Mission Building Quick-Start

Introduction

The first thing to understand is that the game is designed with the procedural missions system in mind. This means that if you construct your level appropriately you will have a great deal of flexibility available to you in terms of adjusting things like the biome, weather, foliage, landforms, and so on. Unfortunately it also means that the process of putting together a custom level and laying out the gameplay manually is significantly more complicated and time consuming than it would be in a typical game.

The purpose of this guide is to go step-by-step through putting together a simple mission taking place on a custom made single-tile level (as opposed to the proc missions in the game which take place on a level comprised of a matrix of terrain tiles).

This guide also assumes you have a decent level of familiarity with level building and basic scripting in the Unreal Engine, so I won't be going into any detail about how to do things other than those which are peculiar to MW5.

Step 1: Plan Events/Objectives

Before starting, you should have an idea in mind of what sort of gameplay you want to have in the mission. There are a number of mission components from which you can assemble a wide variety of gameplay events, and the satisfaction of any of these mission components can be marked as an objective for the player to complete in the mission.

Some examples of events and objectives you can do with mission components are:

- Set a timer
- Spawn AI units to attack a base
- Spawn AI units to attack the player
- Spawn AI units to travel to a location
- End the mission
- Fire artillery
- Player must capture a base
- Player must destroy certain buildings
- Player must defend certain buildings
- Player must scan an item
- Player must target a certain building or unit
- Player must destroy certain units
- Player must go to a certain location
- Player must go to an evac point

Using combinations of these and simple flow logic, you should be able to construct a wide variety of missions. Available, in addition to this document, will be the complete *Mission Flow Node Scripting Reference* document which was written throughout the course of the project by the designers who created the mission components.

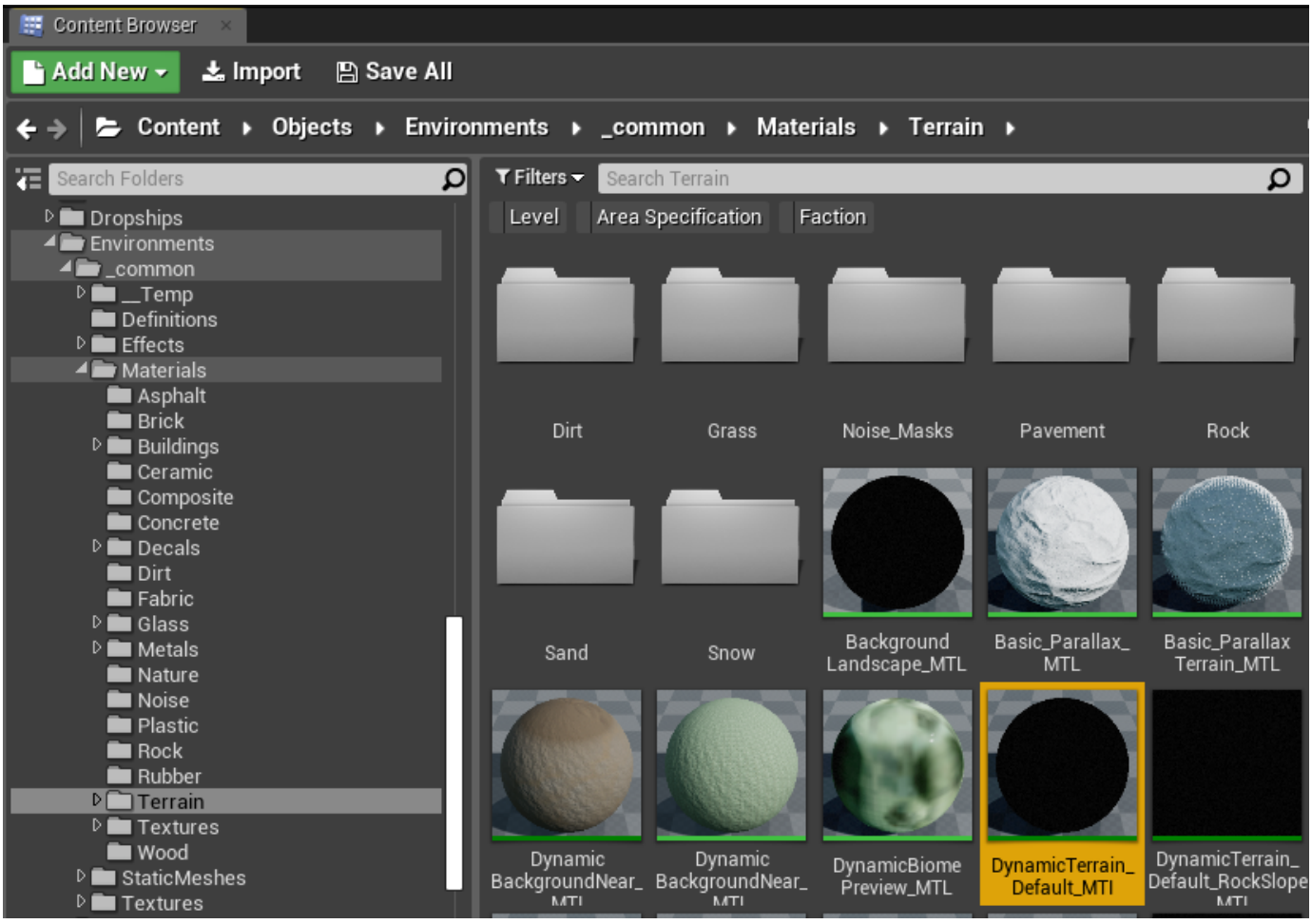
Step 2: Make a Terrain Level

Landscape Object

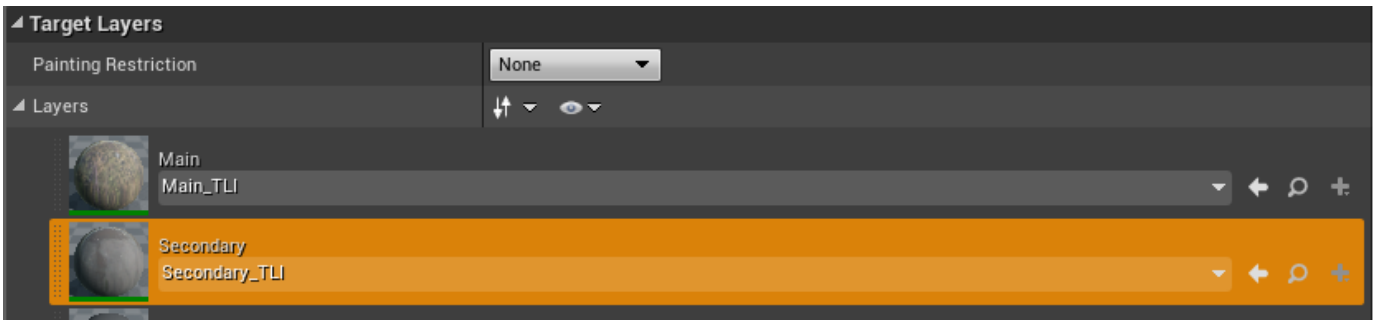
Because we're not using a matrix of connected terrain tiles for this mission, you can make your landscape object any size or resolution you want. For reference, the standard tile size for proc missions is a square 75600 units in diameter and 505x505 resolution. The smallest proc mission level ever would be 3 x 3 tiles (226800 units or 2268m across), and the largest would be 8 x 8 (604800 units or 6048m across). These are not limits, you are free to make your level whatever size you want, performance should be your only limitation in that respect.

For the terrain material, I recommend using one of the **`DynamicTerrain`** material instances found in **`Content/Objects/Environments/_common/Materials/`**

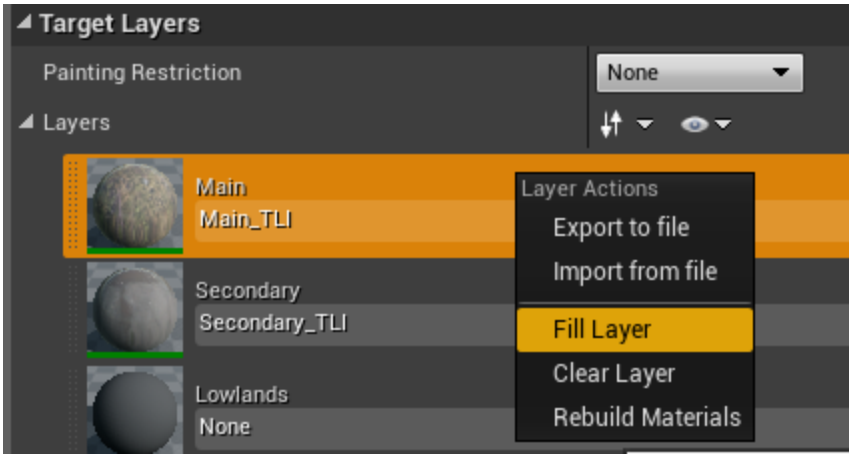
There will be a selection of them, for now I'll suggest **`DynamicTerrain_Default_MTI`**

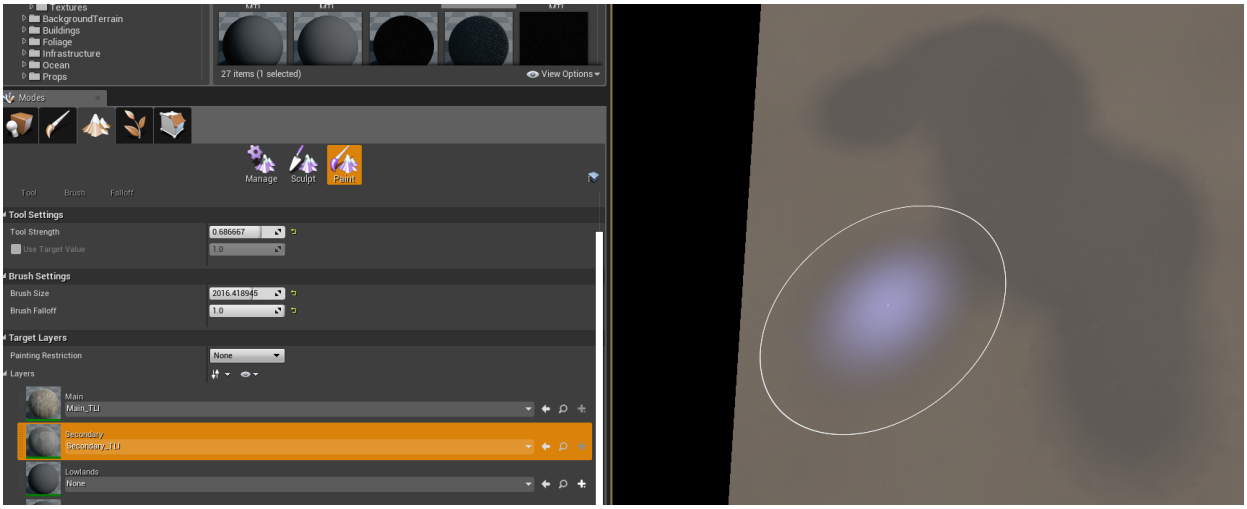


This material has two layers, **`Main`** and **`Secondary`**. You will need to set the **`Layer Info`** objects for each of these. They already exist, just select them.



I recommend filling the landscape with the main layer and painting the secondary layer as you see fit.

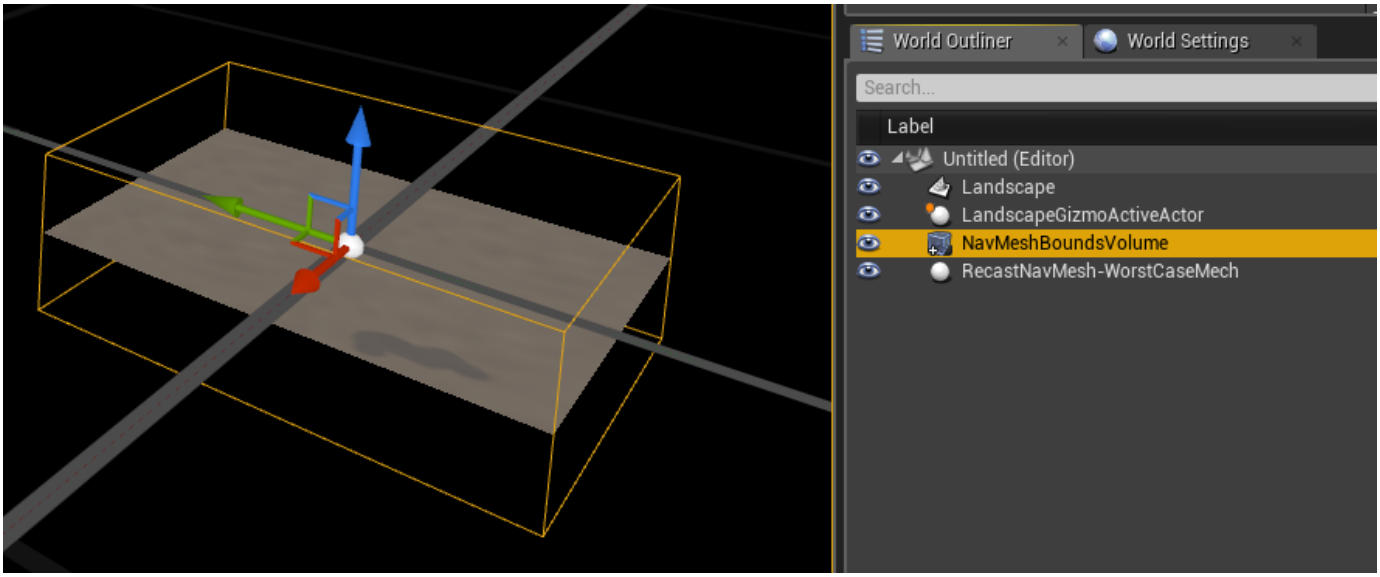




This will work nicely with the biome system. In the Forest biome, for example, the main layer will be green grass and the secondary layer will be brown dirt. Some of the other dynamic landscape materials will have additional layers, but don't use those materials unless you need the additional layers because it will add to the performance cost of the landscape material.

NavMeshBoundsVolume

The main level must contain a **`NavMeshBoundsVolume`** that encompasses any area where you wish for the AI to navigate.



Foliage Spawners

For things like trees, rocks and bushes, we do not place the specific assets directly into the levels. Rather we place foliage spawners. This will allow us to change both the varieties of asset and their respective densities based on which Biome we are specifying for the mission. For example, a tile with 1000 tree spawners will only spawn 10 trees if the tree density is set to 0.01 in the biome.

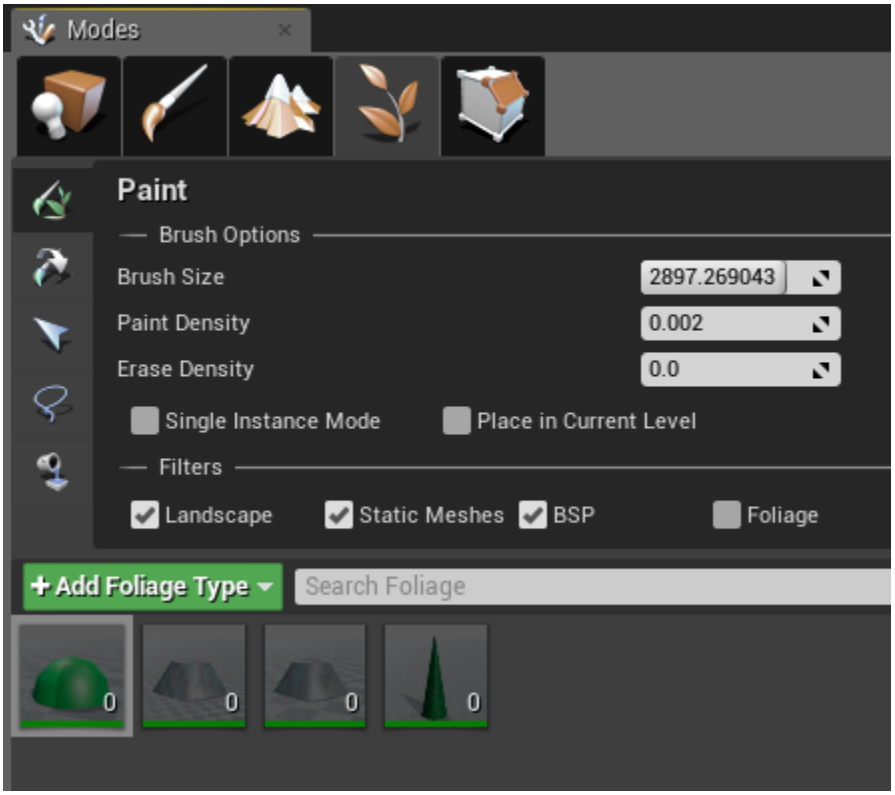
The foliage spawners are found in:

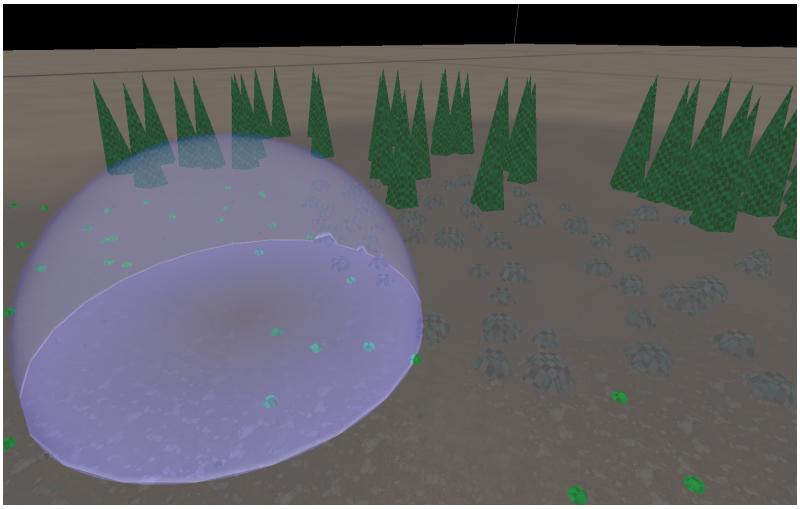
`Content/Objects/Environments/Foliage/_common/FoliageSpawnerTypes`

They need only to be set up as the foliage for that landscape and then can be painted/placed using the foliage editor dialogue. When the mission runs it will select the appropriate foliage items based on the specified Biome and DecorationCollection objects.

The standard ones to use (the ones which will be most likely to have assets specified for them in the existing biomes) are:

- **`FoliageSpawnerType_Bush_FLT`**
- **`FoliageSpawnerType_Rock_Large_FLT`**
- **`FoliageSpawnerType_Rock_Normal_FLT`**
- **`FoliageSpawnerType_Tree_Normal_FLT`**





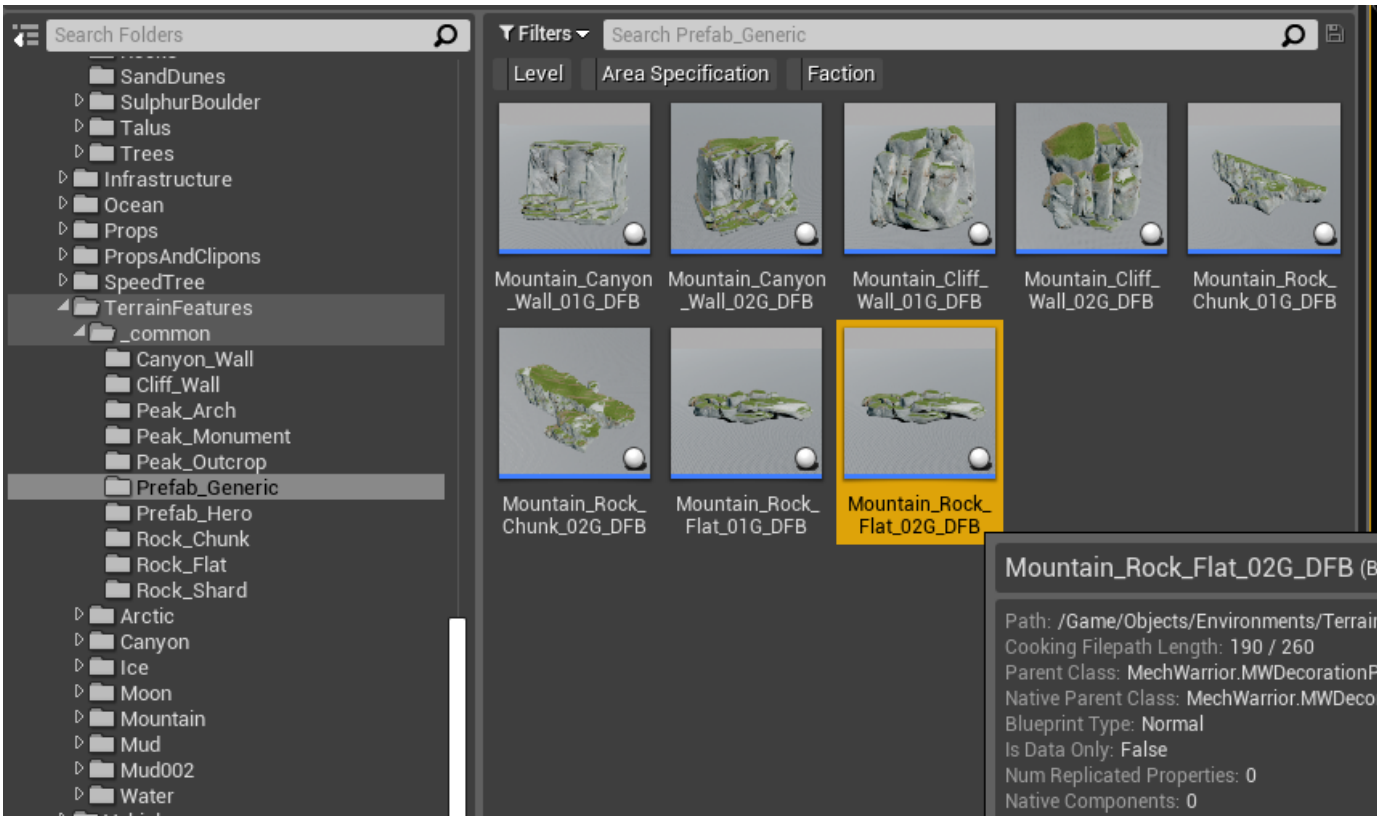
If you populate your landscape with these types of foliage spawners, they will spawn appropriate assets for whatever biome is selected.

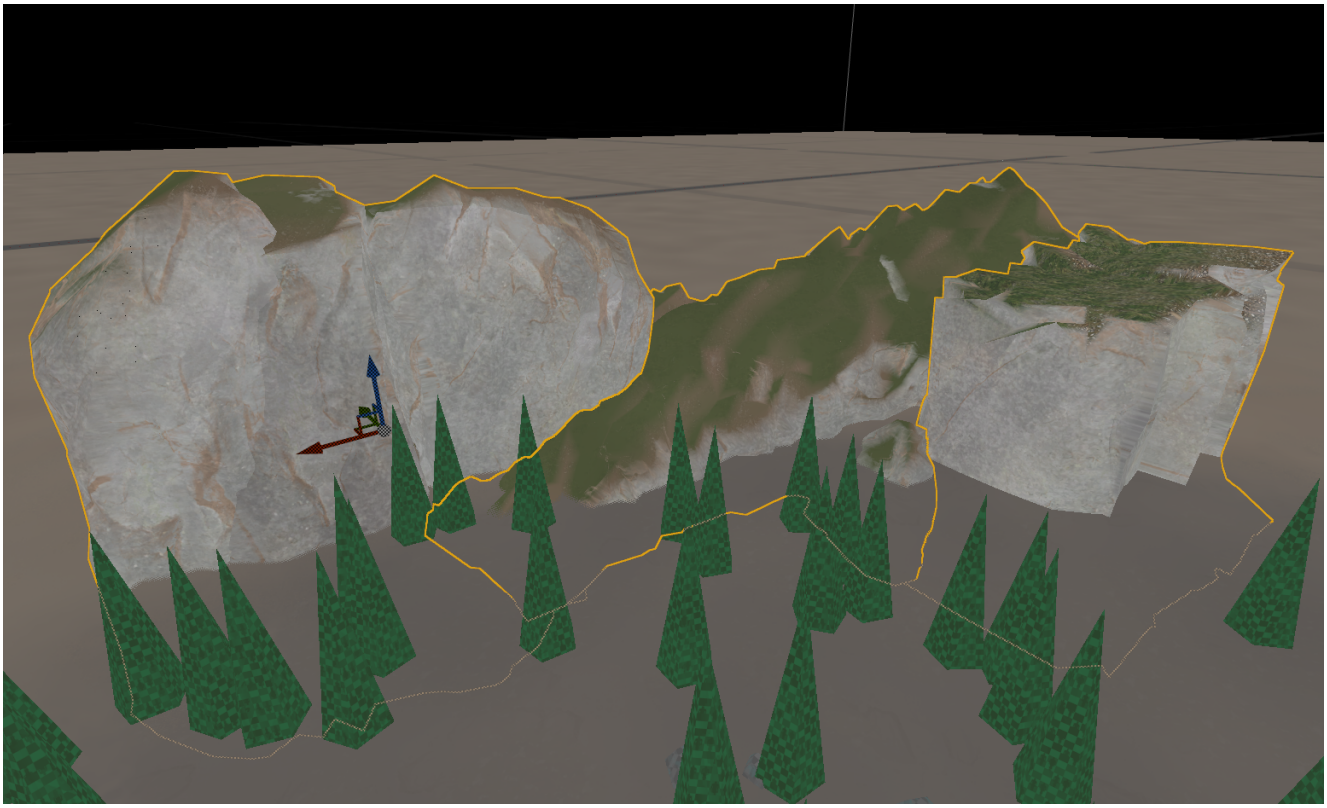
Landforms

Landforms are giant terrain features, in most cases larger than a ‘Mech. They are also selected and spawned via spawners, and placed in the foliage system. The most effective method for placing them is to find the landform blueprint you want in one of the folders located here:

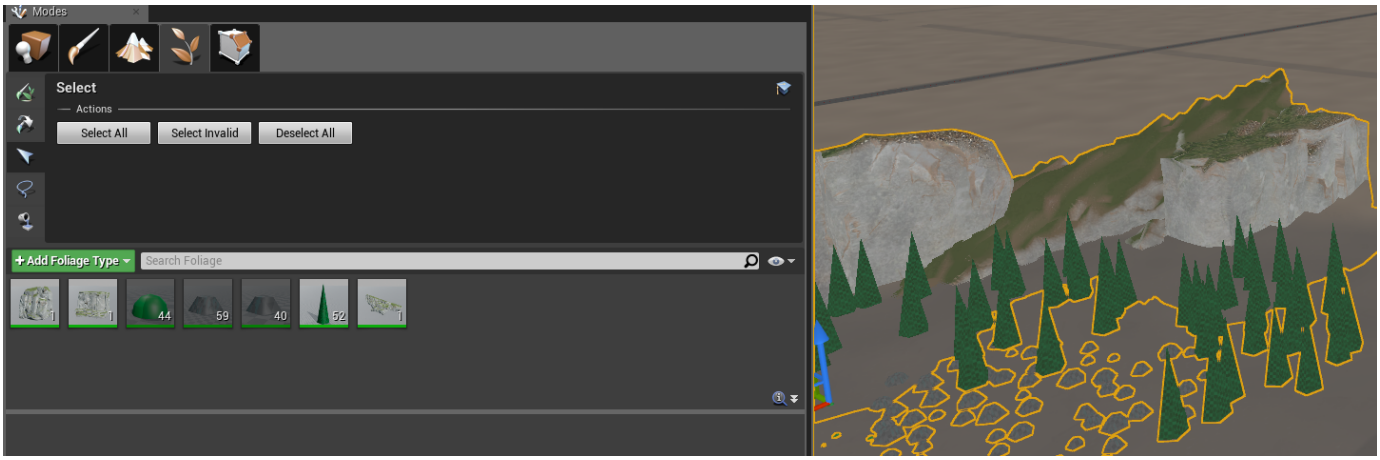
- **`Content/Objects/Environments/TerrainFeatures`**…

Then you can drag and drop the blueprints into the level, and scale and orient them appropriately. There will be a lot of files in these folders, you want to use the ones that are blueprints and in the folders named "Prefab" etc.



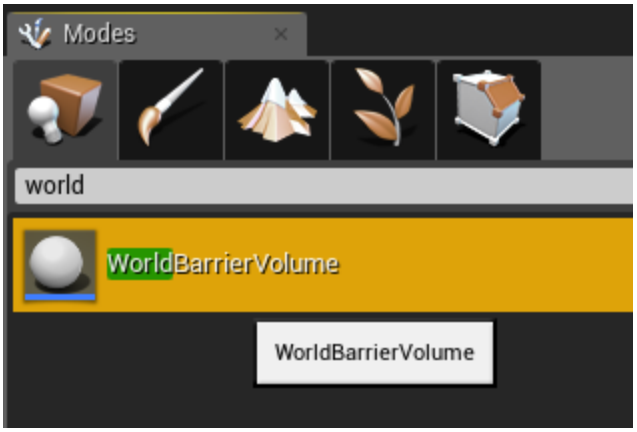


Once you are happy with the location and orientation of a landform you can check "**`Bake Into Instanced Foliage Actor`**" in the Details panel. This will replace the blueprint instance with a foliage spawner similar to the type described above. You can still select and manipulate the spawner, but now you have to do it via the foliage dialogue which is a little less convenient. This is why it is advisable to do this after you have placed all your landforms. The editor may freeze up for a while after you click the checkbox. Don't panic. Once those are baked, they are now part of the foliage system and they will appear in the foliage dialogue.

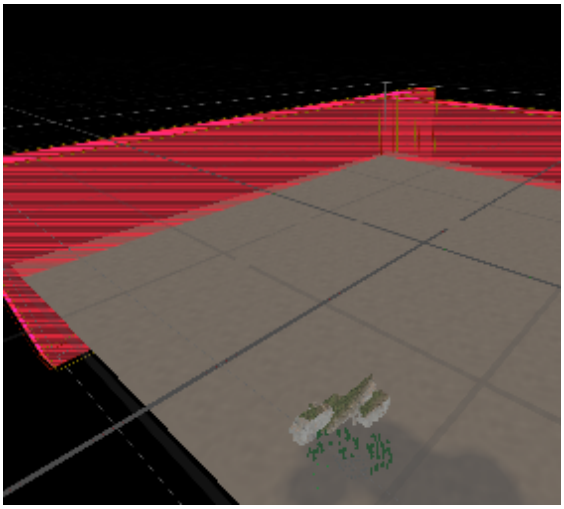


Barrier Walls

If you wish to have collision walls around the level perimeter, preventing the player from exiting the level, add **`WorldBarrierVolume`** objects and scale them to the desired size.



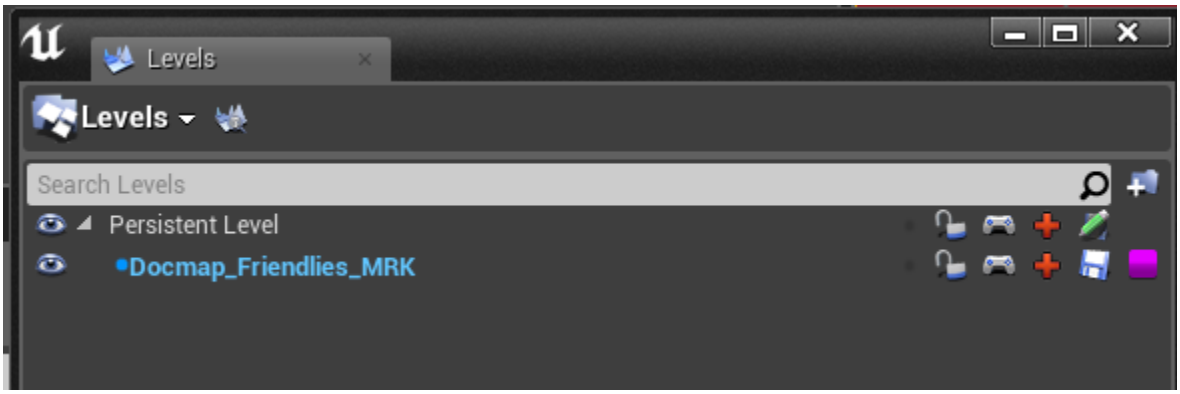
These will also show the red glowing indicator when the player moves within a short distance of them.



Make sure your level is saved and we're now going to create markup levels.

Step 3: Set Up Markup Levels

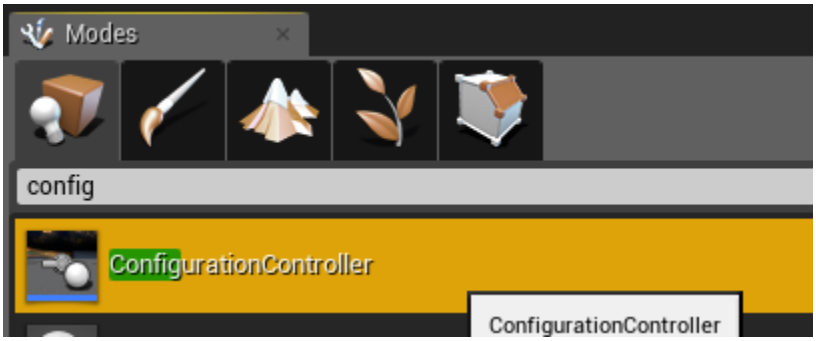
First make a new level and include it as a sublevel in your terrain level from before. This is going to be your markup level. In the game these are typically given the same name as the terrain level with "**_MRK**" tacked onto the end.



For the proc mission tiles, normally some tiles are flagged as enemy tiles and some as friendly tiles. Since we're only making one big tile, we'll have to make two separate markup levels. I'll describe the process just once, but ultimately you'll be doing this twice. Once for friendly stuff, once for enemy stuff.

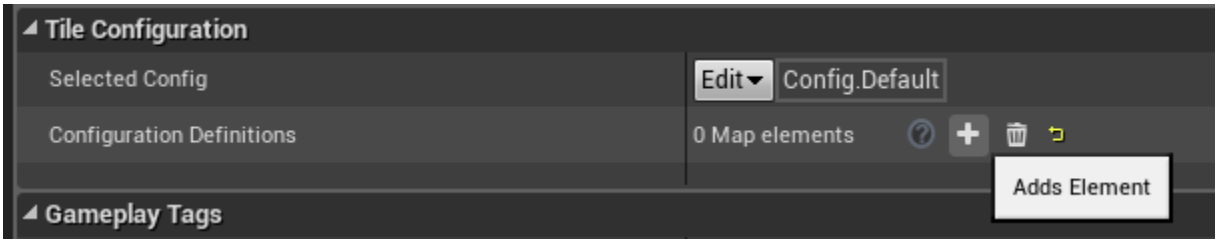
This level is going to be where you put all the “locator” objects that will indicate where things are going to spawn in your mission. (Bases, AI units, triggers, etc.)

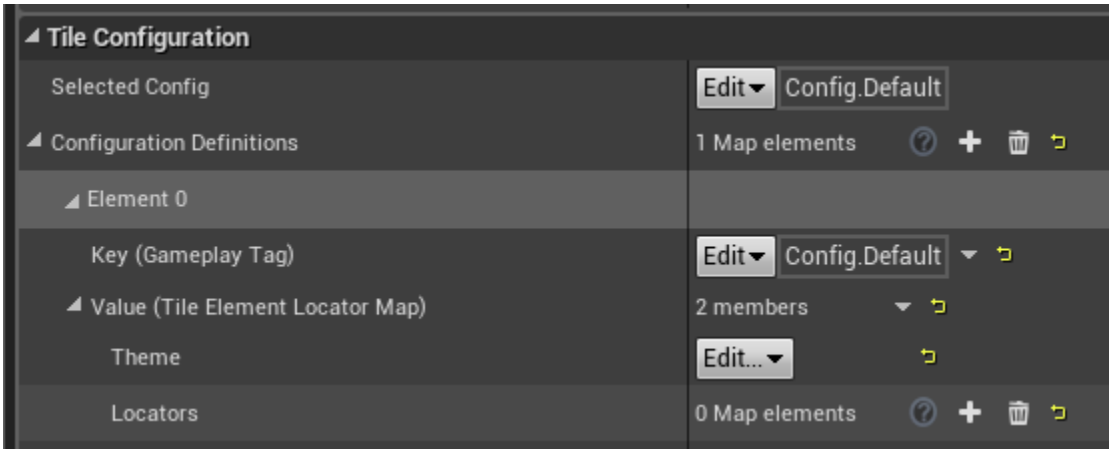
The first thing to place in this level is a **`ConfigurationController`** object. You can find this in the “Place” dialogue in the “Modes” window by typing “Config” into the filter bar.



Since this level is going to be unique to your mission, you’ll only need to set up one configuration for your tile, but we still need the **`ConfigurationController`**.

With your **`ConfigurationController`** selected, go into the Details panel in the **`Tile Configuration`** section and click the + sign to add a new **`Configuration Definition`**. For its Key Gameplay Tag, select **`Config.Default.`**





Do all of the above twice. You can name the levels whatever you want, but I recommend naming them something like this:

`MyLevel_Friendlies_MRK`

`MyLevel_Enemies_MRK`

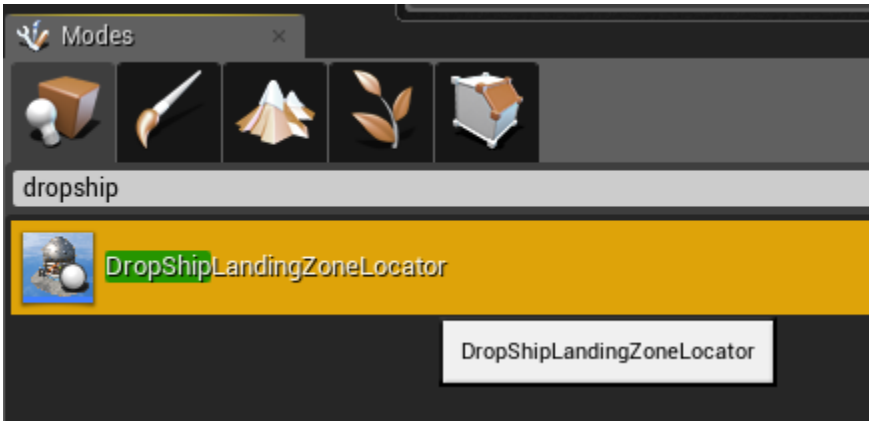
These should both be added as sublevels to your main level, with their offset at 0,0,0 (that’ll be the default unless you were to manually change their offset for some reason).

From now on you'll be opening up your main level file to add anything, but you'll want to be cognizant of which level you are adding things into and make sure to select the appropriate one from the levels panel. To select the level you want to work in, double click it in the Level window. Don't forget the active one will be highlighted in blue text.

Step 4: Friendlies Markup Level

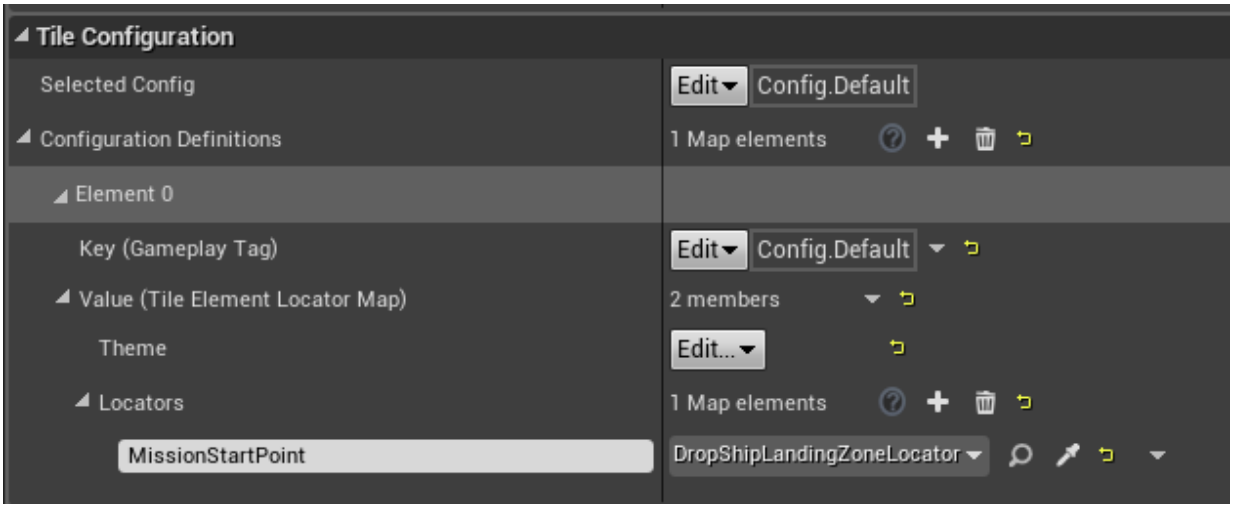
Select the Friendlies markup level you just made in the levels panel so that anything you place in this step goes into the friendlies markup level.

First place a **`DropshipLandingZoneLocator`** where you want your mission to start. This can be used for either disembarking the player from a dropship or just fading into the mission. For this and all the locators you are about to place, I recommend naming the objects in a tidy and readable way – this will help you in the future for finding them in the world outliner panel and adjusting them.



In order to use this locator in the mission, you will need to add this locator to the tile configuration that you created in the **`ConfigurationController`**. To do this, select the **`ConfigurationController`**, go down to the **`Config.Default`** configuration you created and under Locators click the + sign. This will create an entry. In the text box that says "0" by default, enter the name by which you want this locator to be referenced in the future – something like "MissionStartPoint", for example. This is how you'll find this locator inside the **`AreaSpec`**, which will be important later.

Then in the dropdown box next to that, click the arrow and select the **`DropshipLandingZoneLocator`** you just placed. Now you've got a locator set up in a way that you can reference later. This is essentially how you'll be adding all the locators to your mission.



This is all you really NEED to do in the friendlies markup level for now. If you wanted a friendly base, or friendly AI units in your mission, you'd add all the locators for that in this level. But all of that setup is basically the same as setting up enemy ones, so I'll just move on and describe those all in the next section.

Step 5: Enemies Markup Level

Locators for Spawning AI Units

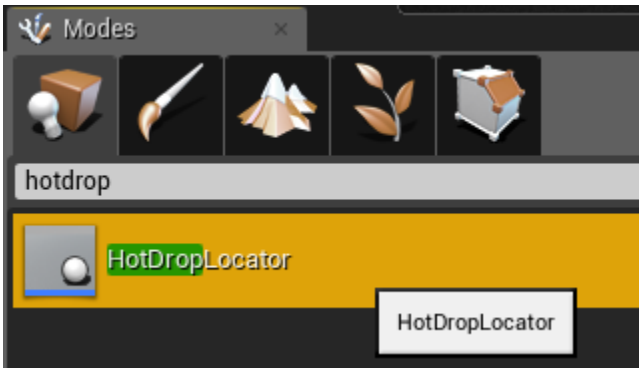
There are three ways to spawn AI units.

a.) Have them drop from a dropship (four maximum, mechs only).

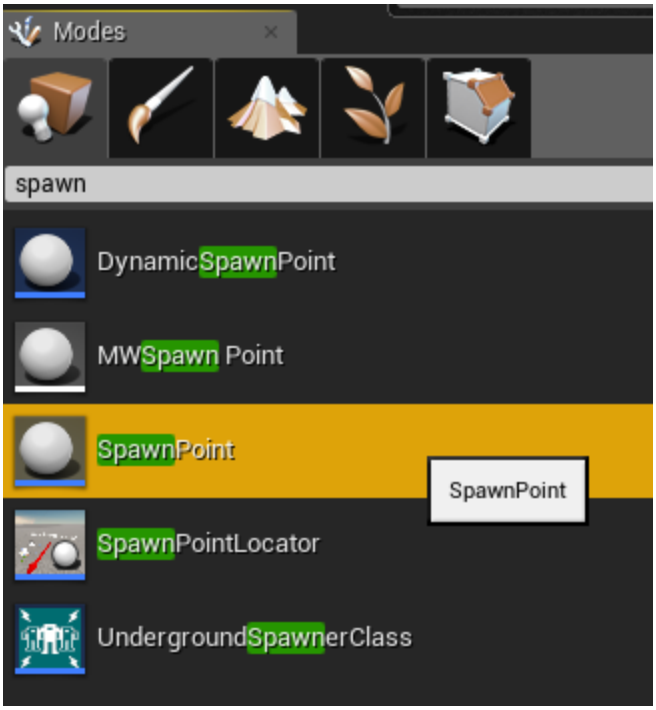
b.) Spawn them from a **`WaveLocator`** (as many as you want, any type, they will pop into existence, you can set them up into waves and add some basic logic for spawning the successive waves – timer, x% killed, etc.)

c.) Spawn them in a base (as long as it has appropriate spawn points, the units will exist from the moment the mission loads)

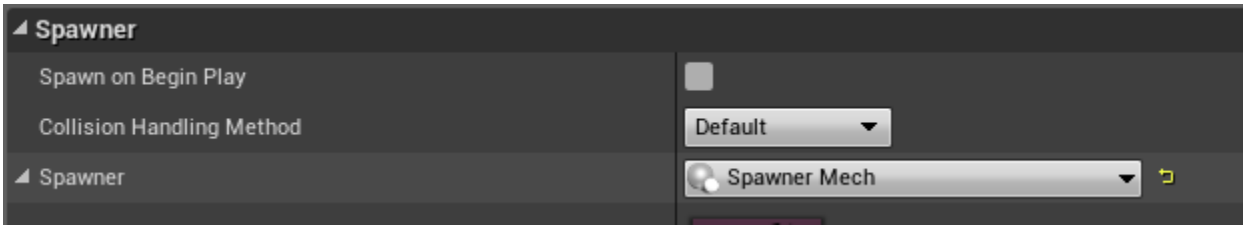
For dropping mechs from a dropship, you just need to place a **`HotDropLocator`**.



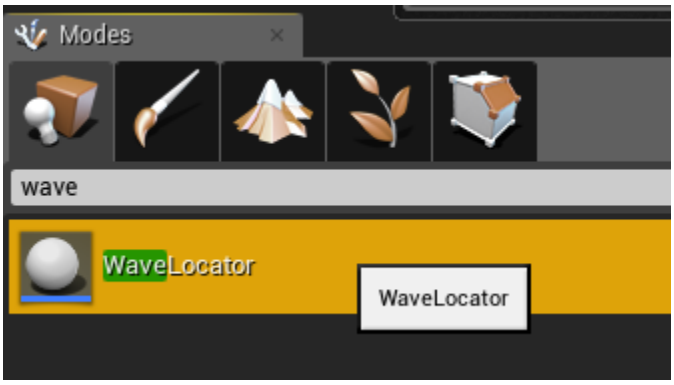
For spawning units from a **`WaveLocator`**, first place **`SpawnPoint`** objects where you want the individual units to spawn. Note: do not use **`MWSpawnPoint`**, do not use **`SpawnPointLocator`**, use **`SpawnPoint`**.

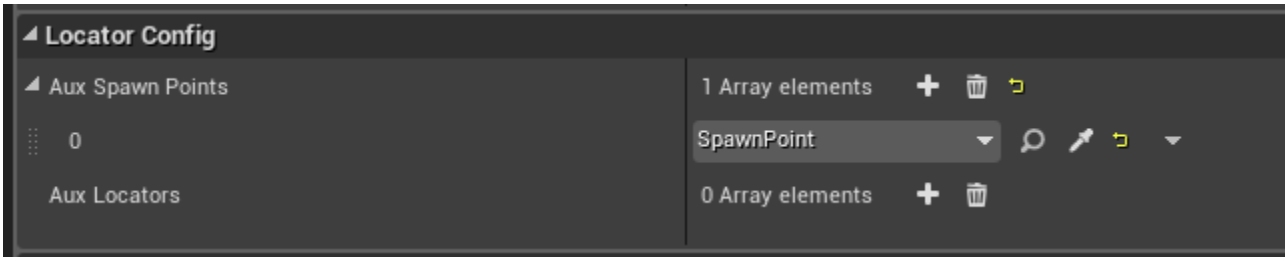


Decide what type of unit you want to spawn from each **`SpawnPoint`**, then in the details panel under **`Spawner`**, select the type of unit – **`Spawner_VTOL`**, **`Spawner_Mech`** or **`Spawner_Tank`**.

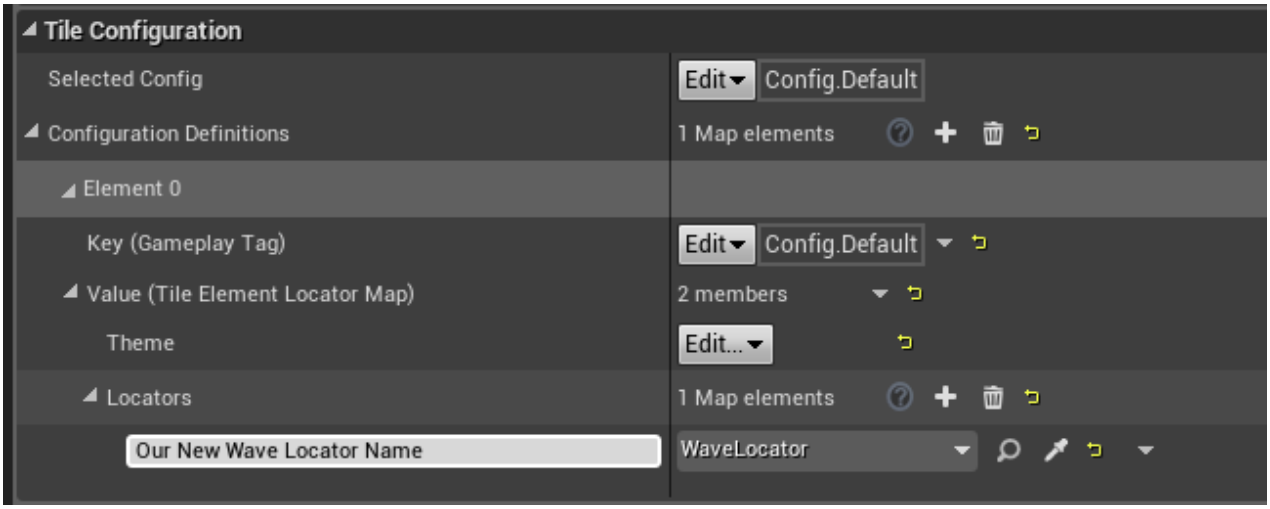


Then add a **`WaveLocator`**. Once your **`WaveLocator`** is added, add spawn points to it in the Details panel under **`Locator Config / Aux Spawn Points`**. Add all the spawn points you added earlier that you want trigger at the same time, or by the same logical structure in the mission.





Don't forget to add the **`WaveLocator`** to the Configuration in the **`ConfigurationController`** and name it appropriately. Don't forget, we're now working in the **`Enemies_MRK`** level which needs its own configuration controller, not the same one we put the **`DropshipLandingZoneLocator`** in in our **`Friendles_MRK`** level.



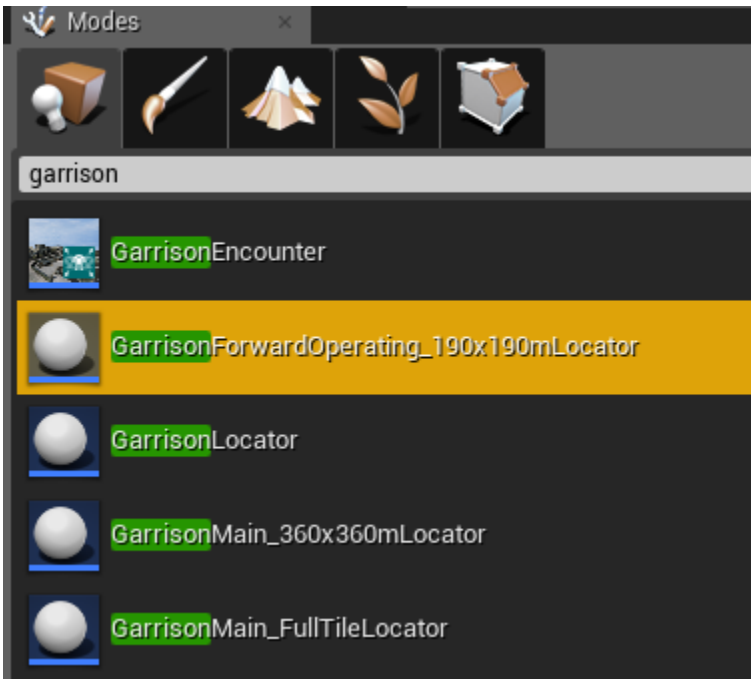
Later on in the **`AreaSpec`** section of this document I'll cover how to specify the units and how to actually spawn them.

Locators for Spawning Garrisons

The third option for spawning AI units I mentioned above is spawning them at bases. You'll notice I used the word "garrison" above. Early in development of this game, the word "garrison" was chosen as a way of referring to bases without confusing the issue of code inheritance and "base classes". The usage expanded, ultimately, to refer to essentially any sub-level that we spawn into one of our main levels. So any base, city, farm, factory, basically anything at all that gets saved out as a discrete level, and then spawned in other levels is generally going to be referred to as a garrison. From here on out I will refer to bases as "garrisons".

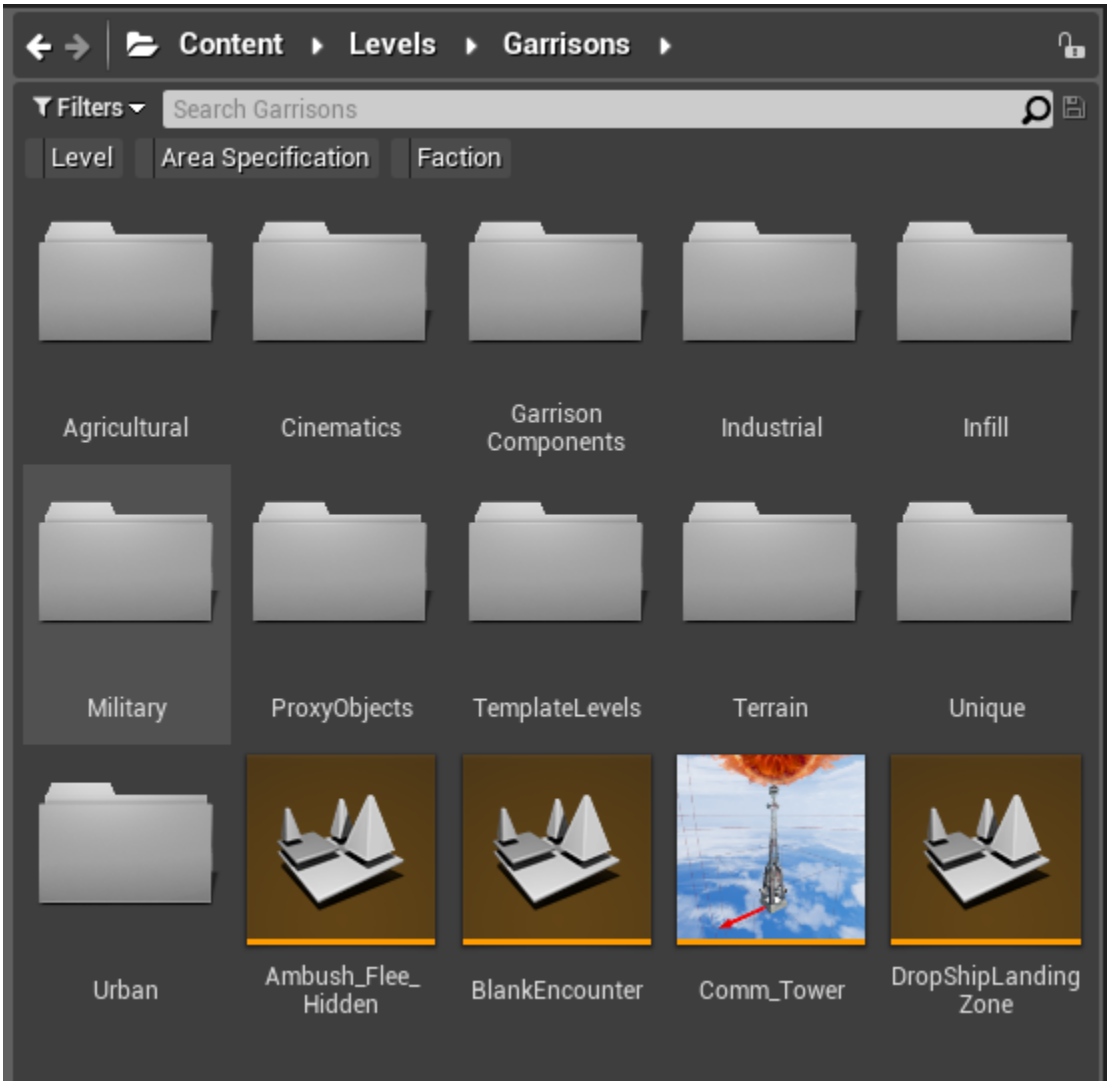
Anywhere you want to spawn a garrison, you need a locator. The three main sizes for garrisons are squares with sides of 190m, 360m or a "full tile" (which is a little smaller than a 756m standard tile). You'll need to make sure if you are spawning a garrison into your level that you have a flat, unobstructed area of the appropriate size. The locators for these three different types of garrisons are:

- **`GarrisonForwardOperating_190x190mLocator`**
- **`GarrisonMain_360x360mLocator`**
- **`GarrisonMain_FullTileLocator`**



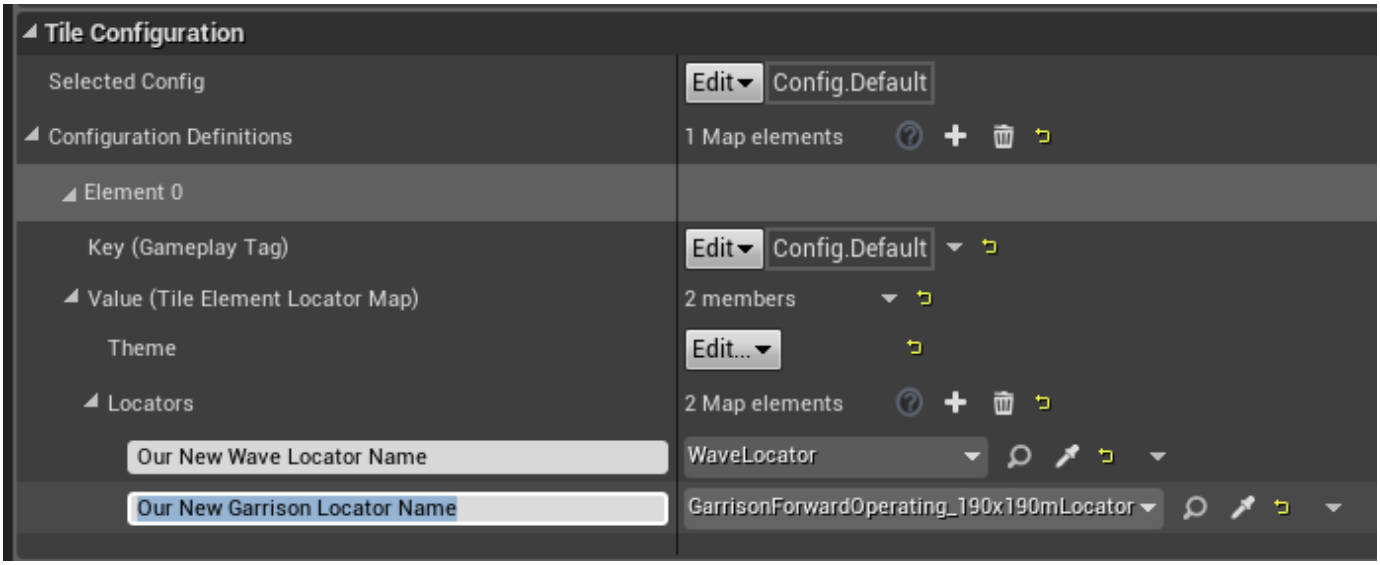
There are a number of other smaller types of garrisons with their own locators as well. You can experiment with these and see which ones will support the garrisons you are looking for.

The garrisons themselves are just levels which will be ultimately spawned into your mission with their origin point at the coordinates and orientation of the locator. They can all be found in the **`Content/Levels/Garrisons`** folder, organized in various subfolders.



As for spawning AI units in these garrisons, that'll be covered in the **`AreaSpec`** section of this document. They will not require locators of their own in the markup level, just having the garrison will cover it.

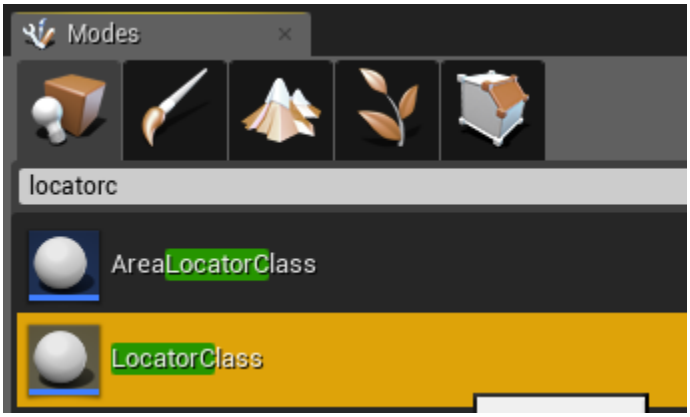
Don't forget to add your garrison locators to the Configuration in the **`ConfigurationController`** and name them appropriately.

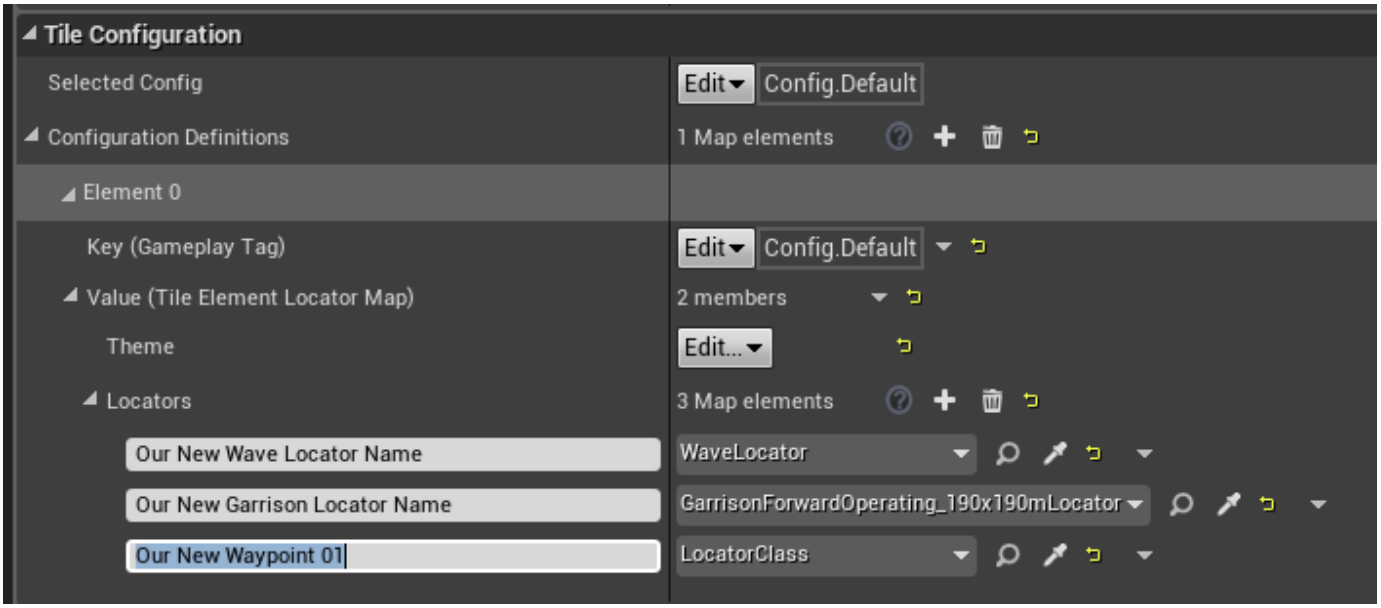


Any of these locators I've described in the enemy markup level can also be added in the friendly markup level if you want them to be friendly forces/garrisons.

Step 6: Other Locators

You will likely need other locators as well, for things like waypoints, triggers, and so on. These can use existing locators you've already placed, for things like spawners or garrisons. Or if you need a special one you can just place a **`LocatorClass`** object. Name it what you want, add it to the **`ConfigurationController`** like anything else, and you'll be able to see and reference it from the **`AreaSpecification`** when you need to.

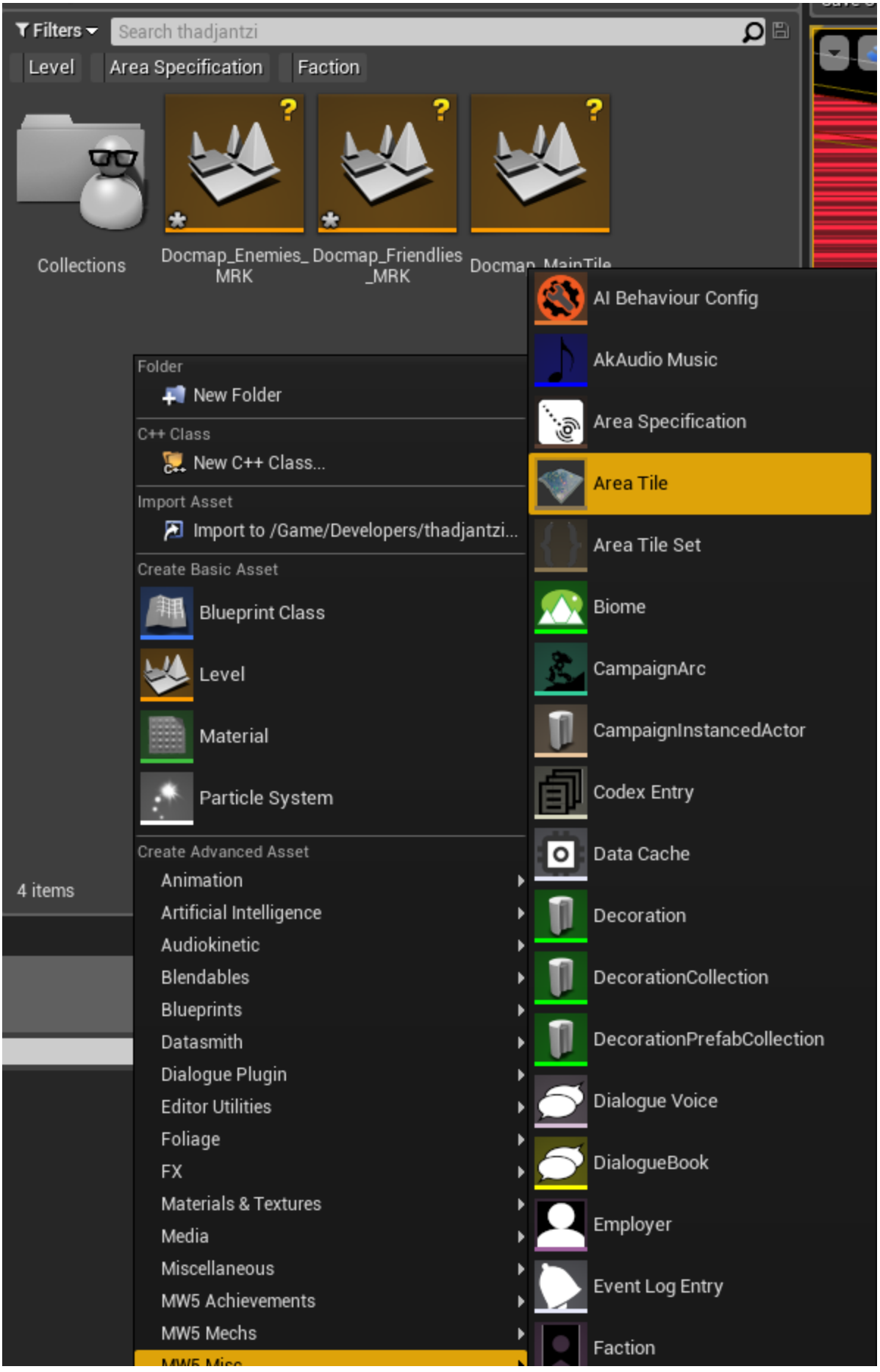




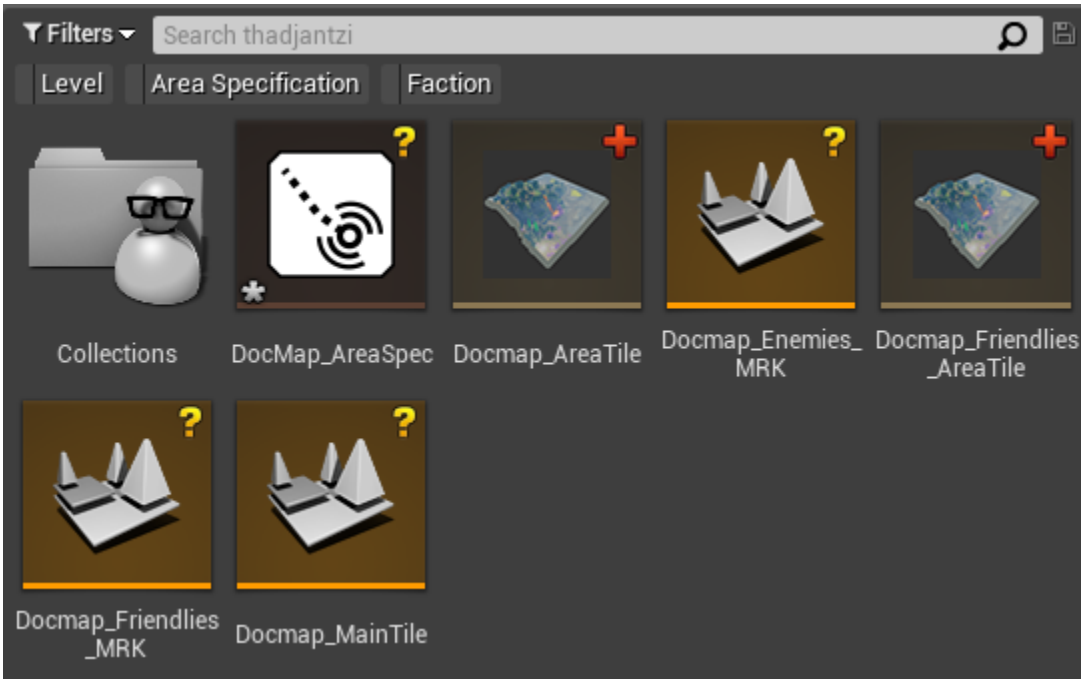
Step 7: AreaTiles

Ultimately the way we will refer to our levels in the mission is by **`AreaTile`** objects. We will need two of these for our mission – one for the level itself and the enemy markup level, the other for the friendly markup (just the markup).

To create one, go into the content browser and right click on the blank space, then under the **`MW5 Misc`** sub menu, select **`AreaTile`**.


Filters
Search thadjantzi
Level
Area Specification
Faction
Collections
Docmap_Enemies_MRK
Docmap_Friendlies_MRK
Docmap_MainTile
4 items
Folder
New Folder
C++ Class
New C++ Class...
Import Asset
Import to /Game/Developers/thadjantzi...
Create Basic Asset
Blueprint Class
Level
Material
Particle System
Create Advanced Asset
Animation
Artificial Intelligence
Audiokinetic
Blendables
Blueprints
Datasmith
Dialogue Plugin
Editor Utilities
Foliage
FX
Materials & Textures
Media
Miscellaneous
MW5 Achievements
MW5 Mechs
AI Behaviour Config
AkAudio Music
Area Specification
Area Tile
Area Tile Set
Biome
CampaignArc
CampaignInstancedActor
Codex Entry
Data Cache
Decoration
DecorationCollection
DecorationPrefabCollection
Dialogue Voice
DialogueBook
Employer
Event Log Entry
Faction

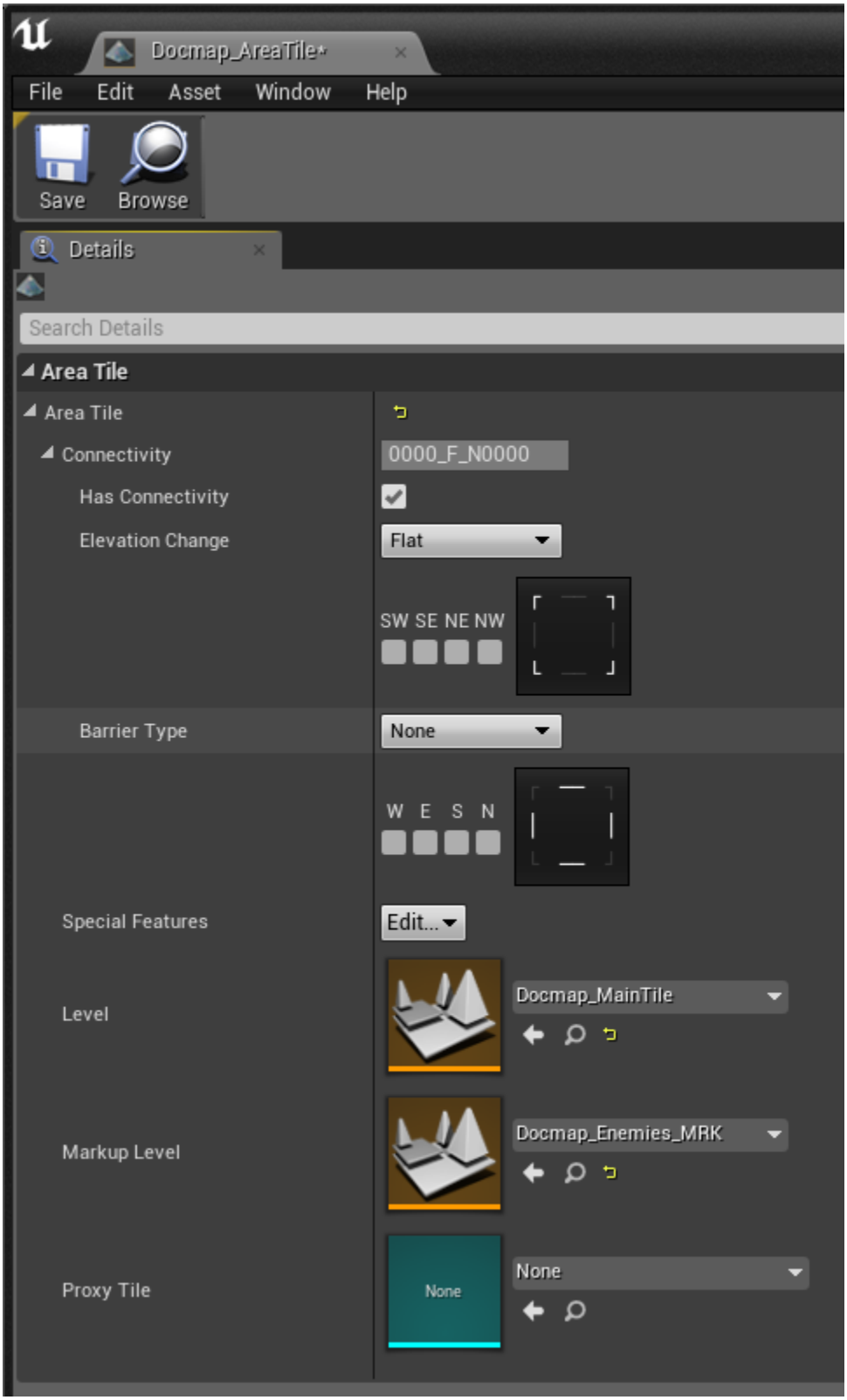
Once you have your **`AreaTiles`** made, name them something like **`MyLevel_AreaTile`** and **`MyLevel_Friendlies_AreaTile`**.

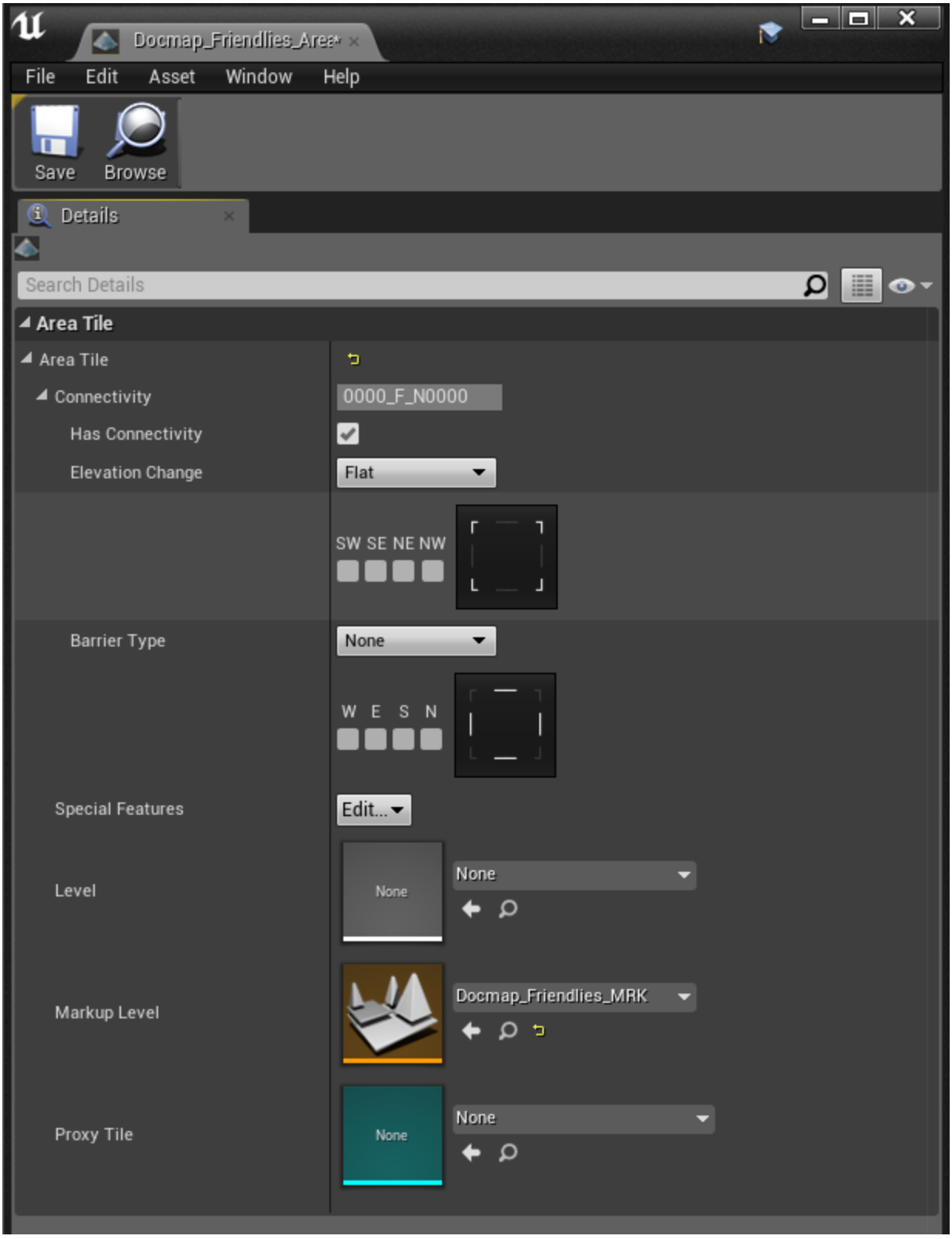


Now double click the **`MyLevel_AreaTile`** and go into it. There are a bunch of settings that we only need to care about if this tile needs to be part of a connected tileset for proc missions. Since this does not, we can ignore everything except the Level and Markup Level.

For **`MyLevel_AreaTile`** you will need set those as your main level and your enemies markup level, respectively.

For **`MyLevel_Friendlies_AreaTile`**, the only thing in that you will need to set is the markup level, which will be your friendlies markup level.


Docmap_AreaTile*
File
Edit
Asset
Window
Help
Save
Browse
Details
Search Details
Area Tile
Area Tile
Connectivity
0000_F_N0000
Has Connectivity
Elevation Change
Flat
SW SE NE NW
Barrier Type
None
W E S N
Special Features
Edit...
Level
Docmap_MainTile
Markup Level
Docmap_Enemies_MRK
Proxy Tile
None
None

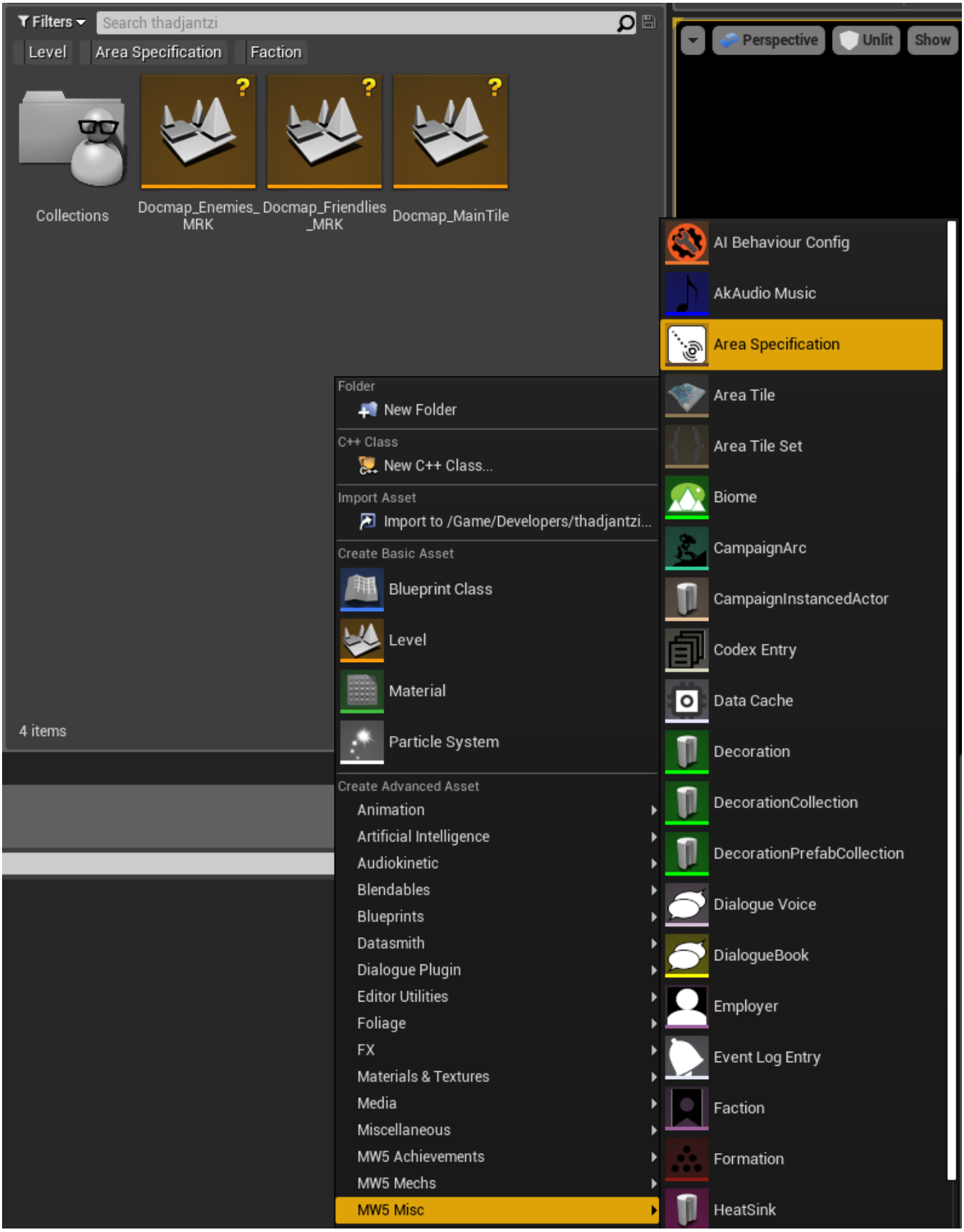


That's all we need to do in the **`AreaTiles`**. At the bottom you have the option to add a proxy tile. This is also something we use mainly for proc mission tiles, for those tiles it is a lo-res static mesh that just illustrates the very basic structure of the tile so we can see how the connectivity looks in a level comprised of a matrix of different tiles. Since our level is just one tile, we don't really need to make one.

Step 8: AreaSpecification

Okay now we've created all the little files that we need to make the mission. From this point on, 99% of our work will take place in the **AreaSpecification** (hereafter "**AreaSpec**").

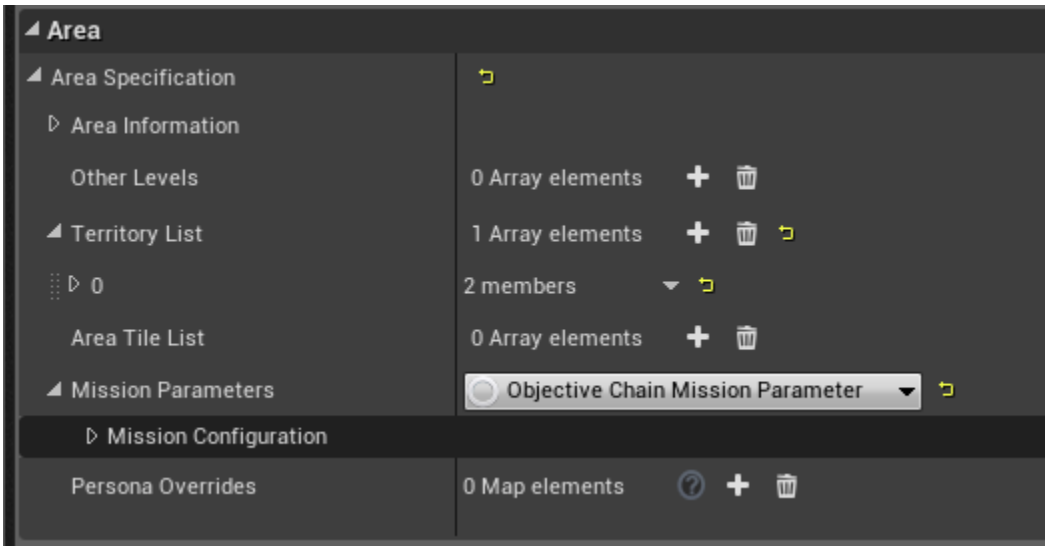
To create your **AreaSpec**, right click in the content browser and under **MW5 Misc** select **Area Specification** and name it.



Note: When working in your **`AreaSpec`**, you should not have a level loaded in the editor. If you have a markup level loaded in the editor, that level's locators will not be available in your **`AreaSpec`**. Before you work in an **`AreaSpec`** you should do **`FileNew`** and just have an empty level in

the editor. If you load up an **`AreaSpec`** some day and notice a slew of red text and missing locators, make sure you don't have the corresponding markup level loaded in the editor.

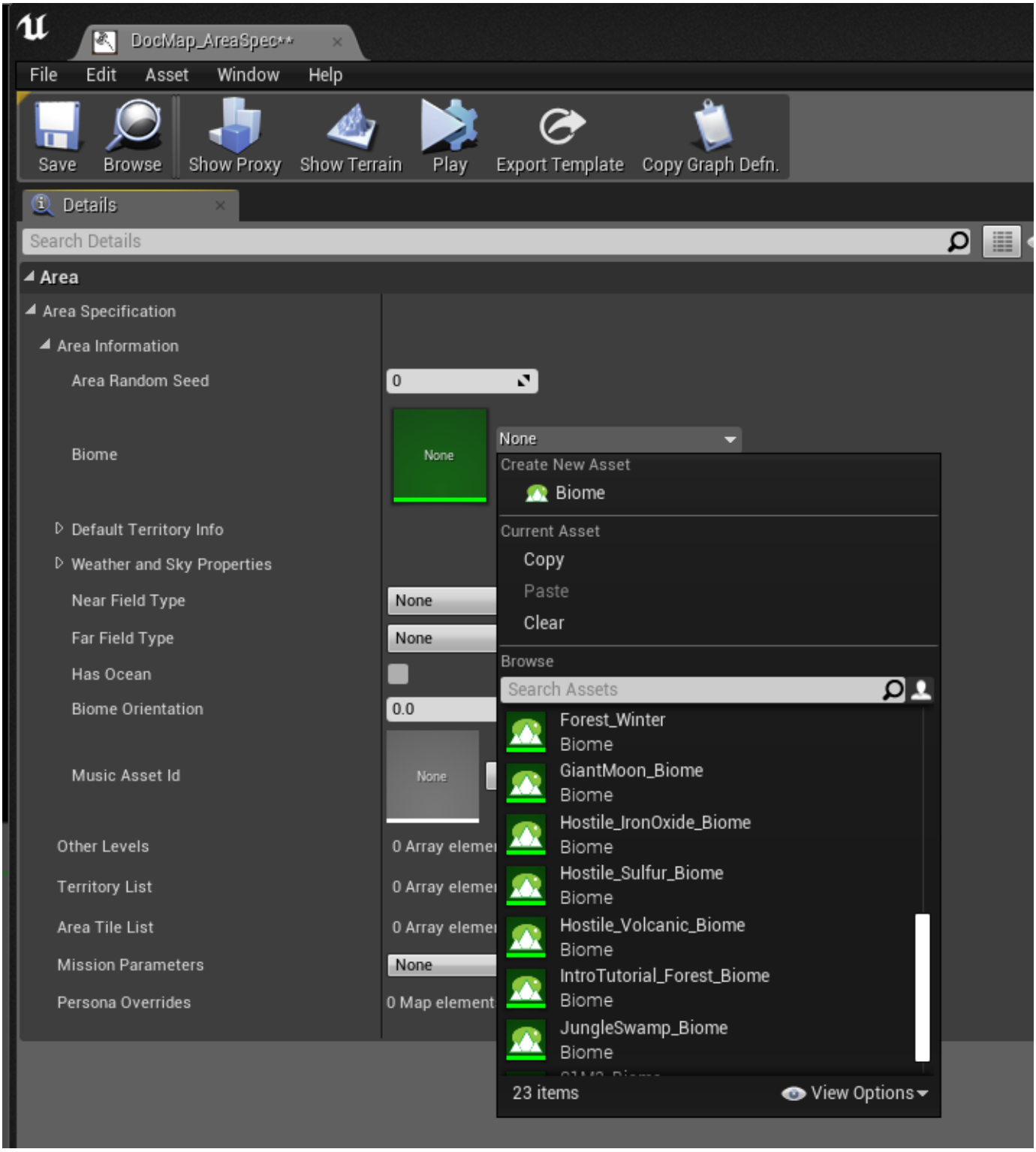
Go down to the bottom of the **`AreaSpec`** right away and set **`Mission Parameters`** to **`Objective Chain Mission Parameter`**. If you don't, nothing will work.



Area Information

Biome

One of the most impactful choices you'll make concerning how your level looks is choosing the biome. You can set it to one of the existing biomes, or you can make a custom one for this mission only. If you want to do that, the easiest way to accomplish it is probably to duplicate one of the existing biomes (the one closest to how you want your mission to look), give it a name specific to your mission and then just make adjustments to that one as you see fit.



Weather and Sky Properties

This panel is filled with settings whose names are pretty self-explanatory. Ultimately it's best to just try them all out and see what you like and what you don't like. I'll only describe a couple of them:

- **`FreezingLevel`** is the Z coordinate above which snow will appear on the ground.

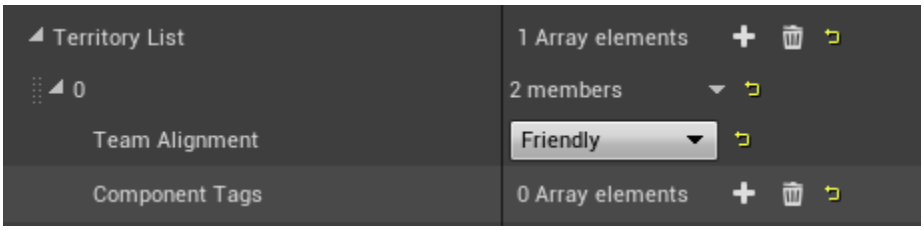
- **`NearFieldType`** (full/half/none) refers to the background mesh (likely mountains, or something). Full means that it will appear on all sides of the level. Half means it will appear only on one side (this is useful for if you want to have an ocean on one side of your level). None means none.

- **`HasOcean`** (true/false) refers to whether there is or isn't a flat plane extending out on all sides of the level. For some biomes this plane is an ocean, for other ones it's just a flat desert. This checkbox simply dictates whether or not the appropriate "ocean" plane will appear.

- **`BiomeOrientation`** can be set to a number of degrees (0-360) that the background level and lighting level will be rotated. Most useful for setting the light direction.

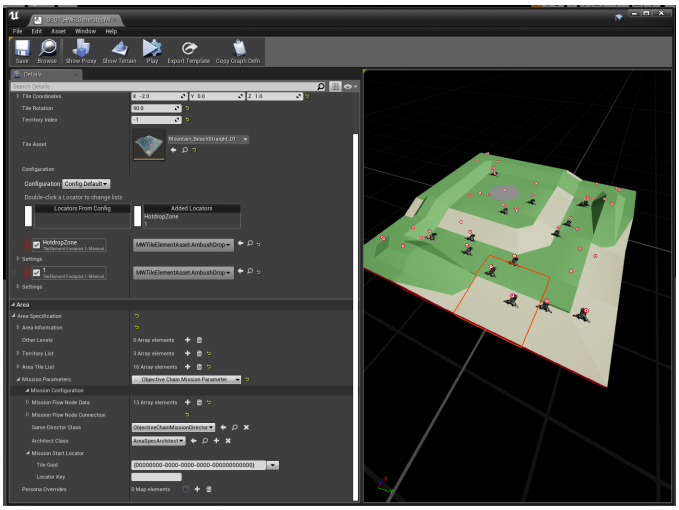
Territory List

Add one entry to this array, and set the **`Team Alignment`** to "**`Friendly`**". This will let us have a friendly tile (the friendlies markup level we made earlier) in addition to an enemy tile. By default, unless set to this friendly territory, any **`AreaTile`** you add will be designated as enemy.

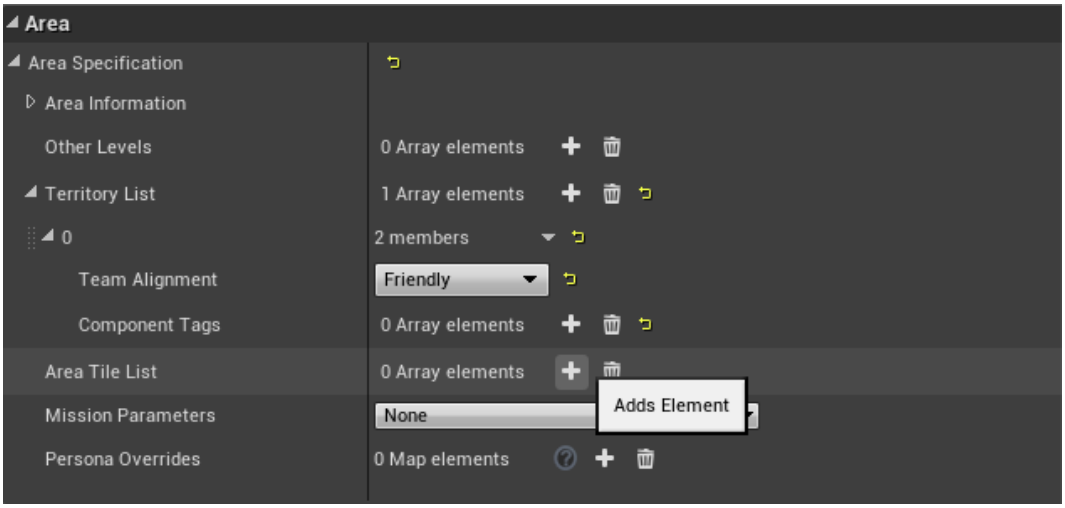


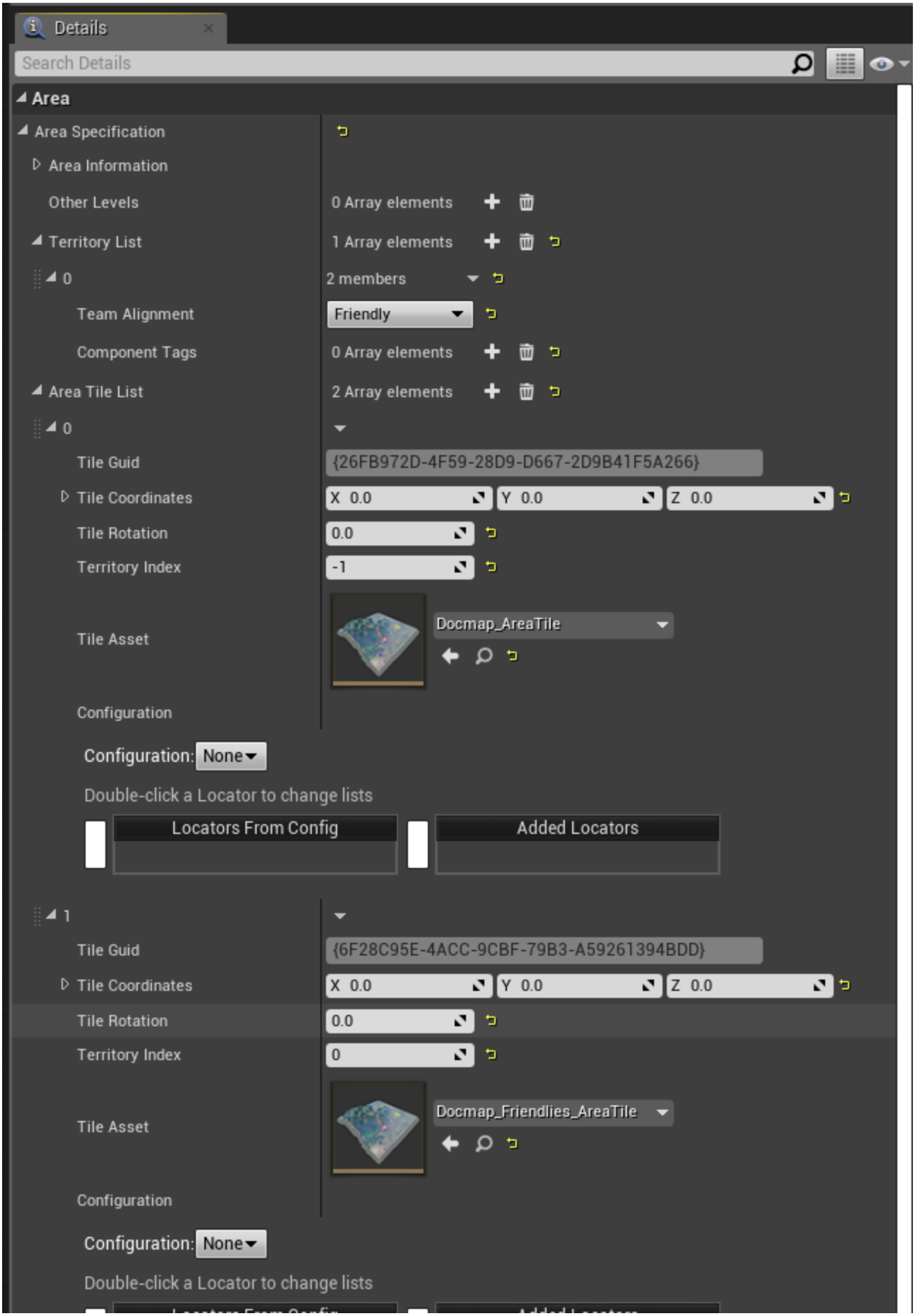
Area Tile List

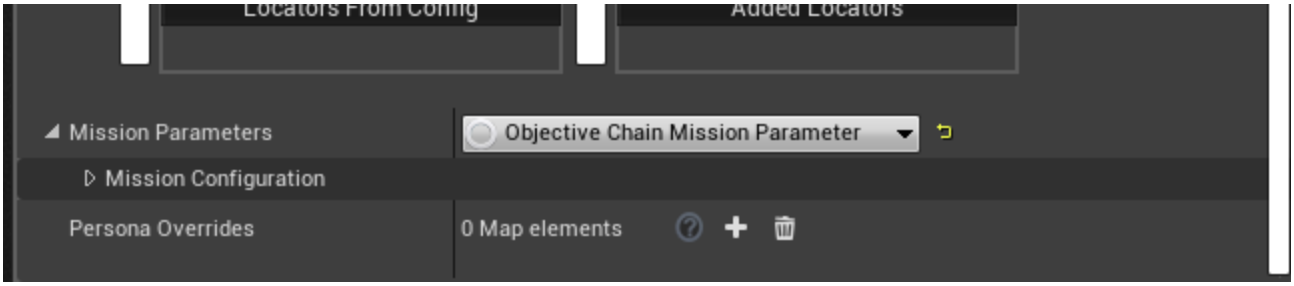
This is where we specify what tiles are included in the level, and their position and orientation. If you were to view the **`AreaSpec`** from a proc mission, this would have dozens of tiles, and their proxy meshes would all appear in the viewport of the **`AreaSpec`** editor.



For our mission we will only need to add two tiles, the two **`AreaTile`** objects we made earlier. Add two entries to the **`AreaTileList`**. Set one to your main **`AreaTile`** object, and the other to your friendlies **`AreaTile`** object. For the friendlies one, set the **`TerritoryIndex`** to 0 (that's because the friendly territory you defined in the Territory List has an index of 0.


Area
Area Specification
Area Information
Other Levels
0 Array elements
Territory List
1 Array elements
0
2 members
Team Alignment
Friendly
Component Tags
0 Array elements
Area Tile List
0 Array elements
Adds Element
Mission Parameters
None
Persona Overrides
0 Map elements


Details
Search Details
Area
Area Specification
Area Information
Other Levels
0 Array elements
Territory List
1 Array elements
0
2 members
Team Alignment
Friendly
Component Tags
0 Array elements
Area Tile List
2 Array elements
0
Tile Guid
{26FB972D-4F59-28D9-D667-2D9B41F5A266}
Tile Coordinates
X 0.0
Y 0.0
Z 0.0
Tile Rotation
0.0
Territory Index
-1
Tile Asset
Docmap_AreaTile
Configuration
Configuration: None
Double-click a Locator to change lists
Locators From Config
Added Locators
1
Tile Guid
{6F28C95E-4ACC-9CBF-79B3-A59261394BDD}
Tile Coordinates
X 0.0
Y 0.0
Z 0.0
Tile Rotation
0.0
Territory Index
0
Tile Asset
Docmap_Friendlies_AreaTile
Configuration
Configuration: None
Double-click a Locator to change lists

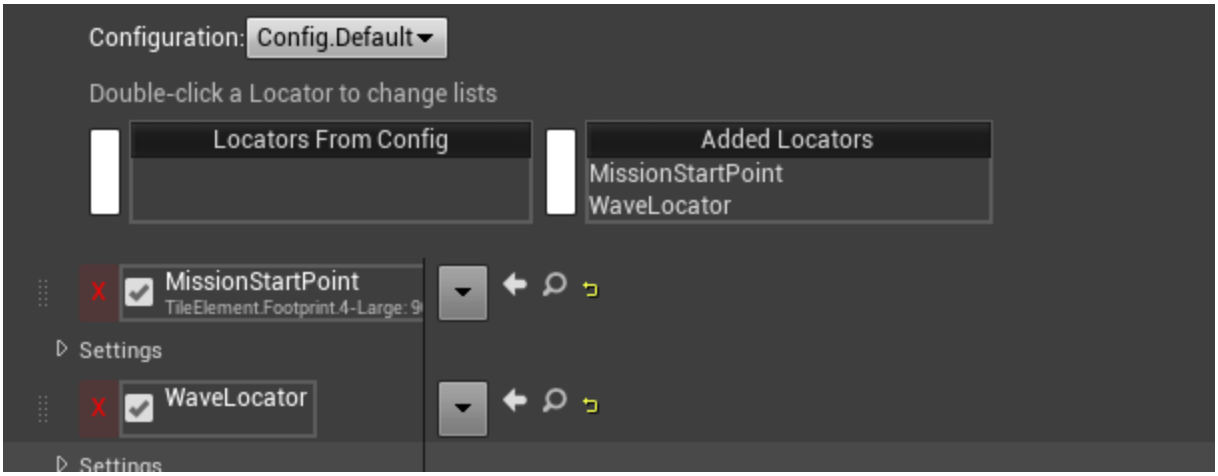


Tile Configuration

Once the **`AreaTile`** object is specified, you need to specify a Configuration for it. If you set the **`ConfigurationControllers`** up as I described above, you need to set this to **`Config.Default`**. At this point all of the locators that you put into your markup level should now be listed under **`Locators From Config`**.

Adding Locators

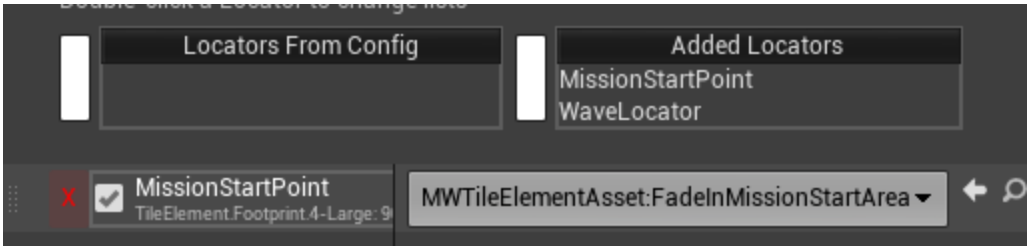
To add the locators into your mission, double click each one in the list. They will then switch to the **`Added Locators`** column, and an entry will appear below the panel. In addition, a red dot will appear in the viewport, denoting that locator's position in 3D space.



Each locator will have a dropdown box to the right of it, where you can specify which **`TileElement`** it will load. A **`TileElement`** is an object which will specify levels to be loaded into the main level, such as garrisons.

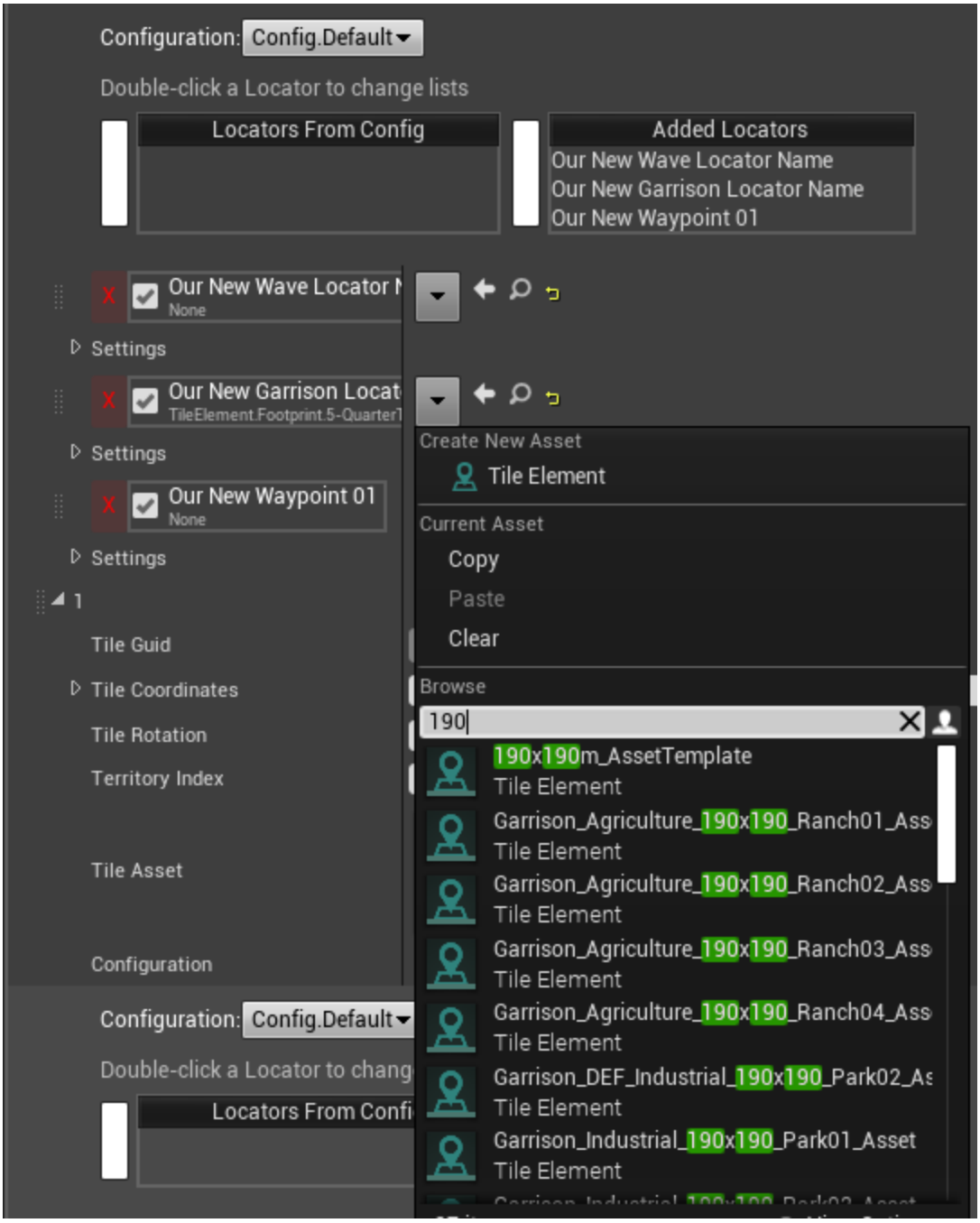
Mission Start

For your **`DropshipLandingZone`** locator, select either **`FadeInMissionStartArea`** or **`LeopardMissionStartDropZoneLandingArea_01`**.



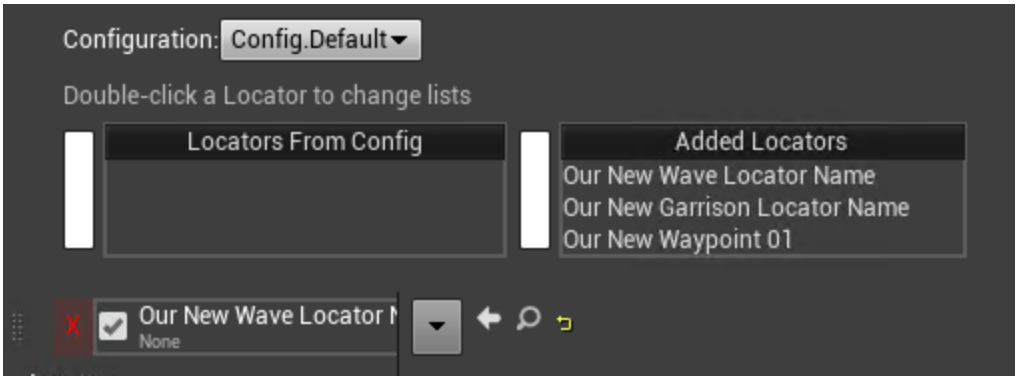
Adding Garrisons

To add garrisons, find the locator you placed in the list, and select the garrison's tile element from the dropdown list.



Locators With No Tile Element

Some locators will not have any **`Tile Element`** associated with them. Things like **`WaveLocators`** for spawning AI, or locators that you've only placed as a waypoint or a trigger. They will still show up in the viewport and can be used, and you do not need to specify a **`Tile Element`**.



Adding AI Units

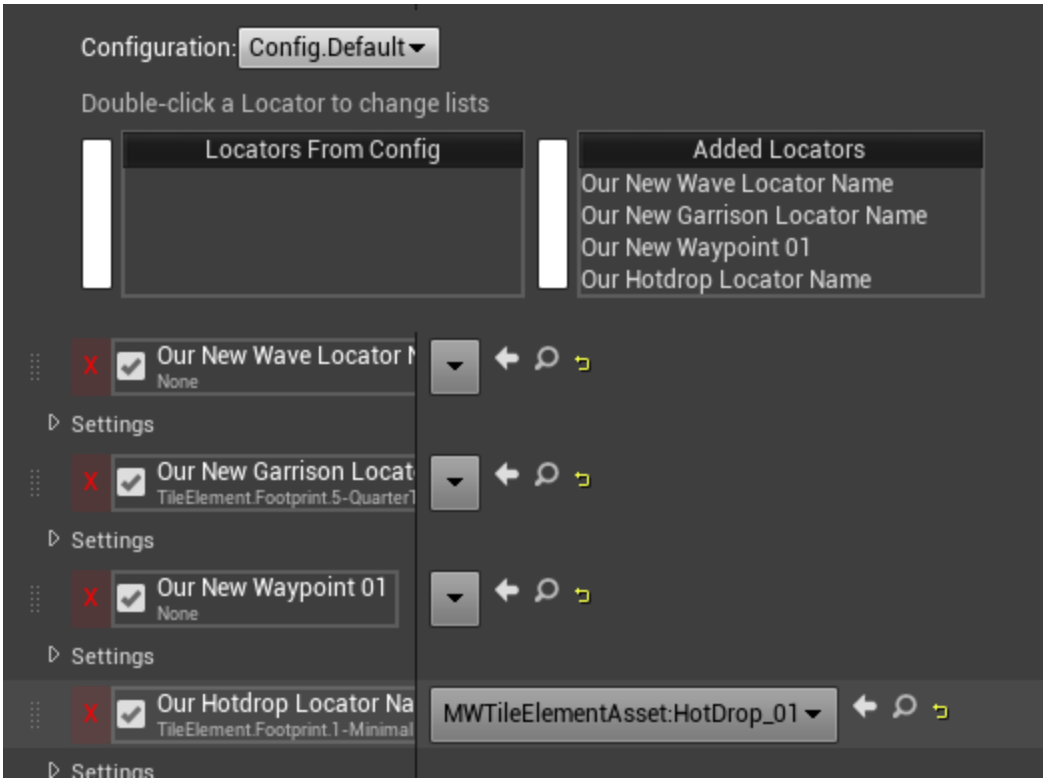
Under each locator you added, there is a Settings arrow that can be expanded to reveal a **`Unit Deck`** entry. This is where we add the AI units. First figure out which locator you want to spawn units from (whether it be inside a garrison, or from a **`WaveLocator`**, or **`HotDropLocator`**). Then go to the **`Unit Deck`** entries for that specific locator and click the + sign to add as many units as you will want to spawn from there.

For each **`Unit Deck`** entry you add a **`Unit Card`**, indicating specifically what type of unit you want to spawn. Here you can also set a **`Faction Asset ID`** to specify the unit's faction, and you can add quirks to the unit such as **`Pilot Skill Level`** or **`Weapon Tech Level`**.

Here you also need to manually generate a **`Unit GUID`** so that you can refer to the unit later. You can do that just by clicking the arrow next to the **`Unit GUID`** field.

If you name your spawn points, you do have the ability here to specify a particular one for a unit to spawn at. If you do not bother with this, they'll just choose for you. I think it just goes in order of when the spawns were created.

If you want to spawn units from a **`HotDropLocator`**, you will need to specify the **`HotDrop_01 TileElement`** for that locator in the **`AreaTileList`**. (Unlike **`WaveLocators`** which do not require a **`TileElement`**).



Mission Parameters

Under mission parameters there are two important sections, **`Mission Flow Node Data`** which is where we will set up all the gameplay elements of the mission, and **`Mission Flow Node Connection`** which is where we will set up all the logic that connects them together.

Mission Flow Node Data

Every element of gameplay for missions from spawning to waypoints to mission objectives gets defined in these **`Mission Flow Nodes`**. There will be a separate document available with complete reference for all of these nodes, but I will do a quick write up here for about elements that are common to all of them, and a quick description of the most commonly used ones.

- **Name:** this property is the name by which you will reference this node everywhere, it is important to make sure that it is meaningful and unique.

- **Start on Setup:** mission components are not active by default, they need to be started. Most of them you will start as a result of completing some other node (and these connections will be described below) but for those you wish to start when the mission starts, check this box.

- **Location:** the specific locator this node refers to. Some nodes have a locator, some do not. The data refers to the **`Tile GUID`**, found in the **`Area Tile List`**, and the **`Locator Key`** (the name of the locator you entered when adding the locator to the **`ConfigurationControlle`** in the markup level). An easy way to enter these is to find the locator (red dot) in the viewport, right-click on it (which is effectively pressing CTRL-C to copy it) then right-click on the word "**`Locator`**" under **`Location`** in the Flow Node, and select Paste. That will copy and paste it perfectly and prevent any errors on the data entry.

- **Node Types:** here you can set what type of objective this node represents, if any.

    1. **Primary Objective / Secondary Objective:** both are necessary to successfully complete the mission, but are listed in separate sections on the HUD
    2. **Optional Objective:** will be listed on the HUD but has no bearing on whether the mission is deemed successful
    3. **Unlisted Objective:** this can be useful for making an objective that will show some HUD elements, like a waypoint, but you do not want a text description on the HUD or for it to be necessary for mission completion
    4. **Non-Objective:** not an objective, obviously. Most nodes will be non-objective.

- **Clear from UI on complete:** if it's an objective, do you want the objective text removed from the HUD when complete

- **Parent Mission Flow Node:** I believe this is useful only for the container node

- **Definition Tags:** this is where you define (using tags) which type of component this node is going to be

- **Additional Mission Data:** this is where most data that is unique to the various flow nodes goes. The values/formats you'll want will be documented in the flow node documentation.

- **Dispatch Data:** this is where data concerning the spawn points/targets for AI units goes

- **Dialogue Script:** this is where we can specify audio dialogue to be triggered when this node is started, completed, failed or aborted

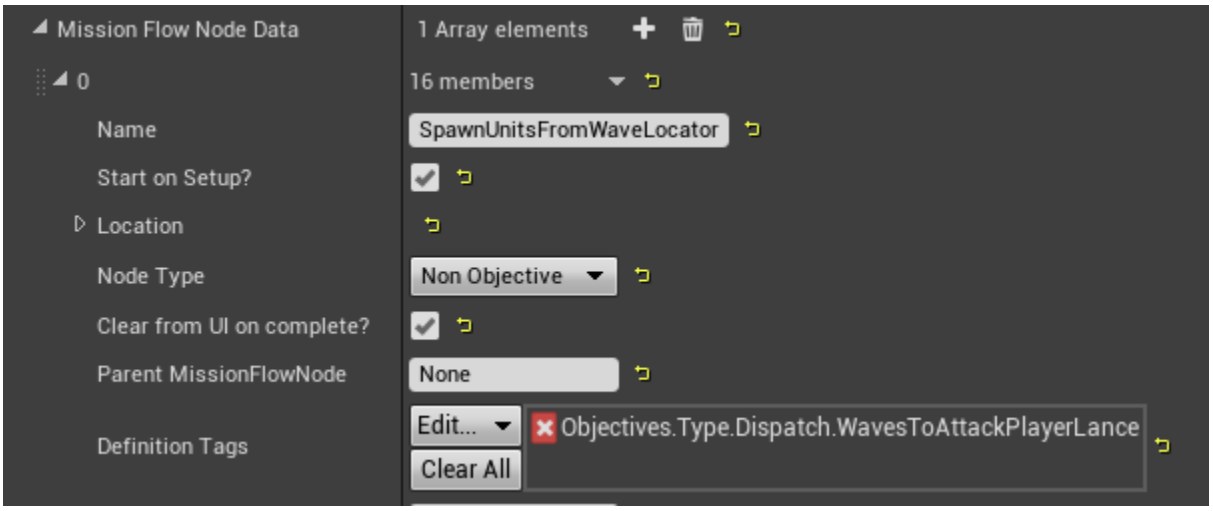
Spawning AI (Objectives.Type.Dispatch…)

To spawn AI units, use one of

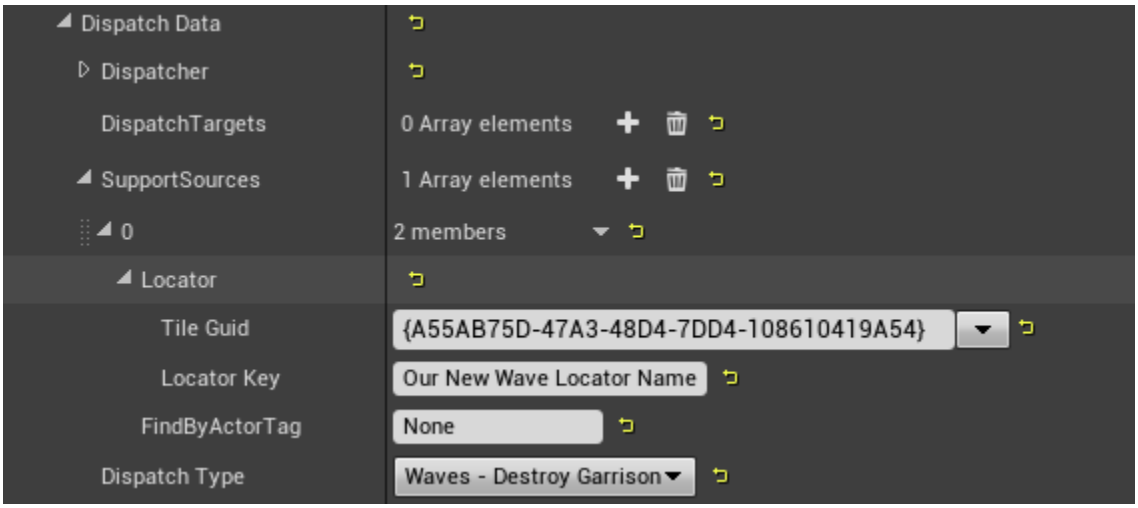
- **`Objectives.Type.Dispatch.WavesToAttackPlayerLance`**

- **`Objectives.Type.Dispatch.WavesToDestroyGarrison`**

- **`Objectives.Type.Dispatch.WavesToGoToTargetLocation`**

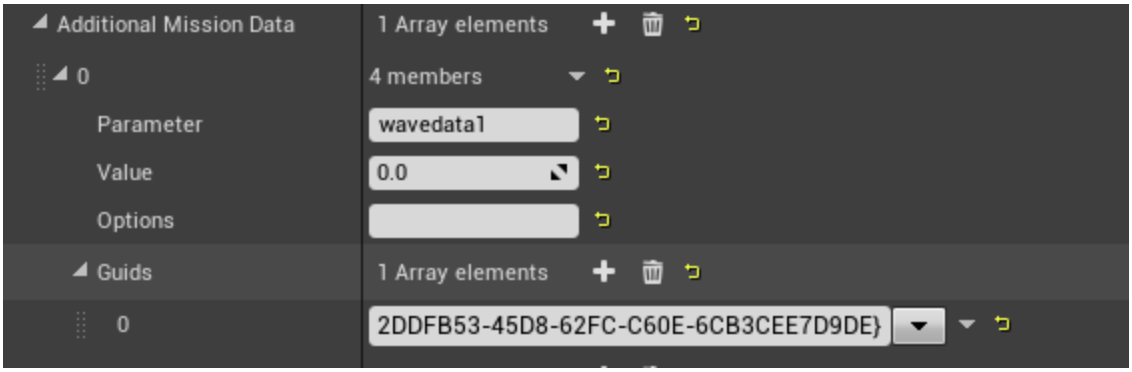


This node does NOT require a Location, rather you specify the relevant **`WaveLocator`** (or **`HotDropLocator`**) under **`Dispatch Data -> SupportSources.`**



If dispatching the waves to attack a garrison or move to a location, specify the target locator under **`Dispatch Data -> DispatchTargets`**.

To specify which units (already having been defined in the **`AreaTileList`**) this node is to spawn, create an entry under **`Additional Mission Data`** with parameter "**`wavedata1`**", and under that add entries to the Guids list, and copy the Guids for the corresponding units in the **`AreaTileList`**.



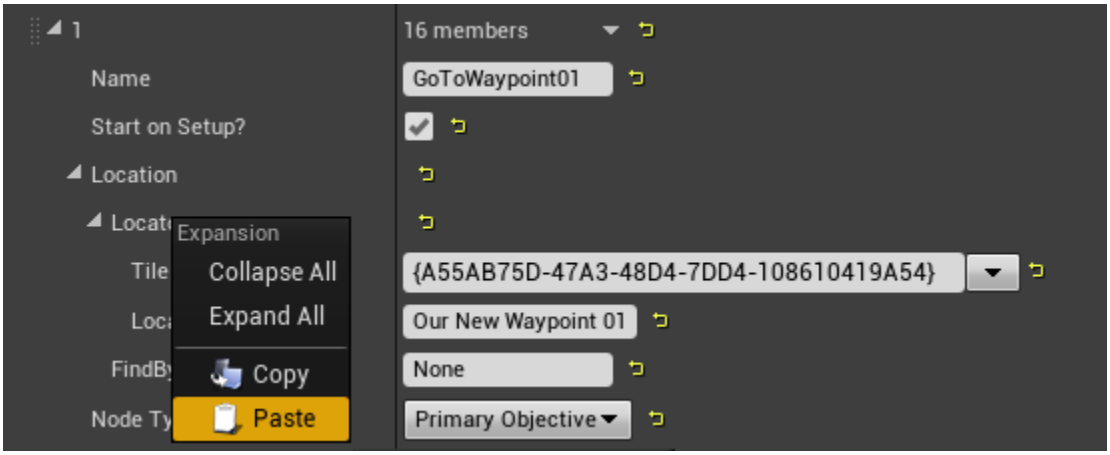
Now this is all set. When the node is started, the units will spawn.

You can specify multiple waves with different conditions for spawning as well. Refer to the Flow Node documentation for more advanced details.

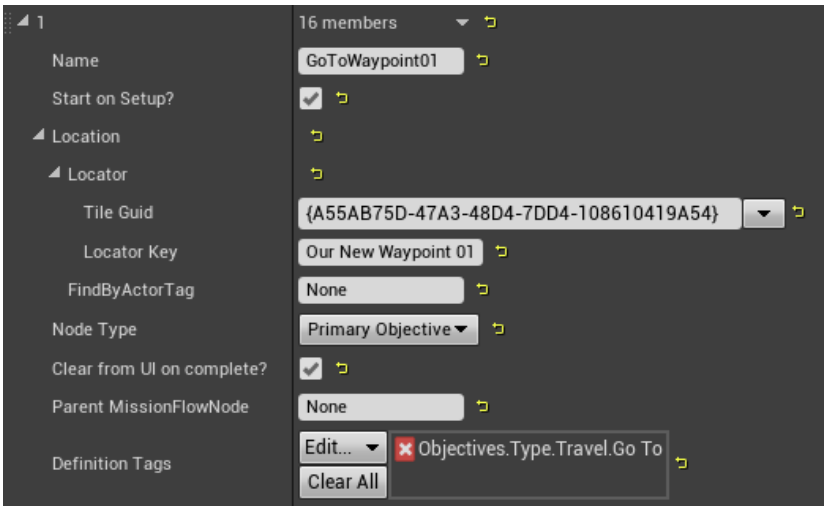
Units specified in a garrison, rather than in a **`WaveLocator`** or a **`HotDropLocator`** will just be there on mission start, no need to spawn them with a flow node.

Area Triggers / Waypoints (Objectives.Type.Travel.Go To)

To create a node that will be triggered when the player enters a certain area you only need a locator placed, it doesn't matter what type of locator. Set the **`Location`** to the locator you want to act as the waypoint. Remember you can copy it from the viewport using the right mouse button, and then paste into the **`Locator`** under "**`Location`**".

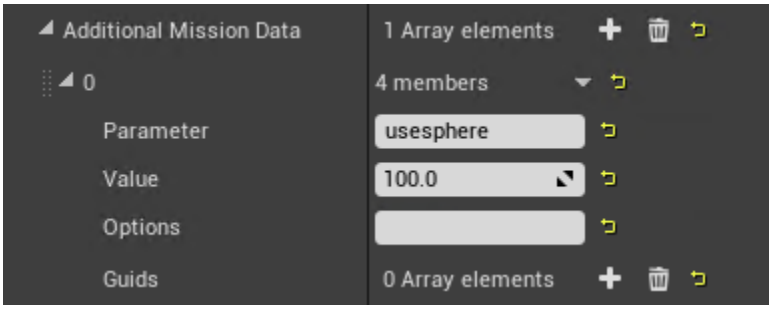


For this the node definition is **`Objectives.Type.Travel.Go`** To

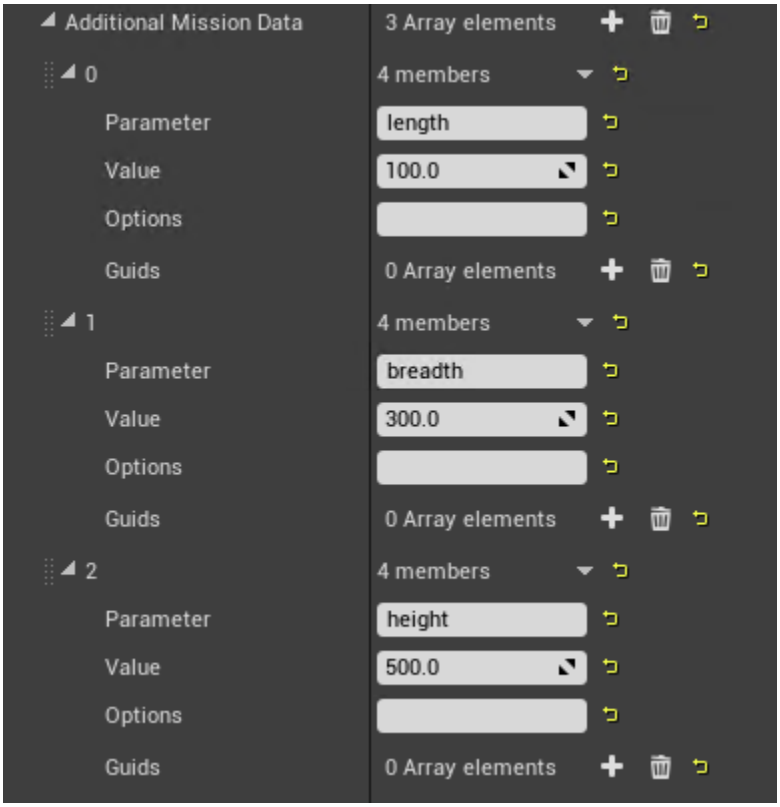


The trigger can either be a sphere or a rectangular volume.

For a sphere create an entry under **`Additional Mission Data`** with the parameter "**`usesphere`**" and the Value equals the radius of the desired sphere in meters (ie: units / 100)



For a rectangular volume, create three entries under **`Additional Mission Data`** with the parameters "**`length`**", "**`breadth`**" and "**`height`**" and the Values also in meters.



Once you do this you should be able to see the size of the trigger in the viewport by hovering the mouse over the relevant locator.

There are three different ways to do a Travel.GoTo trigger:

1.) you want to see the yellow map marker and have it listed among the objectives, you set it to an Objective type: (**`Primary/Secondary/Optional`**)

2.) you want to only see the yellow map marker, but NOT any text, you set it to **`Unlisted Objective`**

3.) you want it to trigger, but not show the player anything, you set it to **`Non-objective`**

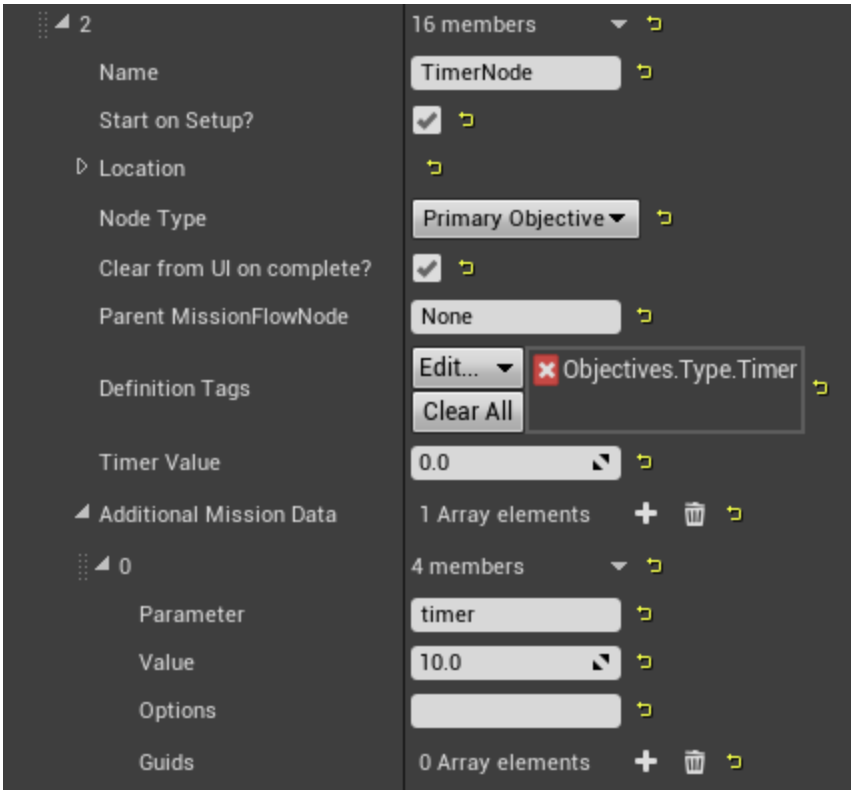
Once this is set up you have a flow node that will be considered complete when the player enters the trigger. Note it won't be active until it's started, so if you want it present from the get go make sure to check **`Start On Setup`**.

Timers (Objectives.Type.Timer)

To set a timer, use **`Objectives.Type.Timer`**

The only value you need to set is to create an entry called "**`timer`**" under **`Additional Mission Data`**, and set the Value to the time in seconds.

Note: there is a "**`Timer`**" property in the node, don't use it. You have to make the "**`timer`**" entry in **`Additional Mission Data`**.



This timer will start counting down when you start the node, and will be considered successfully complete when the countdown reaches 0.

Destroy a Unit or Units (Objectives.Type.Destroy.Unit)

To create a node which will be satisfied when a particular unit or units are destroyed, use **`Objectives.Type.Destroy.Unit`**.

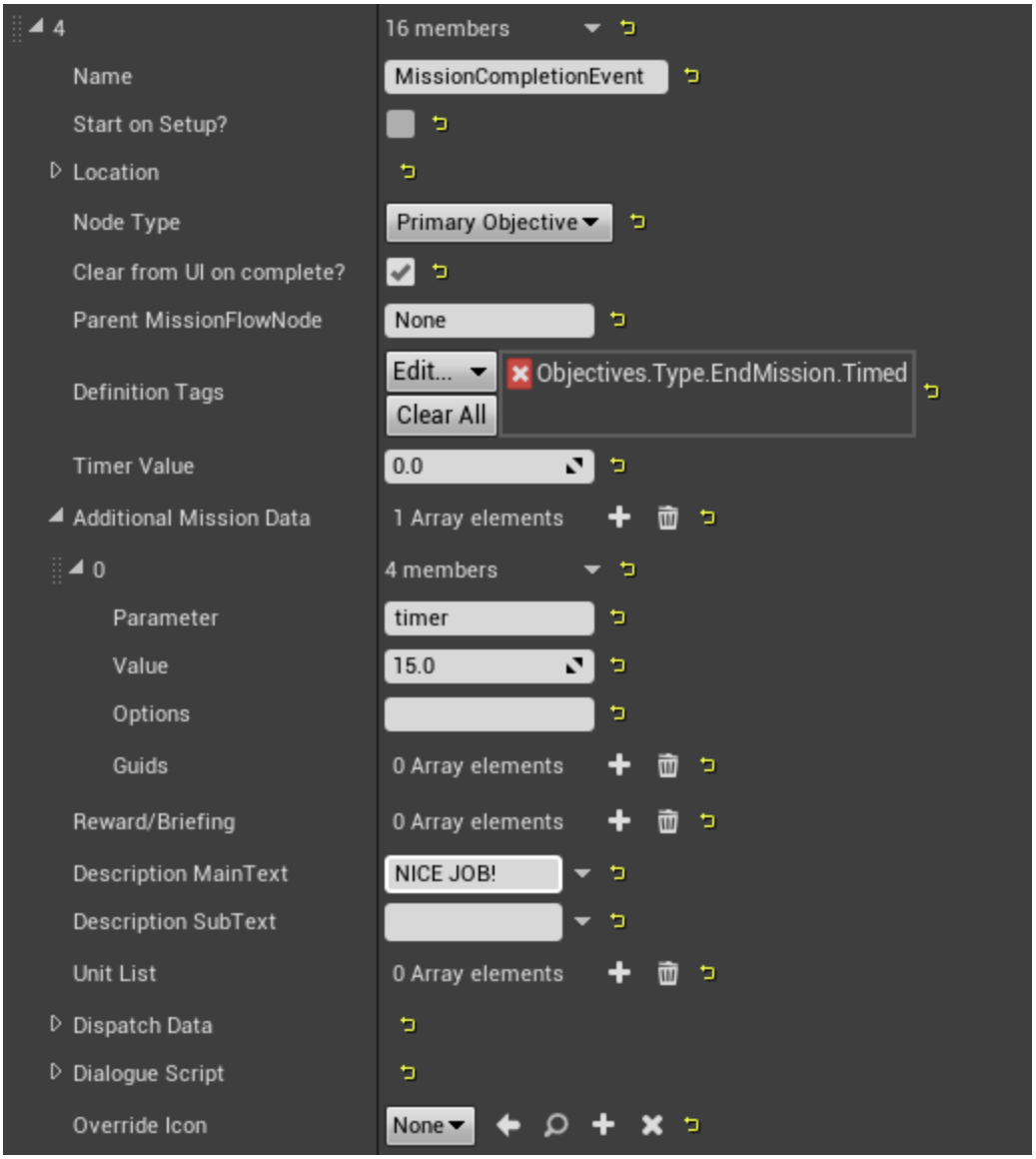
For this you need only specify the units by making an entry in the **`Additional Mission Data`** section with the parameter name "**`Units`**" and a list of Guids below.

Once this is activated, the units will be shown on the HUD with a skull icon. If you don't want that, you can add another entry to **`Additional Mission Data`** with the parameter name "**`nomarker`**".

End Mission (Objectives.Type.EndMission.Timed)

To end the mission, simply trigger an **`Objectives.Type.EndMission.Timed`** node.

To set a timer to wait x seconds after triggering the node, add an entry named "**`Timer`**" into the **`Additional Mission Data`**, with a value of the desired number of seconds. Typically you could trigger some mission ending dialogue when this node is started, and you'd want the timer to allow enough time for the dialogue to complete.



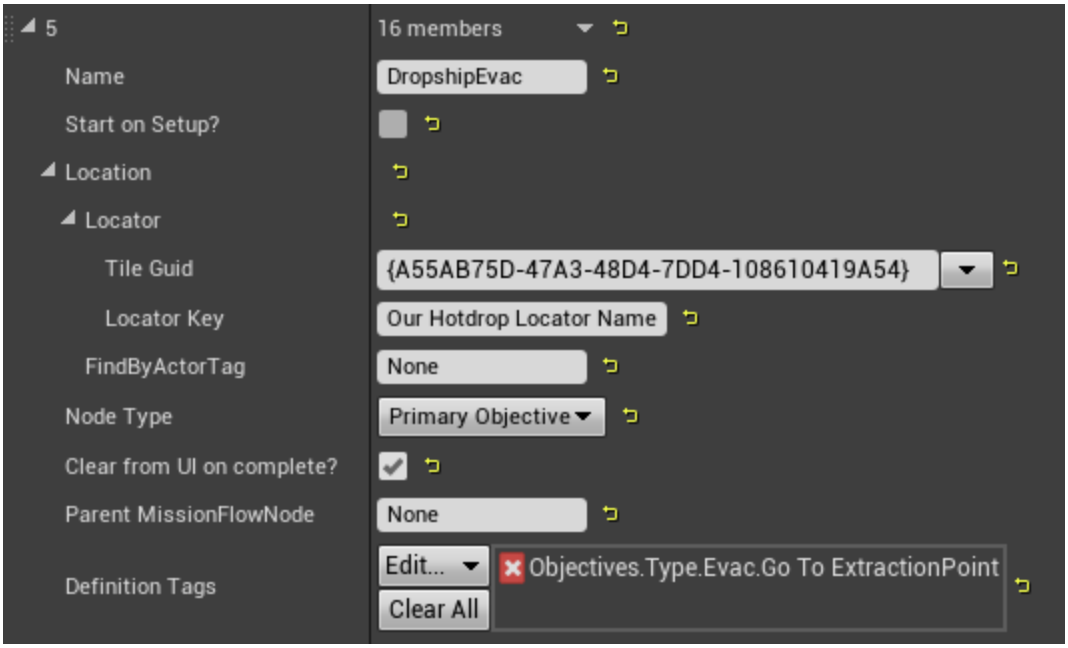
If you want the node to end the mission in a state of failure, add an **`Additional Mission Data`** entry with the parameter name "**`Failure`**".

End Mission with a Dropship Pickup (Objectives.Type.Evac.Go To ExtractionPoint)

To have a dropship arrive to pick the player up, use **`Objectives.Type.Evac.GoToExtractionPoint`**.

This needs to take place at a **`HotDropLocator`** that has been assigned the **`Pickup_01 TileElement`**.

The **`Location`** will need to be set to that locator as well. Once the node is active, the player can arrive at the pickup point, the dropship will arrive and the mission will end.



Some Notes on Mission Flow Nodes

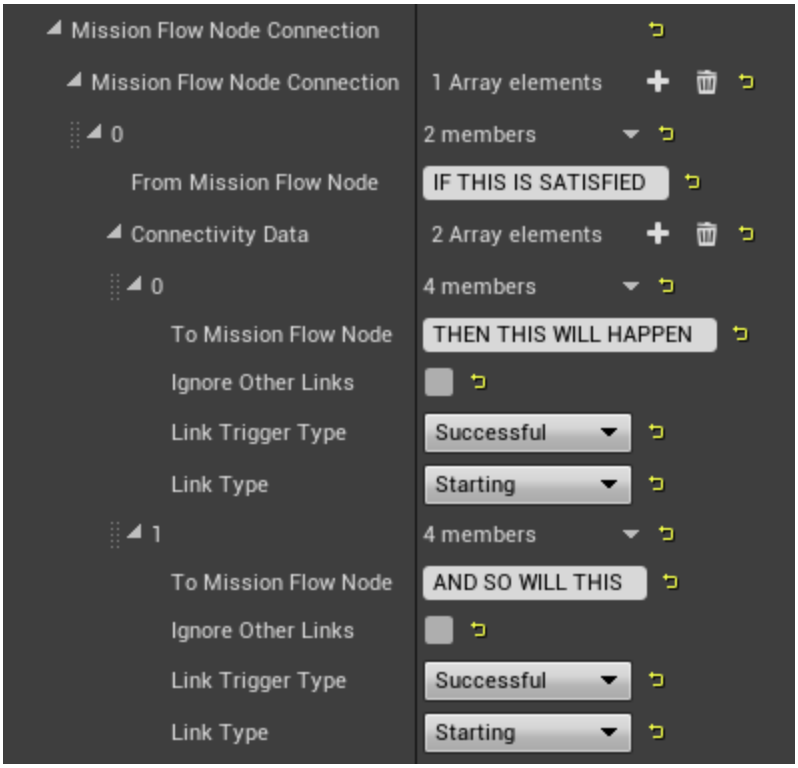
This is by no means an exhaustive list of the different mission flow nodes, but rather a quick description of the most commonly used ones and how to set them up. For a much more comprehensive look at the various nodes you can look at the Mission Flow Node Scripting Reference document. Another valuable reference would be to open up the **`AreaSpecs`** from the missions in the game. All of the story missions and cluster missions should have one – the quickest way to find them is to just go to the root path of the content browser and filter by **`Area Specification`** . If you see something in a story mission you want to do in your mission, that is a good way to find out how it was implemented.

Mission Flow Node Connection

Below the **`Mission Flow Node Data`** are the equally important **`Mission Flow Node Connections`**. This is where we define the logical relationship between all of the nodes.

Each entry has a field entitled "**`From Mission Flow Node`**", this is where you put the name of the flow node whose state you want to evaluate (ie: the flow node we are checking to see is complete, or failed, or aborted).

For each of these entries you can add an arbitrary number of entries called "**`To Mission Flow Node`**". These are the nodes whose state the "**`From Mission Flow Node`**" will be affecting (ie: then we will start this node, or fail this node, or abort this node, or successfully complete this node).



`Link Trigger Type` means: “when the From Mission Flow Node is in THIS state, we will trigger the link”

`Link Type` means: “this is the state we will now apply to the To Mission Flow Node(s)”

In the following example, we have three nodes - **`GoToWaypoint01`**, **`SpawnEnemies`**, and **`DestroyEnemyUnits`**. When the player reaches **`Waypoint 01`**, we would trigger both the **`SpawnEnemies`** node to spawn the enemies, and the **`DestroyEnemyUnits`** node to give the player an objective to destroy the enemy units.

One-to-Many Relationships

Since you can add many entries for "**To Mission Flow Node**" on a single "**From Mission Flow Node**", having the resolution of one node affect several nodes is very simple. Just add the entries logically. Both of the examples I showed above have one "From" node and two "To" nodes.

Many-to-One Relationships

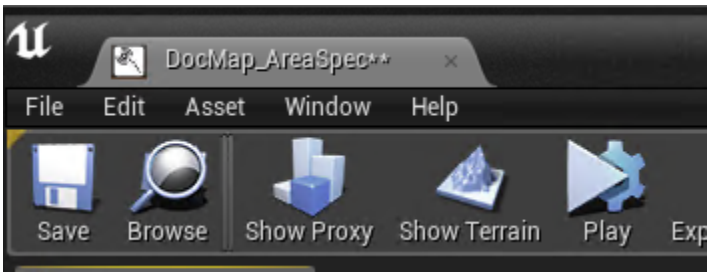
This is a little trickier – if you want the results of several flow nodes to affect a single flow node – for example "I want **Node_D** to start only after **Node_A**, **Node_B**, and **Node_C** are complete", you need to make "**From Mission Flow Node**" entries for each of **Node_A**, **Node_B** and **Node_C** and all of them must have a "**To Mission Flow Node**" entry for **Node_D**. If is set up like that, then **Node_D** will only be triggered once all of the other three are completed.

Let's say that is fine, but you also have another node called **Node_E** that is way more important to the mission than **Nodes A, B** and **C** and you want **Node_E**'s completion to trigger **Node_D**, regardless of what happens in **Nodes A, B** and **C**.

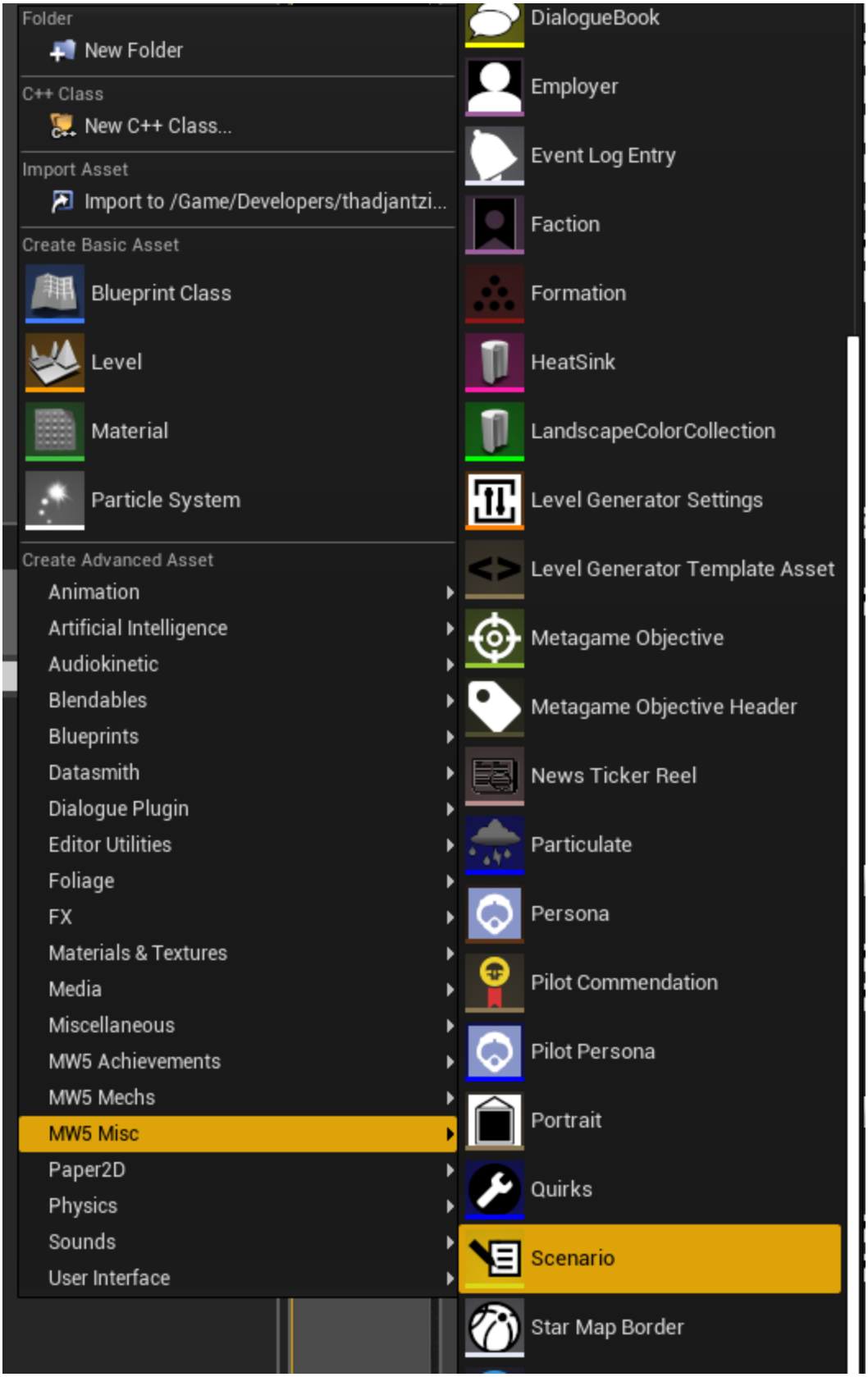
In this case set up **Node_E** the same as **Nodes A, B** and **C** but check the "**Ignore Other Links**" checkbox. This will make the result of **Node_E**'s completion happen no matter what.

Step 9: Playing the Mission

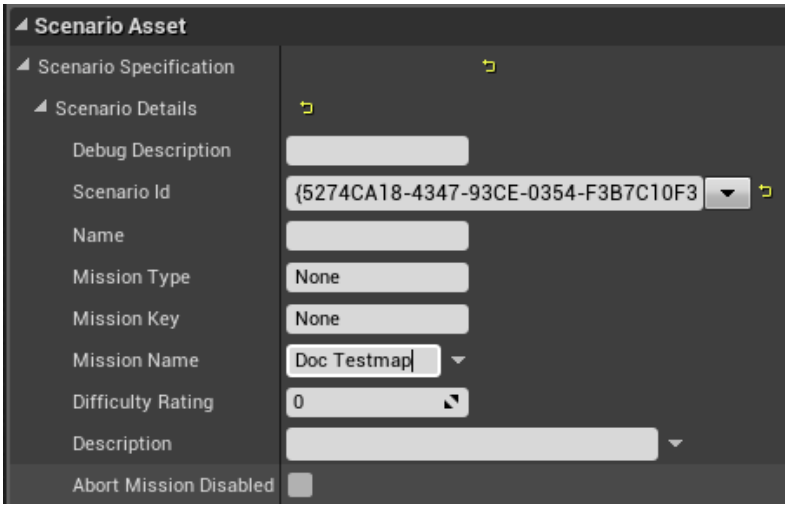
To test the mission, simply click the large "**Play**" button at the top of the **AreaSpec** editor screen.



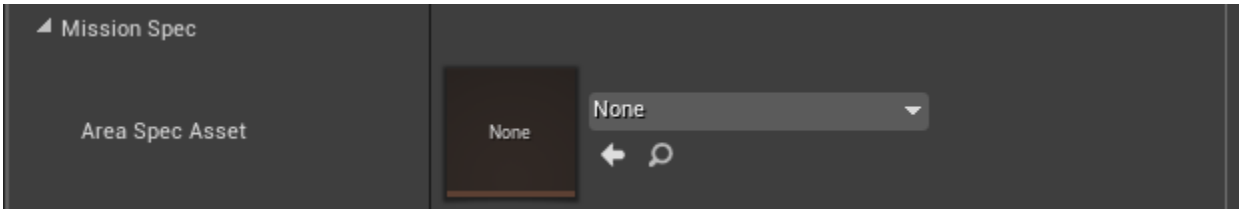
To make your mission available in the **Instant Action** menu in game, you need to create a **Scenario**. Go to the Content Browser, right-click, **MW5 Misc Scenario**.



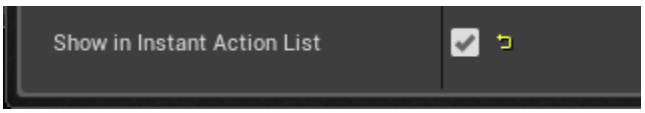
Now give it a GUID and a name.



Now scroll down a bit and set the Area Spec Asset to the Area Spec you've been working on this whole time:



And at the very bottom of the scenario is "Show in Instant Action List". Check this, and you'll be able to play your mission from the Instant Action menu.



For instructions on how to integrate custom missions into the campaign flow, consult the *Adding Custom Mission to Campaign Quick Start* document.

Other Things You Can Do

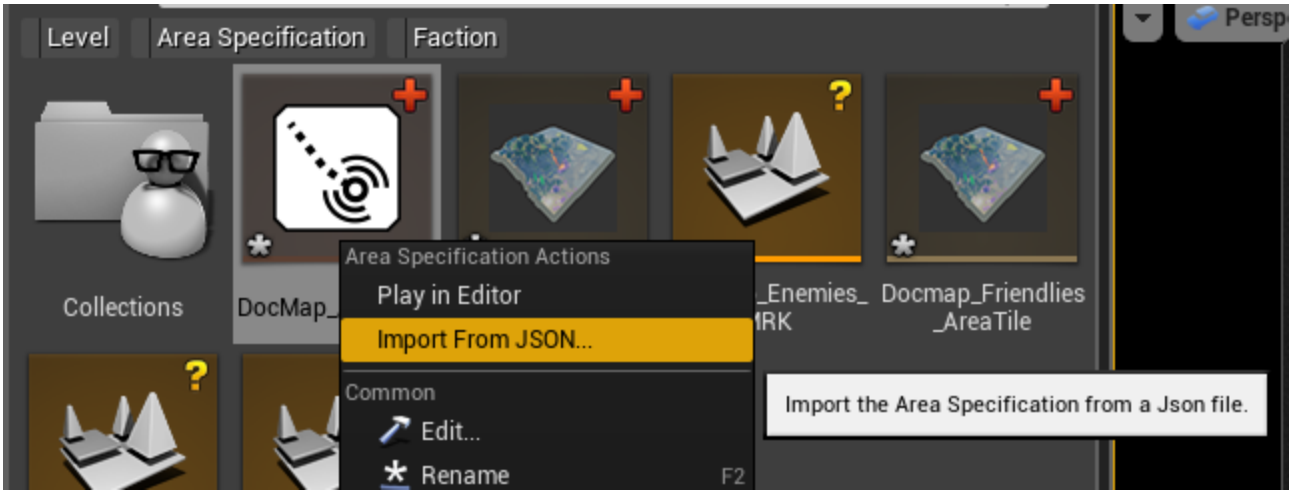
So we've described the process of putting together a single custom mission with one big custom terrain tile. There are a number of other things you can do, though:

Custom Mission Made From Existing Tiles

You could put together a level by manually adding existing tiles into your **`AreaSpec`** and enter in the coordinates/orientations manually. Then select the Configuration for each tile that most closely matches your purpose/desired locators, and add the gameplay manually the same as described above.

"Bake" out a Procedural Mission

When you run the game from the editor, it saves out a .JSON of your current mission in the **`MW5Mercs/Saved/Logs`** folder. If you are playing a proc mission that you like and want to save out, simply keep that .JSON file, make a new **`AreaSpec`** in the editor, and right-click the **`AreaSpec`** in the content browser and select Import from JSON. You'll now have an **`AreaSpec`** of that mission which you can go in and tinker with, change the biome, change the units, move tiles around, do whatever you like.



Custom Garrisons

You can certainly make custom garrisons and add them to both the main game and custom missions. It simply requires a new garrison level, and for each garrison level a **`Tile Element.`** The **`Tile Element`** simply specifies what level will get loaded, and what type of **`Locator`** it is compatible with. I recommend looking at the existing **`Tile Elements`**, and existing garrisons and working from there.

Best Practices

By now it should be clear that the process required for putting these missions together can be extremely complicated and unwieldy. There are a few steps I would recommend taking to try to make it as efficient as possible:

Naming Conventions

Anything that you can give a name to, you should give a name to, and you should do it as consistently as possible for everything you work on. This would extend to everything you place in the levels, especially the markup levels, to all the names of the **`Mission Flow Nodes`**, and above all to your file names and paths.

Document Your Missions

When all of your data is just inside arrays of **`Mission Flow Nodes`** in your **`Area Spec`** it becomes extremely difficult to find anything. I strongly recommend, at the very least, keeping a spreadsheet that lists the array indices, and names of all of your **`Mission Flow Nodes`**. That way if you need to find one particular node in your mission, you won't need to scroll through like sixty nodes trying to find it. (If you do have to do this, the filter bar in the **`Area Spec`** editor will be your friend).

I also highly recommend using this same spreadsheet to list any AI units you are spawning, the unit type, the particular node that spawns them, and their GUIDs. This will save you a ton of time later on if you have to go in and make any adjustments.

If you were really smart you'd keep a flow chart which depicted visually all of your flow nodes, and illustrated the logical links between them as well. This will be invaluable for debugging throughout the entire process. There are plenty of good flowcharting tools out there – I just use Excel, but you'd probably be better off with something designed specifically for the purpose.

Start Small

It is very easy to let your mission logic get bloated past the point of what is manageable. I strongly recommend starting with very simple missions until you're 100% comfortable with the systems.

Work With Flexibility in Mind

Since the MW5 tools allow you to change a great deal of the content in your missions parametrically, why not leverage that as much as possible? You could run the same mission with different biomes, create new biomes, come back to the same area covered in snow, or at night, or with all the trees burnt to a crisp, or with the city now destroyed, or with completely different gameplay.