World Partition Workflow

Last Updated: 10/10/2024 (Matt Canei)

Below is how we should generally be working and creating World Partition maps. The below document contains initial configuration, workflow organizational steps, departmental instructions and other details to facilitate working with World Partition at Believer. As workflows are always subject to improvement this workflow may be updated frequently.

Original World Partition Evaluation (additional details, discoveries and training materials):

World Partition Workflow Overview Training

https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/1a7f1f91-9847-42b4-b82f-302eb7f24c6b/WPTraining.mp4

Initial Setup Training

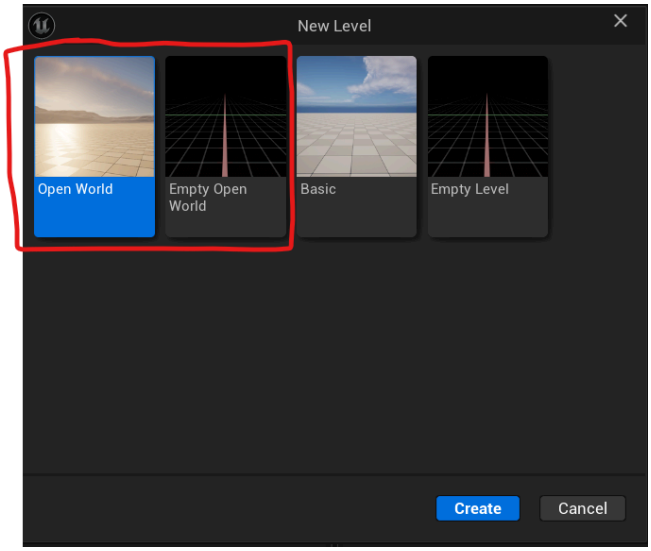
https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/5b343e88-b047-4121-aed0-29cc036b53b0/WPWorkflow_InitialSetup.mp4

Video walkthrough of all of the initial setup steps.

Initial Map Configuration

Suggested Point-of-Contact: Tech Art, Tech Design

This is the technical configuration inside the editor to setup a new World Partition map before anyone else starts working on it.



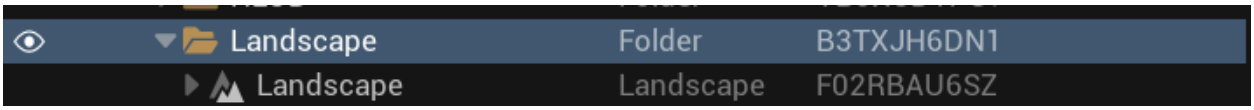
The left two options default to WorldPartition

1) **Default Map Creation**

The default Open World is 2km x 2km, if you need a different size you will need to delete the Landscape and create a new one. If 2km x 2km is fine proceed to **Step 2.**

2) **Creating Organizational Folder for Landscape**

Create a new folder for Landscape called Landscape.



3) **Assigning Game Mode**

Go to ***World Settings*** and assign a GameMode Override:

3a) If we're using Lyra you need to assign whatever our base GameMode Override is -AND- a ***Default Gameplay Experience*** (ping Engineering if you need to know what this is)

3b) If not Lyra, just select our Game Mode.

4) **Setting up our default Streaming Grid**

Under ***World Partition Setup (section)*** find ***Runtime Settings > Runtime Partitions*** and click the + icon 3 times to add 3 more entries (4 total including the Main Grid). It should look like:

4a) Expand each index and fill out the settings to match the table below. These are the default grids we will use our our Open World maps, if a setting is not listed leave the default value:

Index[0] Settings

Setting	Value
Class	RuntimePartitionLHGrid
Name	MainGrid
CellSize	25200
Loading Range	50000

Index[1] Settings

Setting	Value
Class	RuntimePartitionLHGrid
Name	Small
CellSize	12000
Loading Range	25000
	/image

Index[2] Settings

Setting	Value
Class	RuntimePartitionLHGrid
Name	Large
CellSize	40000
Loading Range	80000

Index[3] Settings

Setting	Value
Class	RuntimePartitionLHGrid
Name	Vista
CellSize	150000
Loading Range	300000

5) **Initial NavMesh Setup**

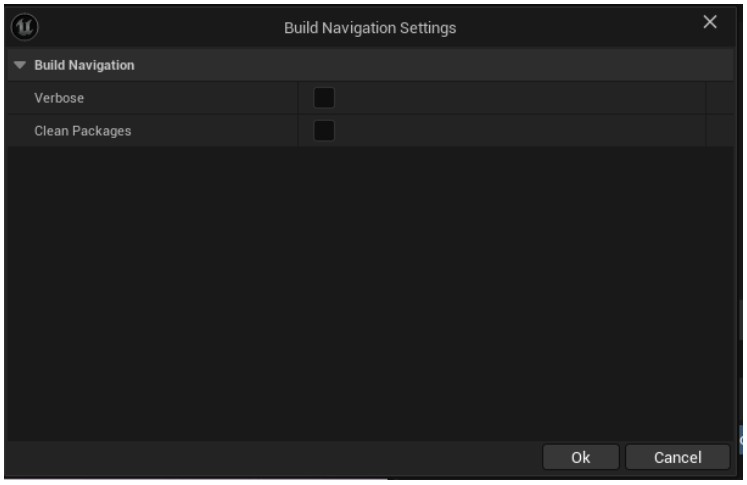
5a) Go to ***Editor Preferences > Level Editor > Miscellaneous*** and make sure ***Update Navigation Automatically*** is turned OFF.

5b) Place a ***NavMeshBoundsVolume*** in the level, set the location to 0,0,0 and set the Brush XYZ to 200000, 200000, 200000 (2k x 2k x 2k) to encompass your entire landscape.

5c) Press ~ to activate console and enter the following command in console, followed by ENTER

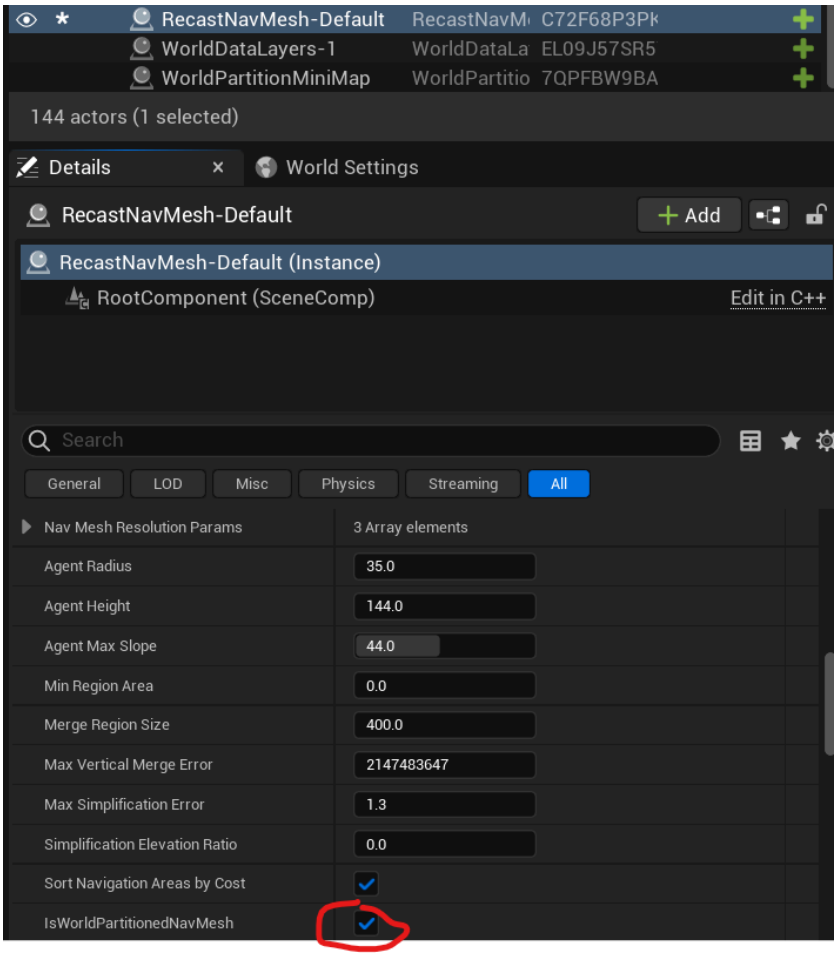
```
n.bNavmeshAllowPartitionedBuildingFromEditor 1
```

5d) At the top menu of the Editor go to ***Build > Build Paths*** and if the command worked you should get the following dialog box:



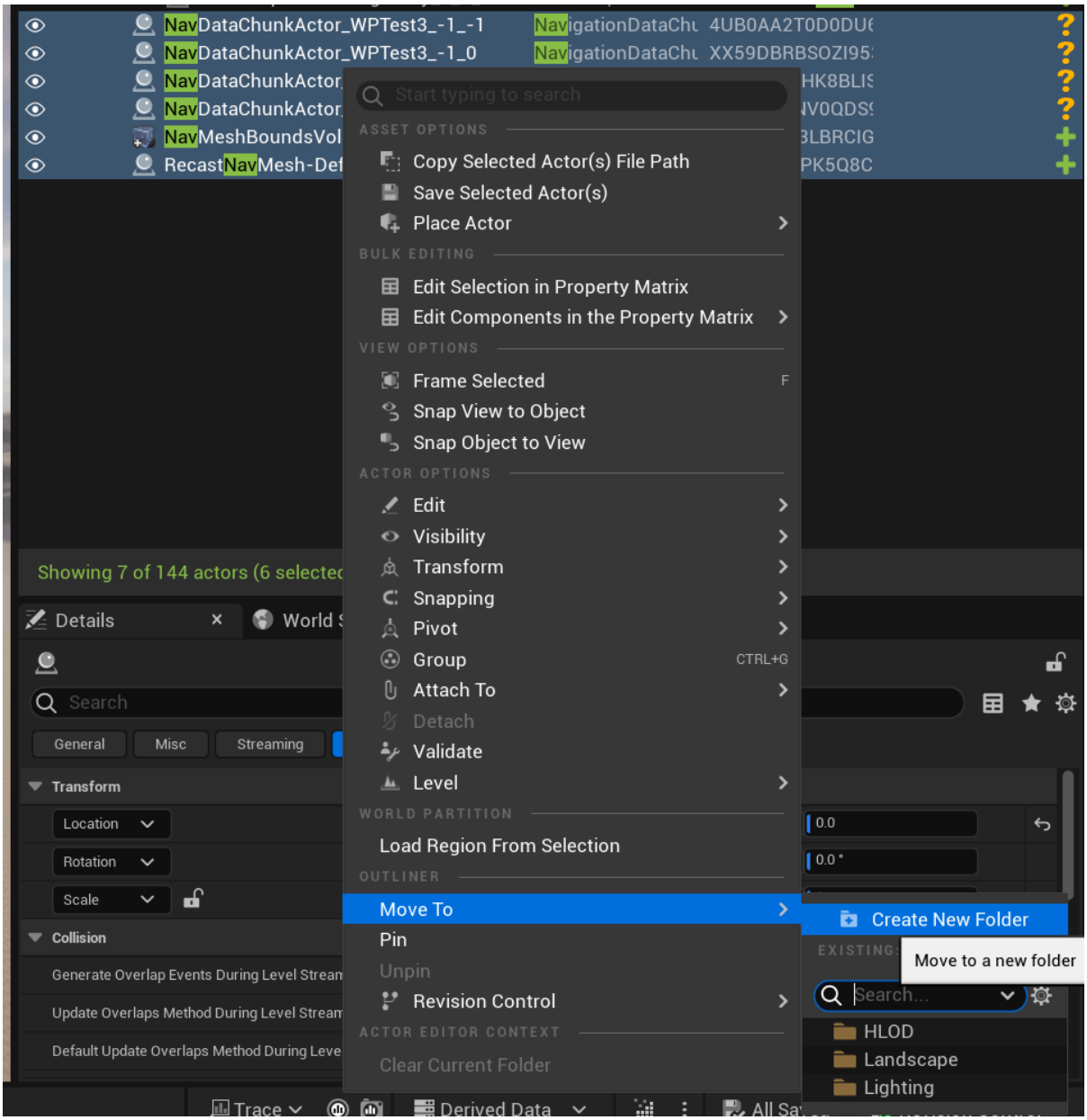
5e) Check **Clean Packages** and click OK. This process ONLY CLEANS, it does not ALSO rebuild paths. Now build paths again with Clean Packages unchecked.

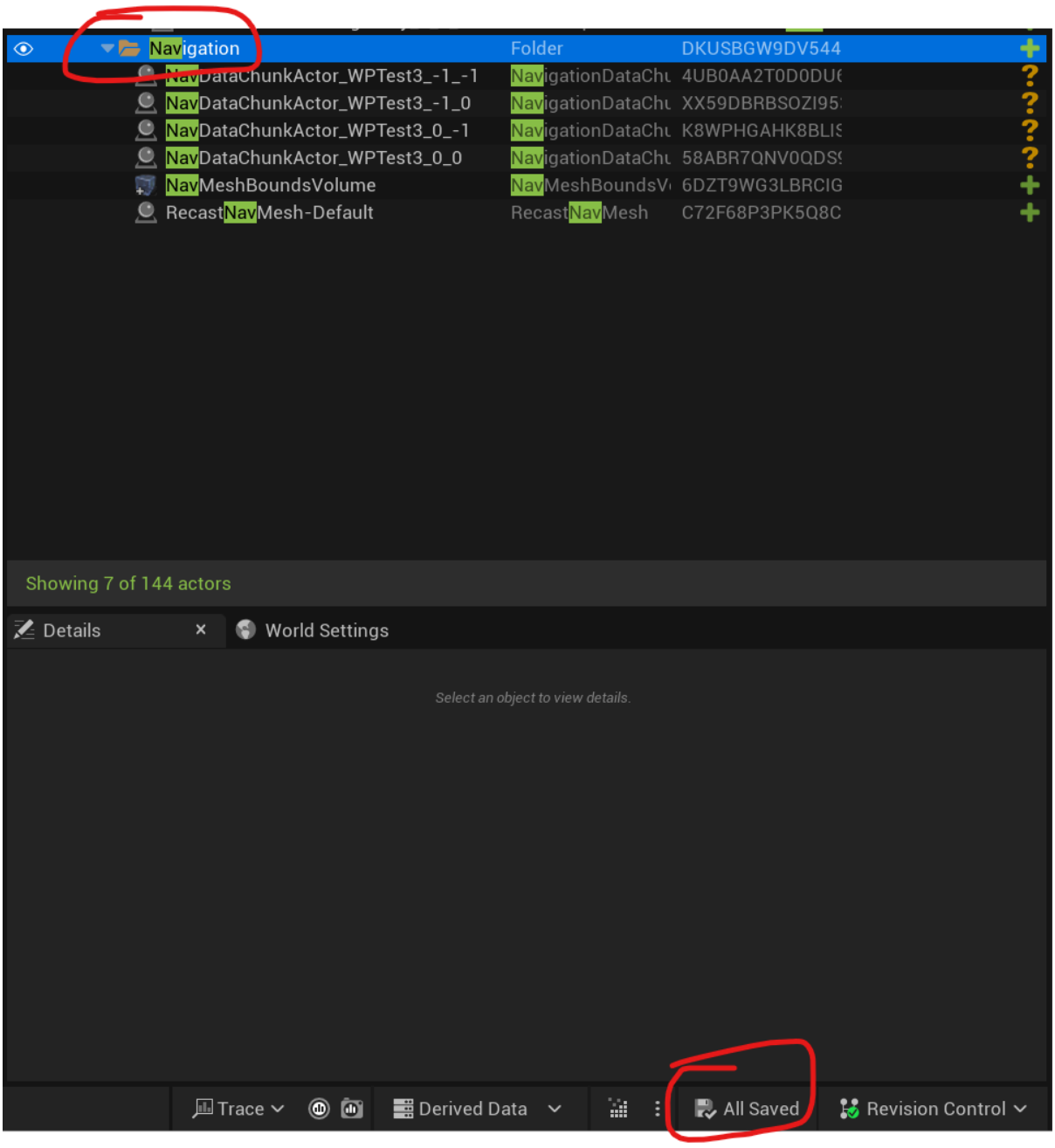
After its complete, find ***RecastNavMesh-Default*** in the ***World Outliner*** and find the setting called ***IsWorldPartitionedNavMesh*** and make sure its set to **TRUE.** If it is already set to true ignore Step 5F and go to 5g.



5f) Go to ***Build > Build Paths*** again and build with the WorldPartition setting correct (only if IsWorldPartitionedNavMesh was false after the first Build Paths).

5g) After the build is complete, in the ***World Outliner*** search for “nav” and select all actors that are navigation related. Create a folder called “Navigation” by Right Clicking on the selection > Move To > Create New Folder - after renaming the folder to “Navigation” hit ***Save*** in the bottom right:


NavDataChunkActor_WPTest3_-1_-1
NavDataChunkActor_WPTest3_-1_0
NavMeshBoundsVol
RecastNavMesh-Def
Showing 7 of 144 actors (6 selected
Details
World
Search
General
Misc
Streaming
Transform
Location
Rotation
Scale
Collision
Generate Overlap Events During Level Strean
Update Overlaps Method During Level Stream
Default Update Overlaps Method During Leve
Trace
Derived Data
Start typing to search
ASSET OPTIONS
Copy Selected Actor(s) File Path
Save Selected Actor(s)
Place Actor
BULK EDITING
Edit Selection in Property Matrix
Edit Components in the Property Matrix
VIEW OPTIONS
Frame Selected F
Snap View to Object
Snap Object to View
ACTOR OPTIONS
Edit
Visibility
Transform
Snapping
Pivot
Group CTRL+G
Attach To
Detach
Validate
Level
WORLD PARTITION
Load Region From Selection
OUTLINER
Move To
Pin
Unpin
Revision Control
ACTOR EDITOR CONTEXT
Clear Current Folder
Create New Folder
Move to a new folder
Search...
HLOD
Landscape
Lighting



6) **Initial Configuration Complete**

At this stage the map is technically configured for use. Proceed to

Initial Organization

Suggested Point-of-Contact: Tech Art, Tech Design, Design, Engineering

The following steps are our standard practice for organizing a World Partition level to best facilitate many developers working in it at the same time. Having an organized structure is key to being able to navigate through the level and find things.

The goal is to have tooling aid/replace doing this manually.

Organizational Editor Tools Usage Matrix

“When to Use” Unreal’s World Partition Organizational Tools:

Outliner Folder	**Data Layer**	**Location Volume**	**Level Instance**	**Packed Level Actor**
- Default organizational tool. - Static content that never changes.	- Used when content may change at runtime. - Used for variations that may swap/randomize.	- Used to load/unload areas in the editor at dev time for editor perf. - Could also be used to define areas for gameplay logic/UI	- Used as a “prefab” tool for ANY mixture of actors. - Used to allow devs to work on an area in isolation without needing the entire World.	- Used as “prefab” tool for static meshes only (ENV use mostly).
Examples: **-** A bunch of props. - Foliage. - A set of spline walls. - Think of these as how we used to work in sublevels (mostly).	**Examples:** **-** Temp pieces or blockout pieces teams need to see but eventually hide/unload at runtime. **-** An enemy camp that turns friendly after defeating it (Two data layers for the two states that can change	**Examples:** **-** LocationVolume covering a large forest area, so only devs working in that area have to have it loaded in editor.	**Examples:** - Large cave system that goes under the landscape that is entirely self-contained while inside. Could also contain randomization logic to do runtime procedural generation. - Large city/dungeon area that needs	**Examples:** **-** Env art or design makes a reusable building out of a modular kit. A PackedLevelActor allows that configuration to be reused. -

	loading behavior with gameplay logic). - A set of environment pieces that swap out after returning to an area.		iteration without needing the open world landscape.	

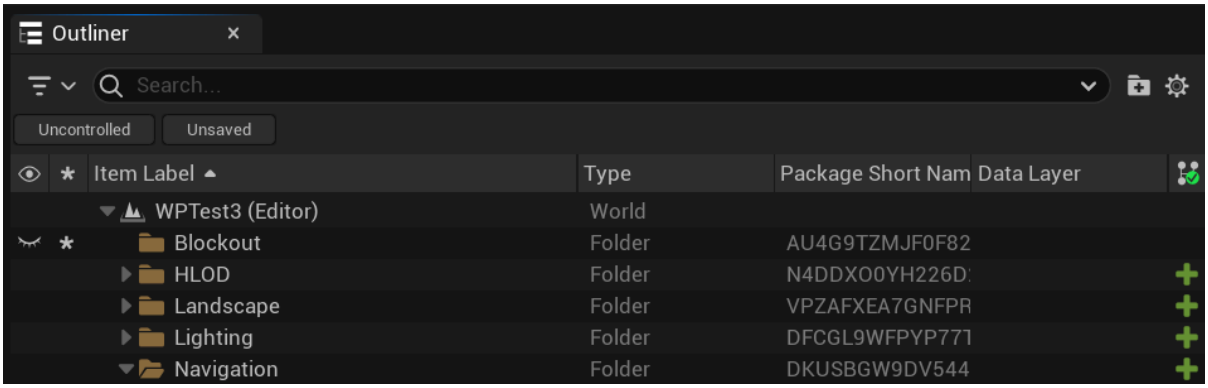
Initial Folders

This setup will cover our bases for most of our worlds and provide a framework for creating folders for specific cases.

1) **Create the following Top Level Folders** in the World Outliner and save them

Note: In World Partition Folders are saved as separate .uassets. Some top level folders may already exist if using default Open World map.

- Blockout (Global, non-area specific blockout)
- HLOD
- Landscape
- Lighting
- Navigation



After top level folders are created, we will follow an ***Area-Based*** standard where similar sub-folders are created by-department and by type for each area.

Each area in our worlds should contain the same departmental AND type folders (ie. DarkForestA below & DungeonB should have Design/AI and so on...)

Folder Standards (using DarkForestA as an area example)

- DarkForestA (Top Level Folder)
  - Design
    - Blockout
    - AI (NPCs)
    - Traversal (rails, etc.)
    - Narrative (story BPs, etc.)
    - Quest (quest BP)
    - Activities
    - Combat (enemy AI, encounter config)
    - Player (spawn points, bases/ships, dropzones, etc.)
  - ENV
    - Foliage
    - Buildings
    - Props
    - LookDev (lighting/post/volumetrics)
    - VFX
  - Audio
    - Ambience
    - SFX
    - Music
  - Cinematics
    - LevelSeq
  - Dev
    - Perf (perf cams, test BPs, etc.)

- Tests (testing BPs not related to perf, etc.)

As more areas are added, repeat the above pattern for each (with tools later)

Initial Data Layers

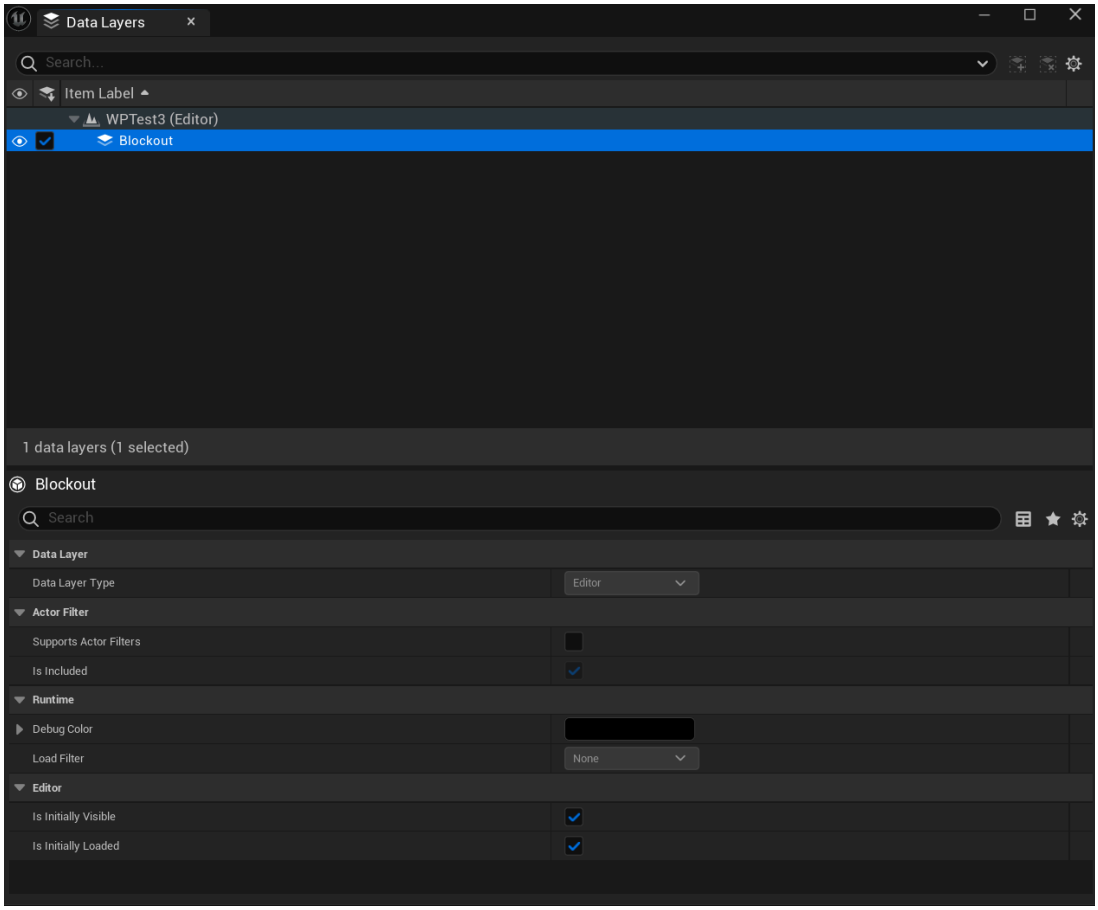
This setup is a bit less obvious as folders because DataLayers can be either Editor OR runtime, but editor ones are more-or-less redundant with folders other than additional functionality of color-coded visualization and unloading/hiding all contents as default flags when testing.

Important Note: Data Layers and Folders can be used at the same time and you can "Make Current" concurrently to organize into a folder and/or data layer by choice.

"When to Use" Private (editor) vs. Data Layer Assets (runtime)

Private Data Layer (editor only)	Data Layer w/ Asset (runtime)
Uses - When you need to organize content that is temporary but eventually will be shut off for everyone. - When you want to organize content that Folders don't cover for, for some reason.	**Uses** - When any world changes require the changing/swapping out of assets (without a cinematic showing the change live). - When there may be many variants of a particular area and we want to load one at random.
Examples - Blockout content that everyone needs to see until art passes begin, and then the Is Initially Loaded/Visible settings in the DataLayer can disable the Blockouts at runtime but let them remain in the editor. - A POI DataLayer within a location, when a team may want to keep track of more granular locations without adding even more folders.	**Examples** - An area that is "cleansed" in a questline when you return to the area would have TWO data layer assets for the initial state and the cleansed state. - We set dress (or generate) 3 possible enemy camps for a location and want the game to randomly pick one of the 3 data layers to load each time you deploy.

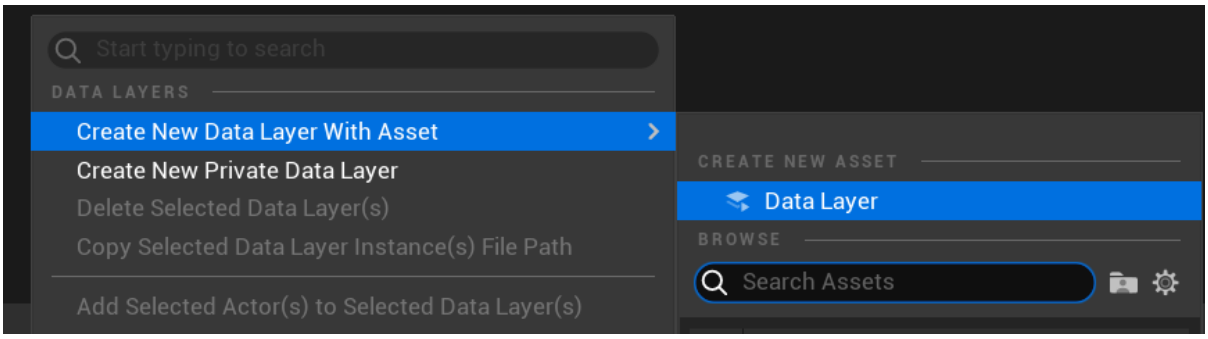
There are a few global Data Layers that are known for just about any world:

- Blockout (meant to totally disable ALL blockout meshes while testing)
  - 

- TBD

If we know from Design that there are areas that will change states based on gameplay, you can setup a Runtime Data Layer for each one of those states following the steps below. We will use the DarkForestA example from above, and assume that the forest can be cleansed, which has visual world changes we want to load in:

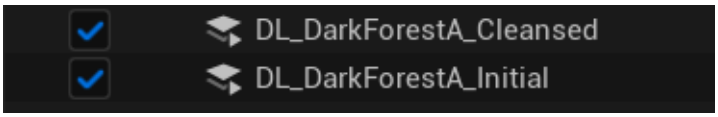
1) In the Data Layers panel, right click on open space and select ***Create New Data Layer with Asset > Data Asset***



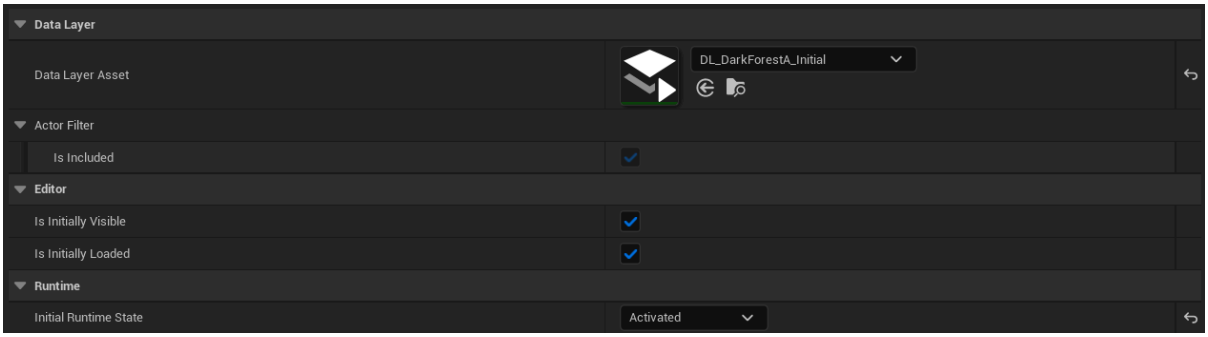
2) Navigate to the same folder as the world (or the folder for Data Layers) and save it as DL_[areaName]_[stateName] ie. DL_DarkForestA_Initial

3) Repeat steps 1 and 2 but make the state name ***Cleansed***.

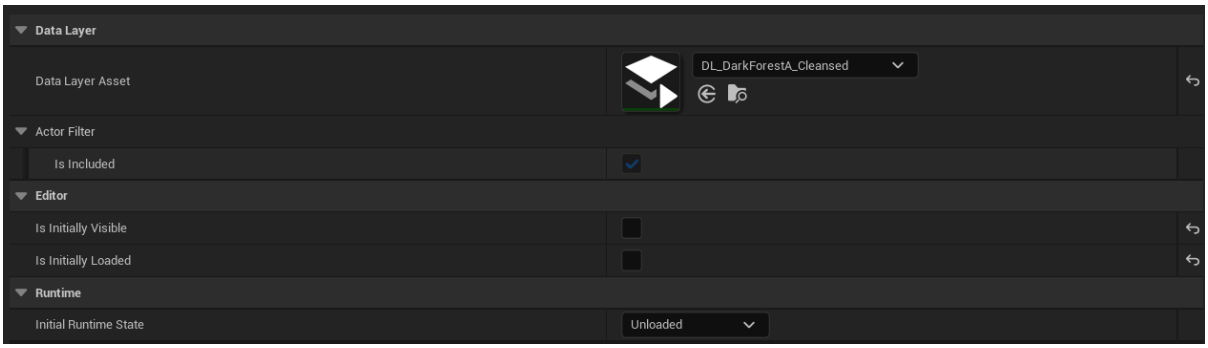
4) Review your data layer panel for the new Data Layers:



5) Since DL_DarkForestA_Initial is the starting state, make sure the settings reflect that:

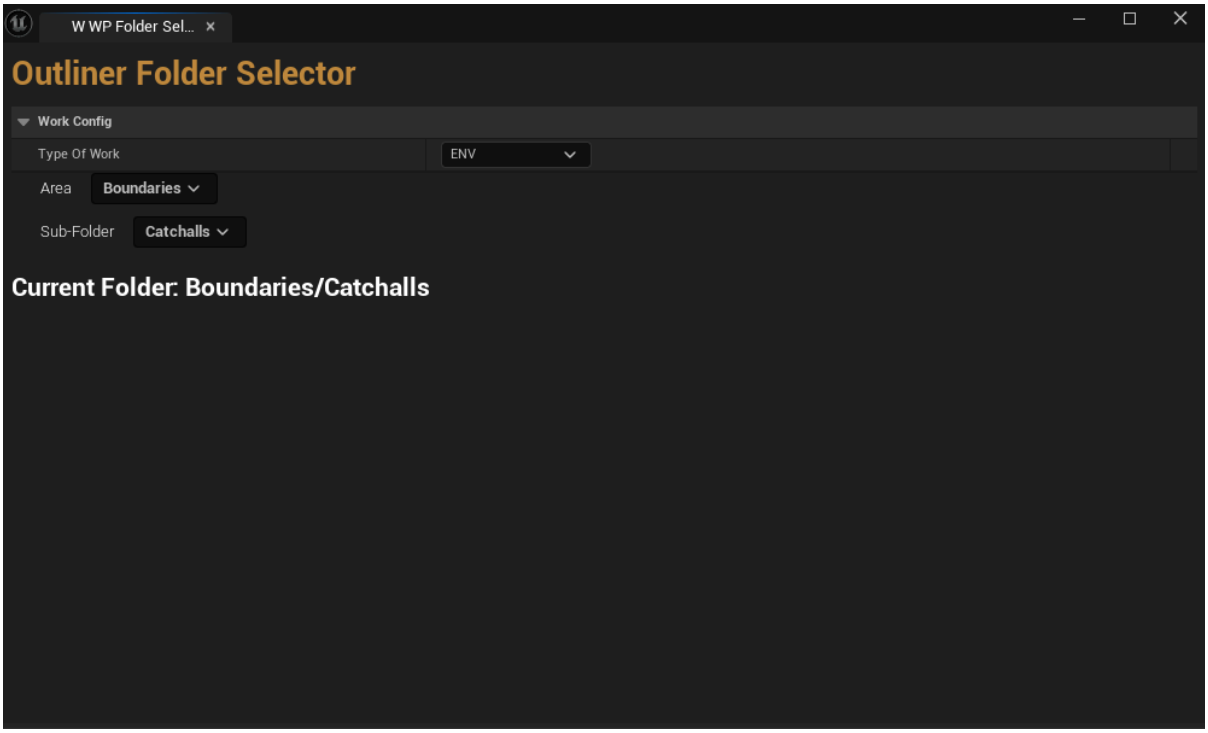


6) Review DL_DarkForestA_Cleansed and notice because this is a gameplay state, the loading/visibility of this data layer must be controlled by gameplay logic, so it will be unloaded/invisible:



Custom Tooling

World Partition Outliner Folder Selector



This tool opens every time you open a World Partition map, by design, to encourage/remind developers to pick a folder to work in. As our scenes get more and more devs touching them, its paramount to work within a proper folder/organizational structure.

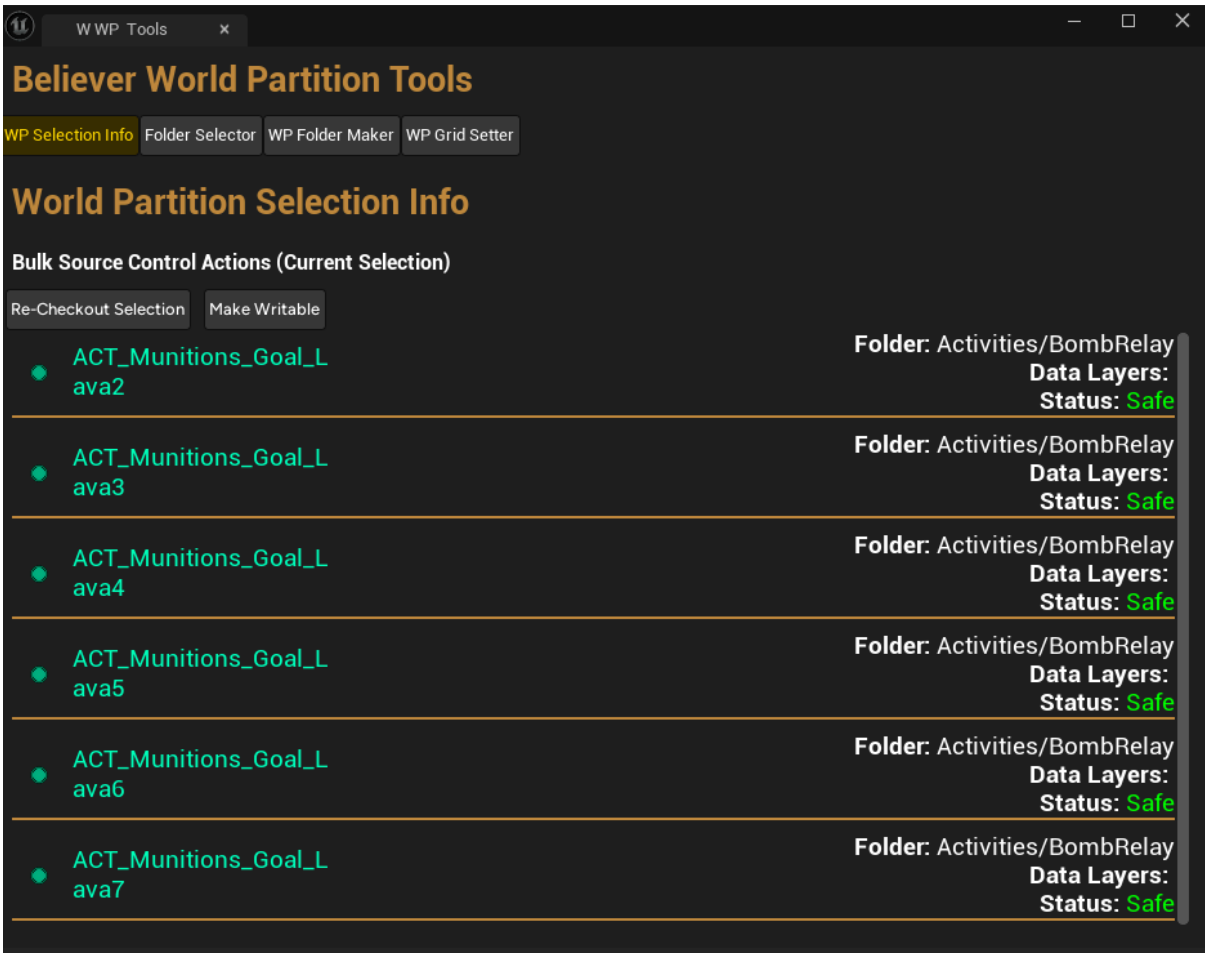
Tutorial

https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/d0abc403-4c1b-45a2-9dab-53caa45004aa/wp_folderselector.mp4

Believer World Partition Tools (Suite)

https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/06058024-0167-43fa-b9e3-daf338283b71/wptools_intro.mp4

WP Selection Info



Selection Info is designed to provide easy to read basic information about your current viewport selection in a more clear way than the default Outliner. The selection info tool will show you the following info about your selection:

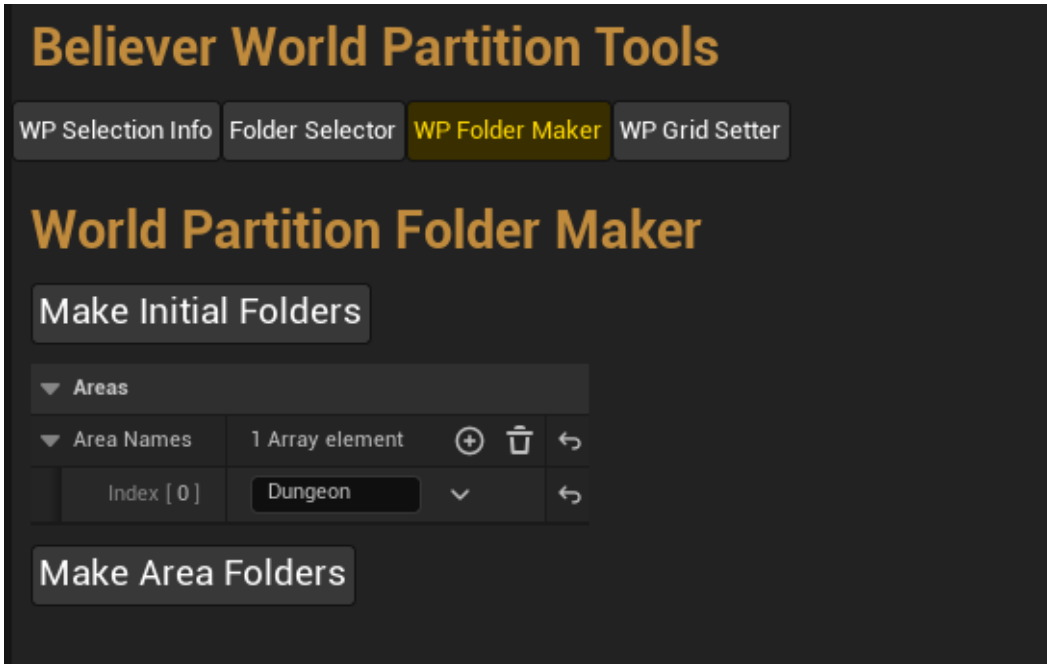
- Is that actor safe to work on? If not, why or who do you need to ask for a file?
- What folder and data layer(s) your selection is part of?

The tool also has two emergency buttons for re-checking out (locking) and making outliner selection writable if needed.

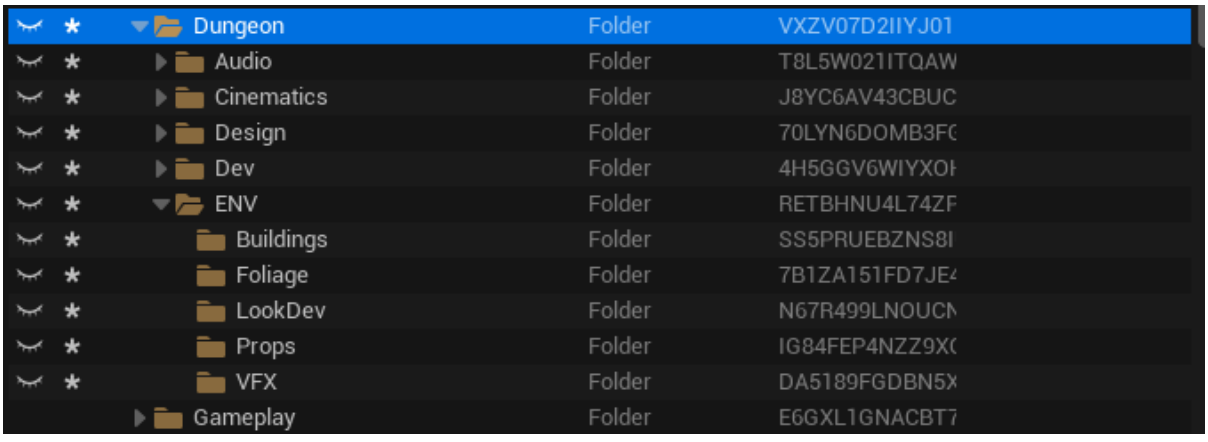
Tutorial

https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/9ffc2ff1-449f-4541-ada5-f9fb509efa79/wptools_selectioninfo.mp4

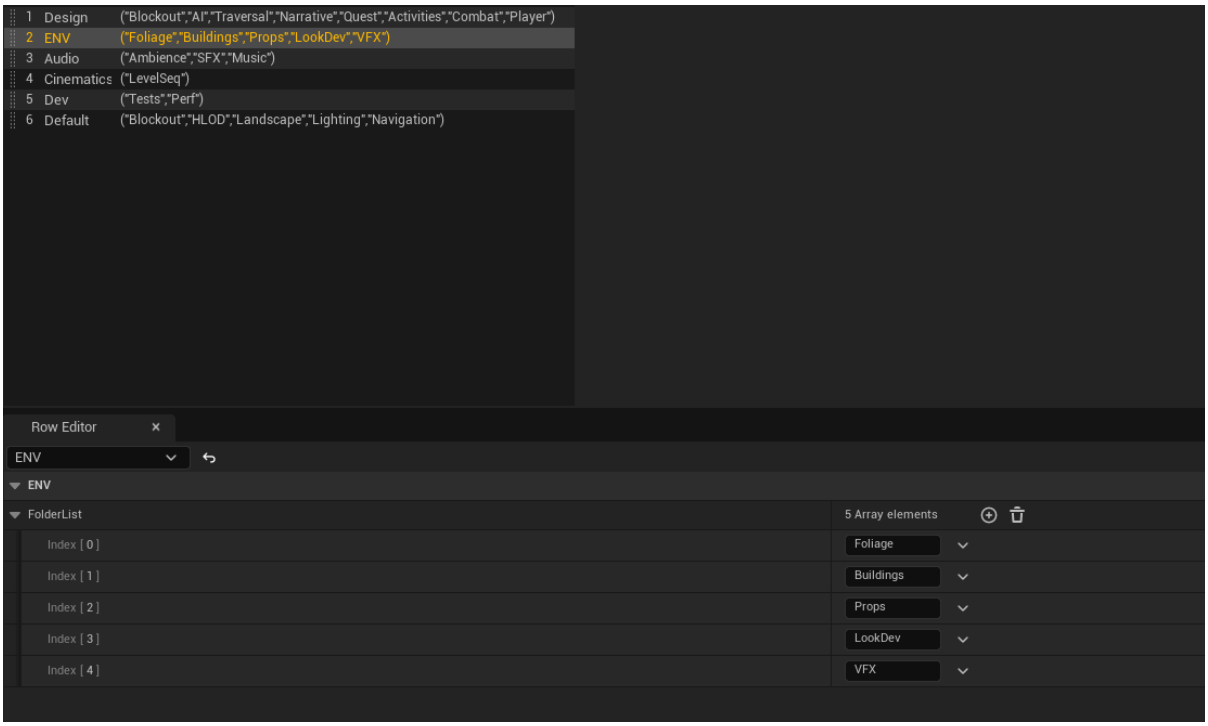
WP Folder Maker



Folder Maker allows devs to quickly generate all of the necessary folders for a level, or to make new department folders for a new area/POI. This tool automatically follows the most current folder structure (shown above in the workflow doc) so you can make the entire folder structure in a single click:



The folder structure is maintained in a data table called at: fellowship://content/Game/Editor/Tools/WorldPartition/Data/DT_WPAreaFolders

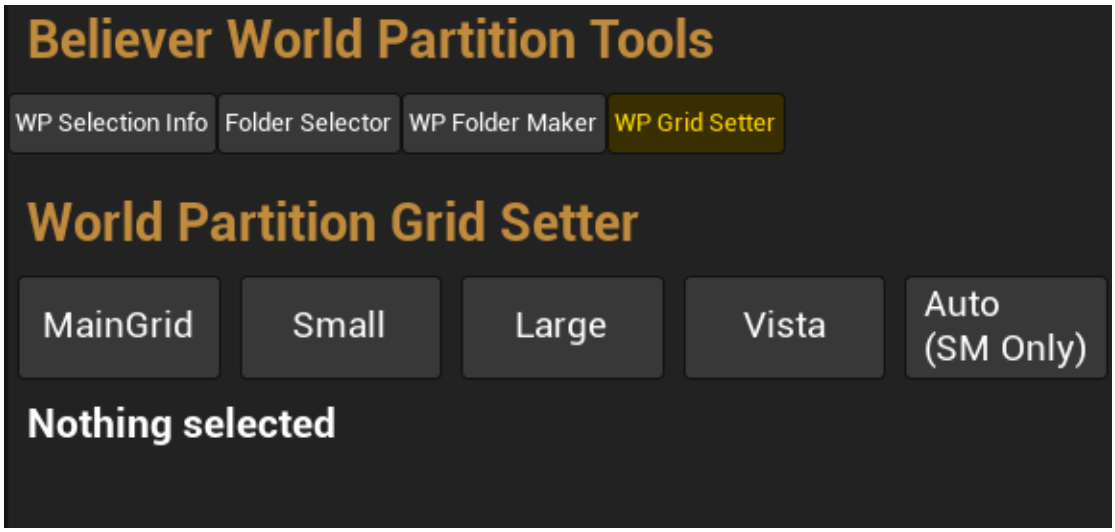


These folders will be created for every area.

Tutorial

https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/a08a0fde-30ea-43f5-acc9-7c902aa77bb3/wptools_foldermaker.mp4

WP Grid Setter



Grid Setter is a tool to apply World Partition Grids in bulk using your current viewport selection. By default, world partition grids are not assigned to things like static meshes, etc. It is SUPER important that everything in the level is assigned to a world partition grid that matches the load/visibility distance intended for that object. For example, the giant crux dish in the background needs to be in the Vista layer so its always loaded and visible, where as a scroll on a table can be in the small grid so it only loads when you're fairly close.

Tutorial

https://prod-files-secure.s3.us-west-2.amazonaws.com/792ae76d-8ed4-4636-9eae-59a35de85eb4/d776881e-4c9c-4b2d-b5af-b106afd6c264/wptools_grid.mp4

Best Practices and Tips

- Do NOT use Level Blueprint in World Partition maps. Build that behavior into contained BP classes.
- Always remember to set a current folder and/or datalayer that you are working in. (This will be aided by tooling later). **You can work in a Folder AND DataLayer at the same time.**

- When trying to decide between folder/datalayer, locationvolumes, world partition minimap and other organizational features refer to
- When doing world building, be aware of what Runtime Grid your assets need to be part of (ie. Foliage in the Small grid). **Future Tooling will help with this.**
- When moving objects around folders/datalayers only the objects themselves should need saved, they store their own references NOT the folder or datalayer asset. Folders only need saved if one of the following happens: created new, moved/nested, renamed.
- When submitting OFPA files, submitting from the editor ensures the right mixture of file actions occur for it to submit correctly. Doing so directly from friendshipper may fail.

FAQ

TBD as they come in