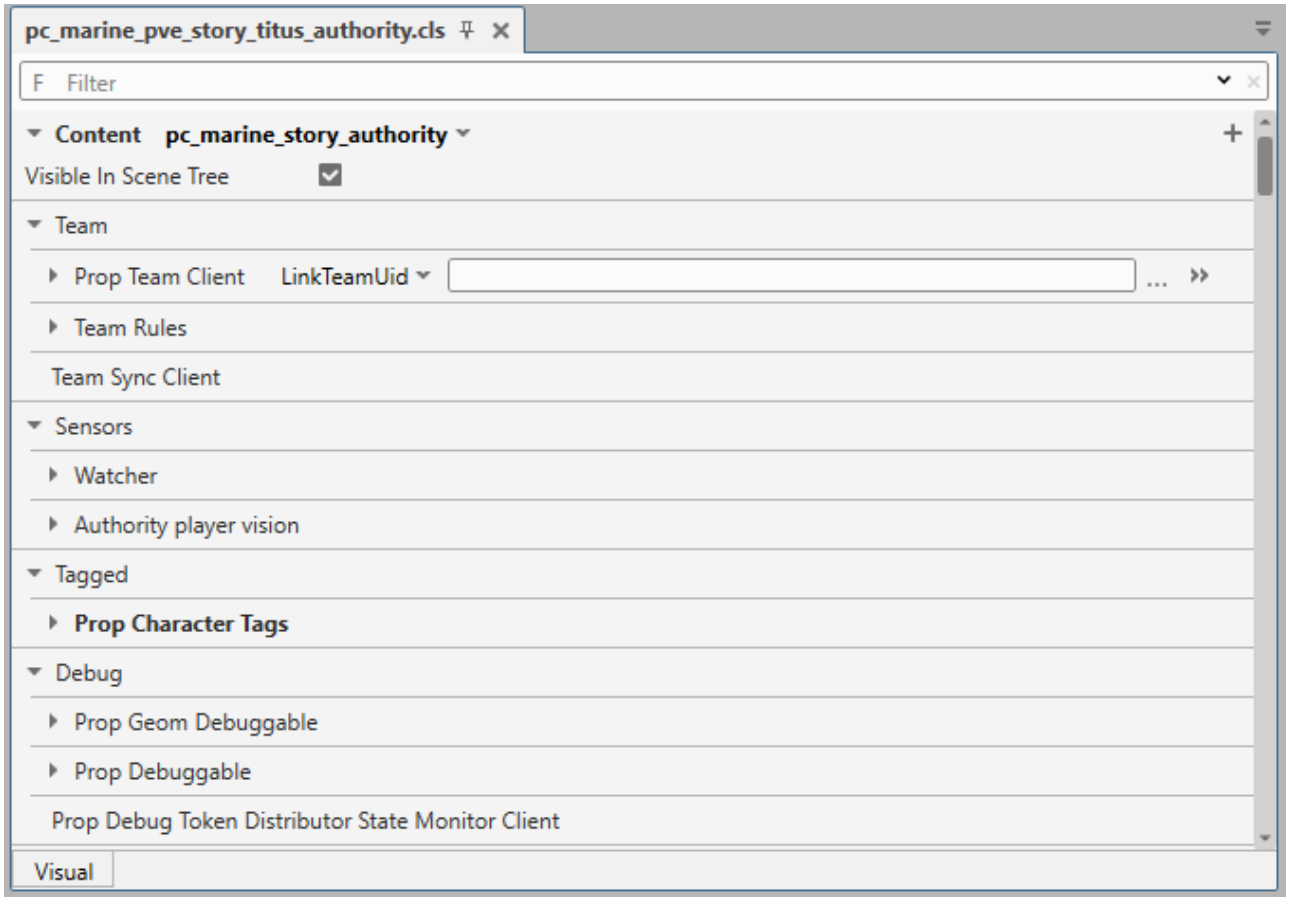
1. Press `Alt+Shift+O` to open the file browser.

2. Search for `pc marine titus authority` and double-click the `pc_marine_pve_story_titus_authority.cls` file.

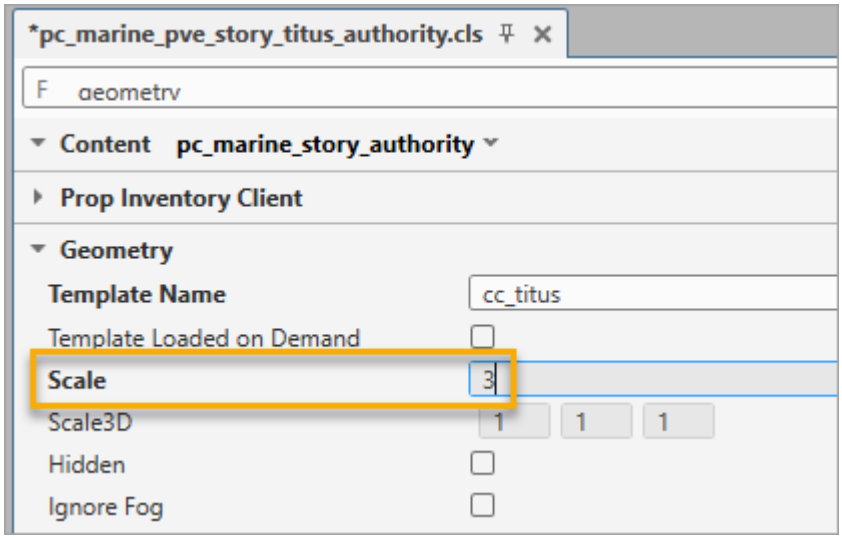
   > **Tip**: See the `authority` part in the filename? **Authority** refers to the current player – you, while **Client** refers to any other player. For another player, they are the Authority, and you are a Client.

   This will open the CLS file for the Titus actor. An **actor** is a game entity that has logic assigned to it: a player, a barrel, and so on. Every CLS contains a set of **properties** that define its actor's logic. For example, the `Geometry` property determines an actor's appearance. Changing it will have an immediate

effect on your game, so this is what we'll do in Step 3.



3. Find the `Geometry` property in the list (use the filter at the top of the list to search quickly). Change its `Scale` setting from `1` to `3`.



> **Tip**: Notice how the changed setting is now highlighted in bold? This indicates that its value differs from the default (or rather, parent's value — but this is the term from the advanced tutorial). To revert to the default value, click the arrow button on the right.
>
> **Scale** 3

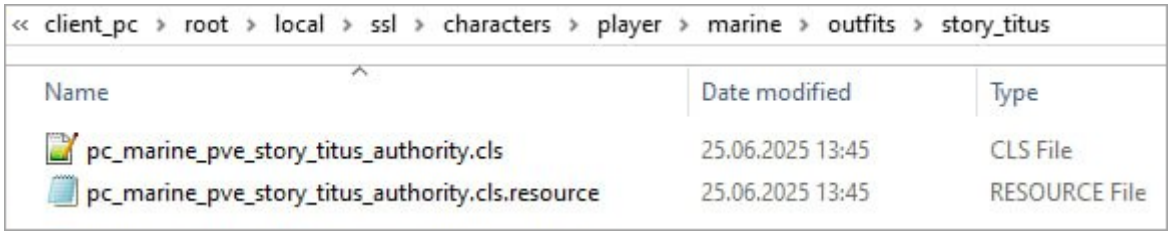
4. Press `Ctrl+S` to save the changes.

5. Right-click on the CLS file tab and select **Show in Explorer**. This will open the folder where the current file is located.
   If this doesn't work, you can alternatively select **Copy Info > Copy Full Path** and then paste the path

into Explorer manually. Be sure to delete the actual filename from the copied path – we want to open the file's parent folder, not the file itself.

6. Copy the `pc_marine_pve_story_titus_authority.cls` and `pc_marine_pve_story_titus_authority.cls.resource` files to the clipboard.

7. Paste the copied files into the `root\local` folder, retaining the relative folder hierarchy — everything that goes after `root\mods_source\default_other\`. In our example, place them in `<path_to_game>\client_pc\root\local\ssl\characters\player\marine\outfits\story_titus`. You'll need to manually recreate all required folders within `local`.



8. That's it! Now you can launch the game and enjoy the new look and feel of Titus. And we're not spoiling this experience with a screenshot.
   Don't worry when you see a warning about corrupted files and the label `MODS DETECTED`. This just means the magic is working.

If you don't see the expected result in-game, try the following:

- Check if the game shows the `MODS DETECTED` label. If not, it likely means the files weren't moved to the `local` folder correctly — please revisit Step 7 above.
- If the game crashes, verify that you unpacked the game files correctly into the `mods_source` folder, as described in the Install Mod Editor section. If this step is done incorrectly, IntegrationStudio may generate an invalid `.resource` file. You can confirm this by deleting the `pc_marine_pve_story_titus_authority.cls.resource` file and launching the game — it should start without crashing. Be sure to redo the extraction from scratch to prevent further crashes. While a `.resource` file isn't essential for this tutorial, they will be critical for your future mods.
- If the game starts correctly and shows the `MODS DETECTED` label, but Titus is still the same size, double-check that you modified the correct CLS file. There are several files with similar names – for example, Deathwatch Titus is a different actor, so changes to that file won't affect the one we're editing here.

Next Level: Learning Modding Concepts

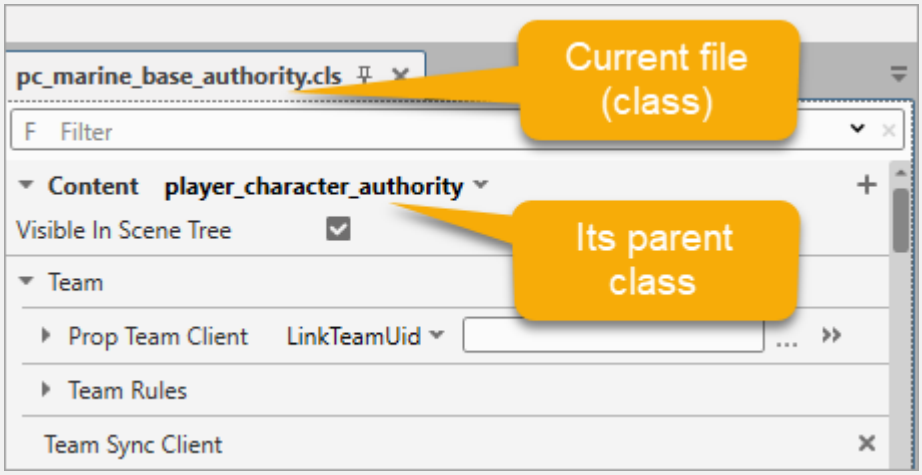
In this tutorial, you will learn the basics of systems like inheritance and blueprints, which are essential for creating advanced mods in Space Marine 2.
By the end, you'll have a mod that makes characters in the game change size every second.

To focus on new theory, we'll assume you've completed the Quick Start tutorial.

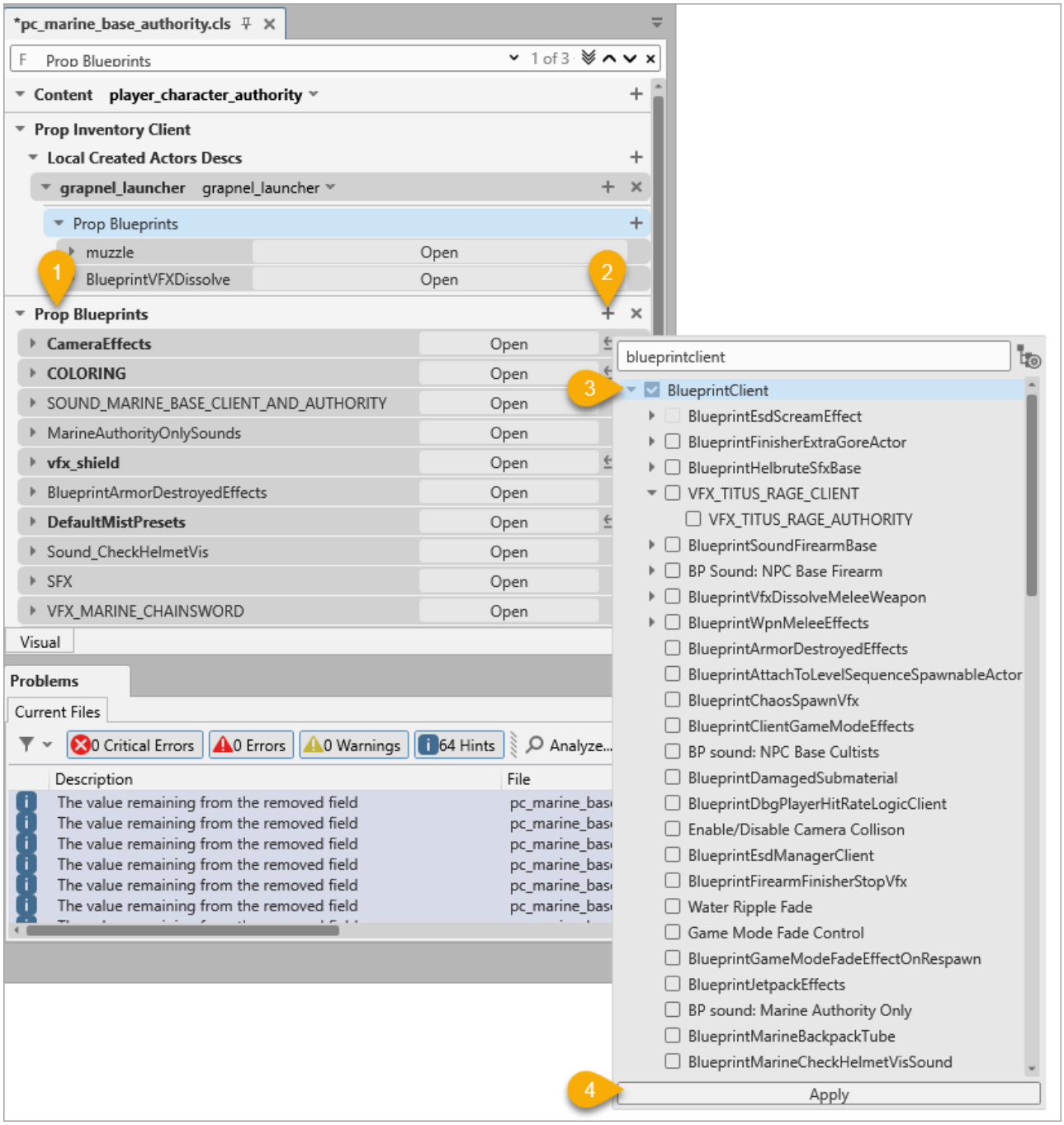
1. Open the `pc_marine_base_authority.cls` file.

> **Theory**:
> This is the main, or **parent**, class containing settings shared by all marine characters in the game.
> As you may guess, the `.cls` extension stands for "class".

According to programming terminology, individual classes of Titus, Deathwatch Titus, Chairon, Gadriel, etc. are **child** classes of `pc_marine_base_authority`.
Every property defined in a parent is inherited by its children. However, a child can override an inherited property, applying a new value only for itself. In the Quick Start, you redefined the parent's `Scale` value of `1` with the child's value of `3`. This mechanism makes it easier to distribute shared settings while allowing customization of individual actors.
The parent class of the current class is displayed at the top of the settings tab. In our example, `player_character_authority` is the parent of `pc_marine_base_authority`.



2. Find the `Prop Blueprints` property, click **+** , select `Blueprint Client`, and click **Apply**.



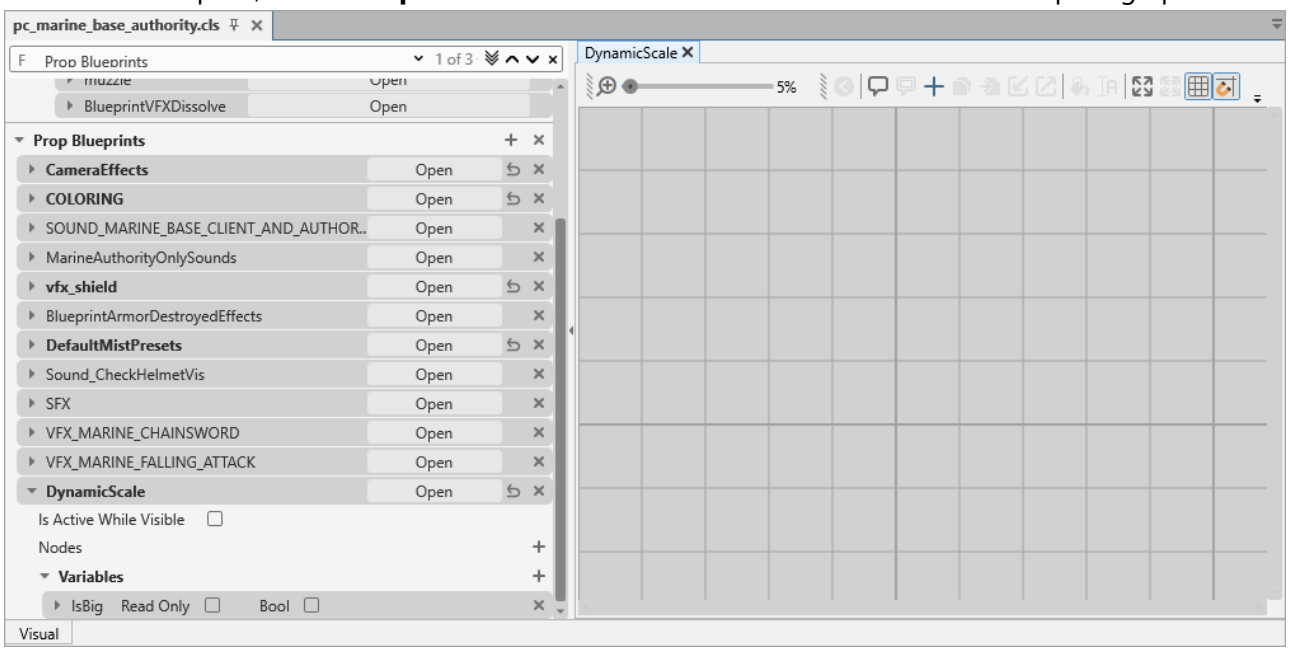
Give your blueprint a meaningful name that reflects its purpose – in this tutorial, we'll call it `DynamicScale`.

> **Theory**:
> Blueprints are logic scripts that let you define advanced behaviors without writing raw code. Think of them as visual flowcharts, or graphs: they react to events, execute logic, and update variables.

3. Expand the newly created blueprint entry to see the `Nodes` and `Variables` sections. `Nodes` is a service section you won't need, but `Variables` is where you'll add the values used by the blueprint. Just like

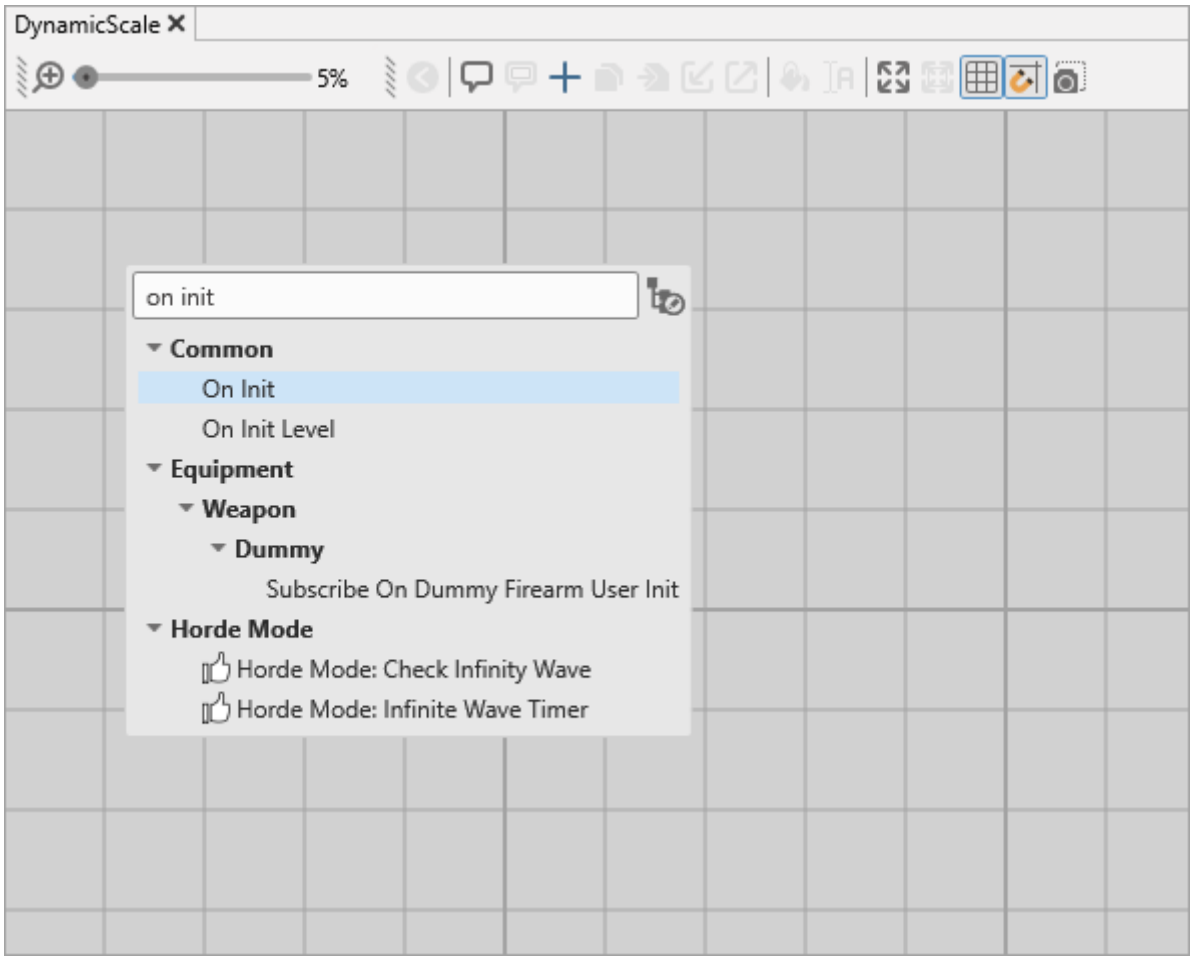
you added the blueprint, add a new `Bool` (boolean) variable called `IsBig`.

4. To edit the blueprint, click the **Open** button next to its name. This will launch the blueprint graph editor.



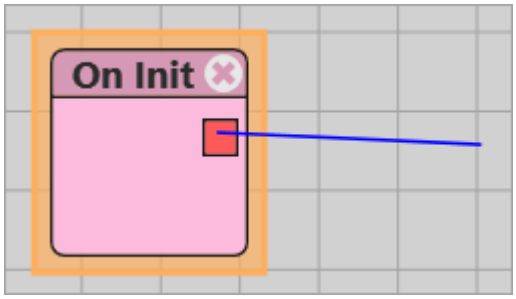
> **Tip**: You can learn more about how the game's logic works by opening existing blueprints and exploring them. For example, open the `COLORING` blueprint and navigate the graph with your mouse. Use your mouse wheel to zoom in/out. Drag nodes around for better visibility.

5. Add the first node of our `DynamicScale` blueprint: right-click anywhere on the grid in the blueprint editor and select the `On Init` node from the menu.



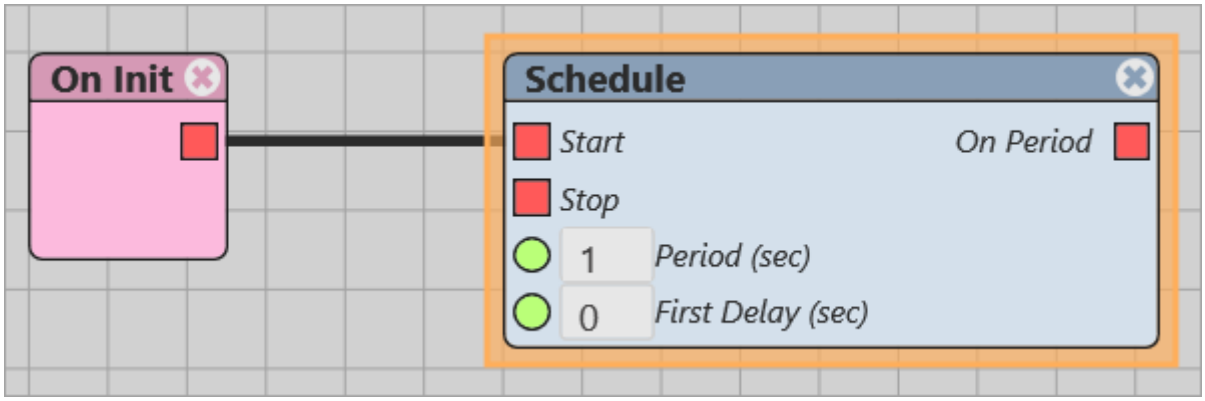
> **Theory**:
> The `On Init` node starts the blueprint logic when your actor (character) is created in the game. Because we want every actor to begin this behavior as soon as it spawns, we start our graph with this node.

6. Add the second node as a continuation of the first: click and drag from the red square on the right side of the `On Init` node.



When you release the mouse button, the node menu will appear. This time, select the `Schedule` node,

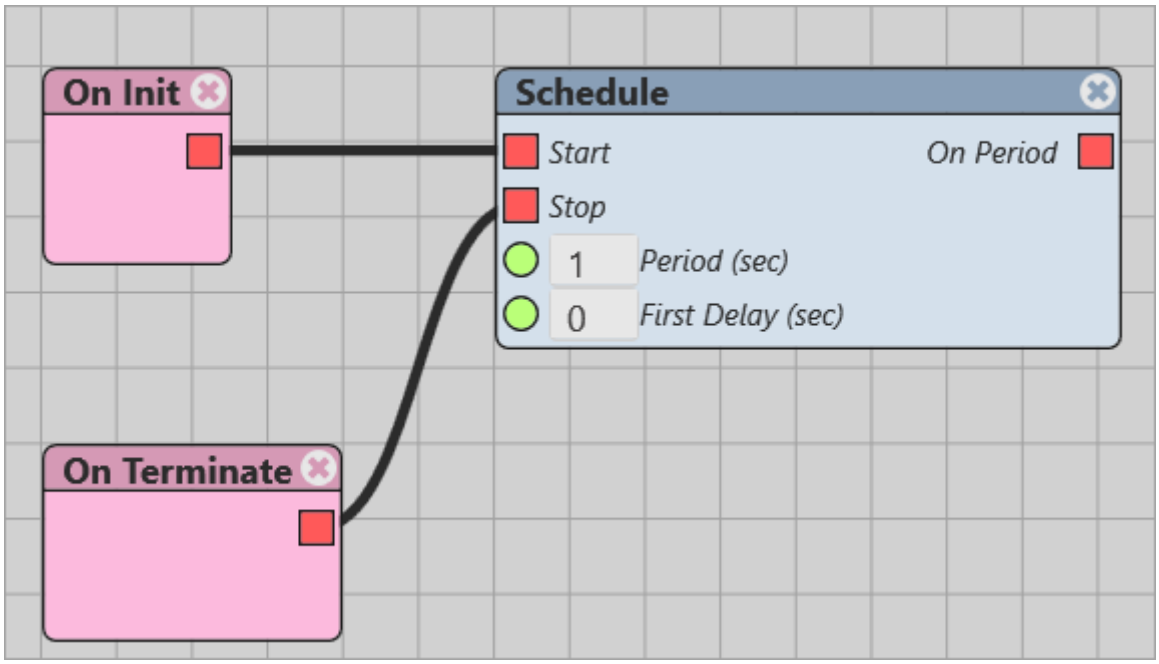
then select `Start`.



> **Theory**:
> Let's break it down. Pink nodes represent **events**, like `On Init`, which triggers when the actor is initialized. Each graph can only have one node for a specific event.
> Red squares in the nodes are **execution pins** that define the order in which nodes run. When `On Init` triggers, it starts the blueprint's scenario, which — through the connection you made — immediately starts the `Schedule` node's execution.
> The `Schedule` node sets up a repeating sequence: once started, it runs the node connected to its `On Period` pin every specified number of seconds (by default, every 1 second).

7. Next, define when the schedule execution should stop. Drag a connection from the `Stop` pin of the `Schedule` node and select the `On Terminate` event node. This ensures the schedule stops when the actor is removed from the game.

> **Tip**: When you draw a connection from a node, the editor will suggest only suitable nodes that can possibly follow this node.



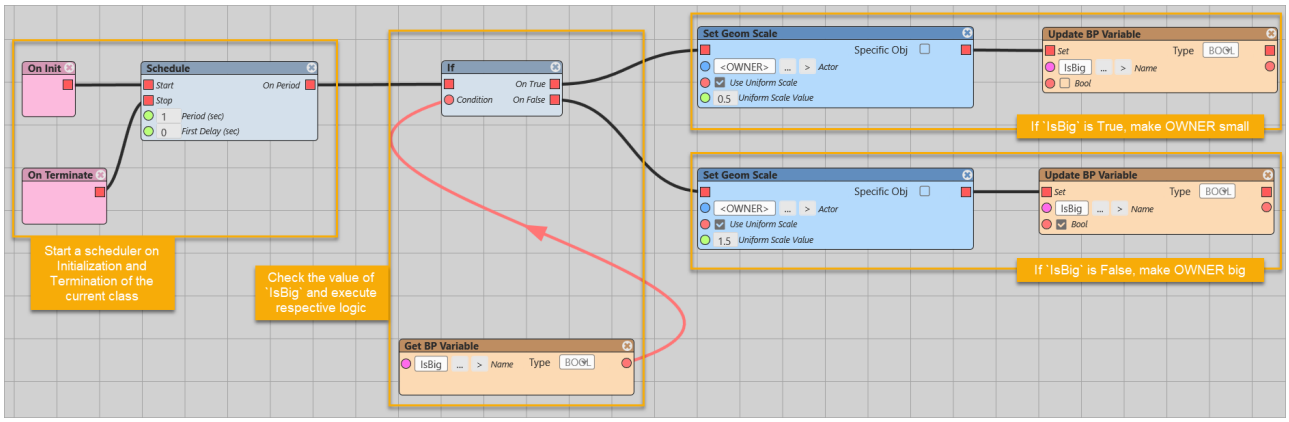
8. Now recreate the rest of the blueprint graph as shown below to complete the behavior:

   - **If**: checks a `Condition` and triggers one of two nodes depending on whether it's `True` or `False`.
   - **Get BP Variable**: retrieves a blueprint variable's value.
     This is a **getter** node: it has no execution pin and simply returns a value requested by another

node. In our example, it gets the value of `IsBig`, the boolean variable you created in Step 3. A new `Bool` variable is `False` by default (unchecked box).

> **Tip**: To quickly access a list of all values available in a field, click **...** next to it.

- **Set Geom Scale**: sets a custom value for the `Scale` property of a given actor.
  Leave the default value `OWNER` (which refers to the current class). Enter a smaller value in the node that runs when `IsBig = True` and a larger value in the node for `IsBig = False`.
- **Update BP Variable**: updates a blueprint variable's value. In our case, `InBig`.
  After increasing the actor's scale, mark the `Bool` checkbox so `IsBig = True`; after decreasing the scale, leave the checkbox unchecked. This way, the `IsBig` value toggles with each iteration of the schedule, switching the actor between big and small sizes.



9. Save the changes in `pc_marine_base_authority.cls`.

10. Copy the modified `pc_marine_base_authority.cls` and its corresponding `.resource` file as you did in the Quick Start. Note that in this tutorial, the `.resource` file is essential – the mod won't work without it.

11. Launch the game and watch every character cycle through growing and shrinking every second!

Sharing Mods

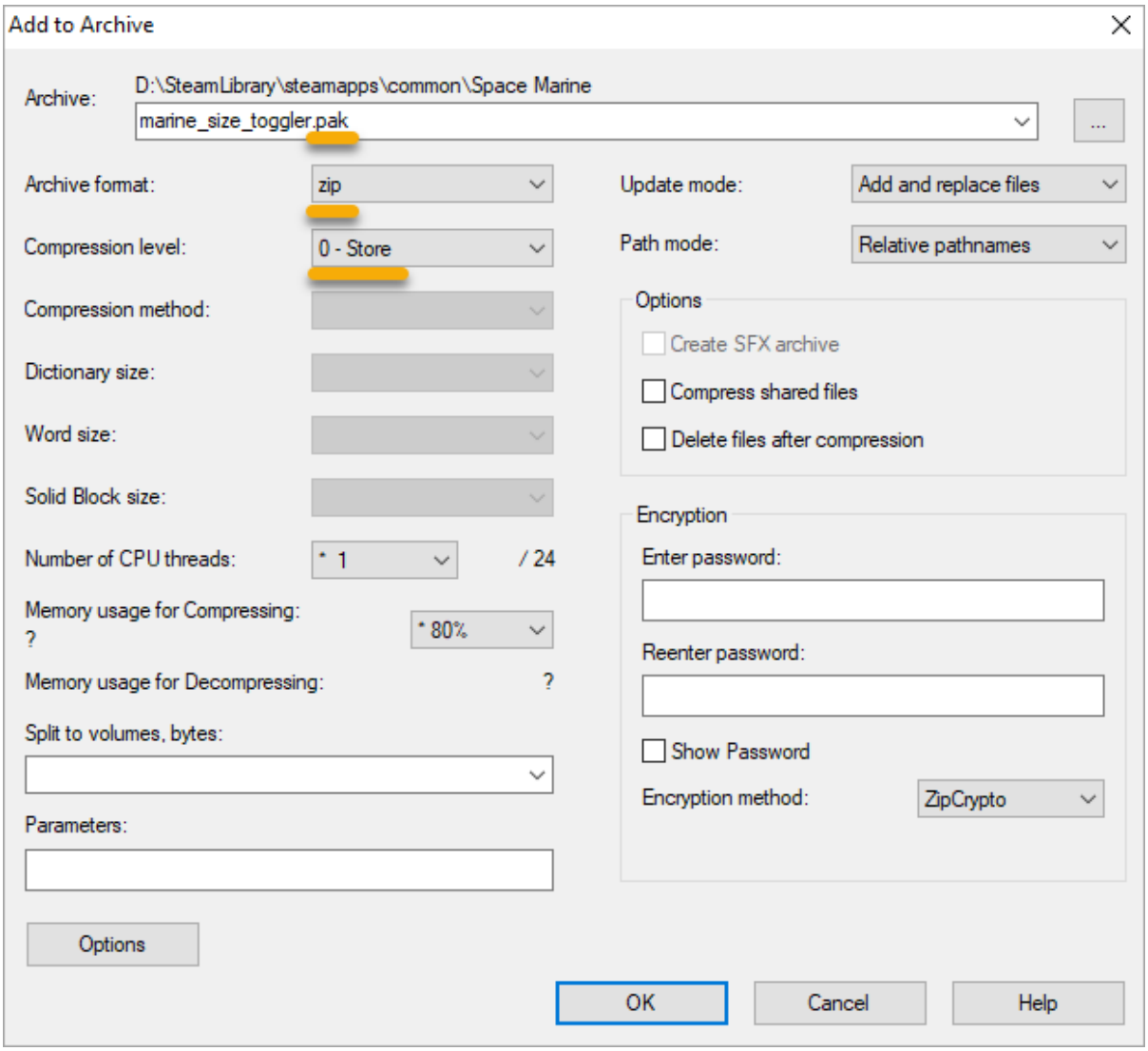
If you want to share your mod with another players, pack it into a `.pak` file. Here's how:

1. Open the `<path_to_game>\client_pc\root\local` folder.

2. Select all the files that belong to your mod. Your goal is to pack all of them into a single archive while retaining the original folder hierarchy. For example, in the Quick Start, your archive should include the following nested folder structure: `ssl\characters\player\marine\outfits\story_titus`, containing two files `pc_marine_pve_story_titus_authority.cls` and `pc_marine_pve_story_titus_authority.cls.resource`.
   Pack the mod files into an archive with the following settings:

   - **Archive format**: zip.
   - **Extension**: `.pak`.
   - **Name**: any descriptive name reflecting your mod's functionality, e.g., `marine_size_toggler`.

- **Compression mode**: 0-Store (no compression).
  You can use any suitable archiving tool, such as 7-Zip.



> **Tip**: To make packing mods easier, we recommend keeping your `local` folder organized so it contains only the files related to the mod you're currently working on. Otherwise, it can be difficult to keep track of which files belong to which mod.

3. Move the `.pak` archive to the `<path_to_game>\client_pc\root\mods\` folder. Delete the mod files from the `local` folder – since you now have a proper mod pack, there's no need to keep duplicate files locally.

> **Note**: Mod paks don't require creating `.cache` files. If you notice those inside the `paks` folder, just ignore them.

4. Launch the game to verify that your mod works as expected.

5. Share your `.pak` archive with the Space Marine 2 community.

You can install an unlimited number of mod packs in your game. However, note that if multiple packs contain the same files, the game will use the file from the archive that comes first alphabetically. For example, if the mod packs `a1.pak` and `a2.pak` both modify the same game file, the game will use the version from `a1.pak`.

Hotkeys

Common editor hotkeys:

- `Ctrl+O` – open a file via Explorer.
- `Alt+Shift+O` – browse all files available for editing. Offers a handy filter by filename or extension.
- `Ctrl+N` – create a new file in the current folder open in the Content Browser.
- `F2` – rename a file or folder selected in the Content Browser.

To assign custom hotkeys, go to **Edit > Hot Keys**.

Additional sources

- Unofficial Space Marine 2 Modding Hub on Discord
- Space Marine 2 on NexusMods