Unturned

Release 0.1

Smartly Dressed Games

Apr 22, 2024

ABOUT

You can add command-line arguments within Steam:

1. Right-click Unturned in your Steam Library
2. Select Properties… > General
3. Find the Launch Options field
4. Type options separated by spaces. For example "-FallbackGizmos -Width=1920 -Height=1080" will enable the FallbackGizmos flag, set Width to 1920 and set Height to 1080.

CHAPTER

ONE

GAME OPTIONS

Some command-line arguments are primarily intended for use with the Unturned Dedicated Server app.

+connect: Connect to a server, in the format of `+connect <ip address>:<port>`.

-DisableCullingVolumes: Disable object culling distance overrides. Please refer to *Manual Object Culling* for more details.

-DisableLightLODs: Disable fadeout of dynamic lights. Could be useful for high-quality screenshots.

-FullscreenMode=: Window mode override.

-FallbackGizmos: Use 3D Unity line renderer component for debug visualization rather than pixel-perfect lines. Performance with these is lower than the default, so only intended for cases where the default is unimplemented.

-FarClipDistance *float*: [16.0, 2048.0] overrides the maximum draw distance in the graphics menu. By default the lowest max draw distance is 614.4 meters which is slightly higher than the network distance of 512.0 meters. Useful for players who are willing to gain performance at a significant gameplay disadvantage.

-ForceTrustClient: Disables movement validation (e.g., position difference between ticks matches speed) for vehicles. Using this is not recommended! It is easier for cheaters to fly cars with movement limits disabled. This flag should eventually be removed when(/if) vehicle movement is made server authoritative.

-FrameRateLimit= *int*: Overrides the frame rate limit specified in the display menu. Negative values disable the limit. Useful if game is running at thousands of FPS on the loading screen and overheats.

-GameSense: GameSense integration.

-Glazier= *enum* (`IMGUI`): Use the legacy IMGUI rather than the default uGUI.

-h *int*: Alias of `-height`.

-height *int*: Override in-game resolution height.

-Holiday= *enum* (`AprilFools`, `Christmas`, `Halloween`, `HW`, `PrideMonth`, `Valentines`, `XMAS`): Override the active holiday.

-HostPlayerLimit= *int*: Clamps max number of players to this number. Useful for hosting providers.

-LegacyConsole: Use the legacy console rather than the default threaded console.

-LogAssemblyResolve: Log when the resolution of an assembly fails. Useful when working with non-Rocket plugins.

-LogBadMessages: Log when the game ignores a network message, including from whom. This is only recommended if trying to narrow down whether a connection is trying to waste time on the game thread by potentially exploiting something. By default the server automatically disconnects clients that are sending invalid messages, whereas the instances logged by this option could potentially be false positives.

-LogLevelBatchingTextureAtlasExclusions: Please refer to *Level Batching* for more details.

-LogSpawnTablesAfterLoadingLevel: Log all spawn chances.

-NetTransport= *enum* (`SteamNetworking`, `SteamNetworkingSockets`): SteamNetworkingSockets was used to enable the ISteamNetworkingSockets networking API, but this has since become default. SteamNetworking can be used to revert to the older, deprecated ISteamNetworking networking API.

-NoDefaultLog: Disables log file creation unless a plugin calls setLogFilePath.

-NoDeferAssets: Disable the deferring of loading vehicles and level objects until map load time, and instead load on startup.

-NoSteamTextFiltering: Disable Steam text filter, and instead revert to the old naïve filter.

-NoWorkshopSubscriptions: Disable loading of all Steam Workshop subscriptions. This can be helpful when troubleshooting issues.

-OfflineOnly: Disables requests to the internet. For LAN servers, it skips the Steam backend connection and uses locally-cached Workshop items.

-PreviewLevelBatchingTextureAtlas: Please refer to *Level Batching* for more details.

-RazerChroma: Enable Razer Chroma integration on compatible devices.

-RefreshRate=: Monitor refresh rate override.

-SkipAssets: Disable loading asset bundles and Workshop content. This is useful for quickly iterating on serverside code.

-TimeOverlay: Show seconds since startup under FPS in the upper-left corner.

-ui_scale: UI scale override. A common usage is to set UI scale back to its default scaling, with `-ui_scale 1`.

-UseLevelBatching *bool*: Overrides whether level batching can be enabled. Per-level support for level batching is still required. For example `-UseLevelBatching=false` disables it. Please refer to *Level Batching* for more details.

-ValidateAssets: Perform *additional health checks* on assets during start-up.

-ValidateLevelBatchingUVs: Please refer to *Level Batching* for more details.

-w *int*: Alias of `-width`.

-width *int*: Override in-game resolution width.

CHAPTER

TWO

UNITY OPTIONS

Unity's built-in command-line arguments take priority over *Unturned*'s equivalents. Some of the more relevant Unity arguments are mentioned below, but the rest can be found in the Unity User Manual.

-batchmode: Run in batch mode.

-force-glcore: Force OpenGL.

-force-vulkan: Force Vulkan.

-nographics: Do not initialize the graphics device when running in batch mode.

CHAPTER

THREE

GETTING STARTED

Installing the Unity Editor is required for exporting custom content into the game. Most 2021.3 LTS version should be compatible; Unturned currently uses 2021.3.29f1. View Download Links

Once Unity is installed, a project can be created to house custom content. At this point, it is recommended to import Unturned's provided Unity packages.

3.1 Unity Packages

Unturned provides multiple Unity packages with the base installation of the game. These packages include examples that can be referenced when creating custom content, and provide the tools necessary to export content from Unity.

These Unity packages are located in the `.../Unturned/Extras/Sources` directory, and are regularly updated along-side any major updates to the game.

1. Open your Unity project.
2. Select **Assets > Import Package > Custom Package…** from the toolbar.
3. In the file browser, navigate to the `.../Unturned/Extras/Sources` directory.
4. Import the `Project.unitypackage` file; importing the `ExampleAssets.unitypackage` file is optional.

When importing a Unity package, all of the items in the package will be installed by default. You may deselect any items that you do not want to import.

3.1.1 Project.unitypackage

This package contains the bare-bones required to export custom content:

- Default Project Settings
- *Asset Bundling Tools*
- *Mod Hooks* (optional)

3.1.2 ExampleAssets.unitypackage

This package contains vanilla content examples, and several useful prefabs:

- `CoreMasterBundle` directory has an example of each type of vanilla asset.
- `Game/Sources/Animations` directory has all of the vanilla item animations.
- `Resources/Characters/Preview.prefab` is helpful for previewing clothes.

Warning: Custom content should not be placed into the CoreMasterBundle directory. Instead, create a separate directory to house your custom content.

CHAPTER

FOUR

ASSET BUNDLE CUSTOM DATA

Unity `ScriptableObject` which can optionally be created in a *Master Bundle's* root for Unturned-specific asset bundle metadata.

`Owner Workshop File Id` *uint64*: ID of a file published to the Steam Workshop. If Unturned is loading this asset bundle from a Steam workshop file but the file ID does not match then loading will be canceled. Prevents the asset bundle from being easily copied/stolen.

4.1 How to Set Owner Workshop File

1. Within the Unity project window find your master bundle's root folder. This is the same as the Asset_Prefix specified in your MasterBundle.dat file. For example Hawaii's root folder is Assets/HawaiiMasterBundle.
2. Create the `AssetBundleCustomData` object by Right Clicking > Create > Unturned > Asset Bundle Custom Data
3. Find your workshop file's ID. This is the number after `https://steamcommunity.com/sharedfiles/filedetails/?id=` in the URL of your workshop file's page.
4. Set `Owner Workshop File Id` to match your workshop file's ID.
5. (optional) Check that Unturned is finding the custom data by looking for "Loaded (your asset bundle name) custom data from (path)" in the log file, or "Tried loading (your asset bundle name) optional custom data from (path)" in the case it is not found.

CHAPTER

FIVE

ASSET BUNDLES

The game loads textures, audio, meshes, prefabs, etc. from **Unity Asset Bundles** at runtime. How these are setup and used has evolved over the years from individual *.unity3d bundles to *.content bundles to *.masterbundle files.

Master Bundles should be used for essentially all new projects.

5.1 Tool Setup

Prior to using any of these tools they must be imported into a Unity project

1. Inside Unity open the Assets > Import Package > Custom Package… wizard.
2. Find the Unturned installation directory.
3. Navigate to the Extras/Sources directory.
4. Import the Project.unitypackage.

5.2 Master Bundles

Most official files including curated maps have been transitioned to master bundles (*.masterbundle), and they will be used for the forseeable future.

5.2.1 File Setup for Master Bundles

Master bundles can be loaded from any directory the game loads *.dat files from. Unless an override is specified, the nearest master bundle in the file hierarchy is used. While loading each directory is checked for a MasterBundle.dat file signalling the presence of a master bundle. For example, refer to the core.masterbundle in the Bundles directory.

MasterBundle.dat can set the following keys:

```
// Name of asset bundle file in the same directory as MasterBundle.dat.
Asset_Bundle_Name core.masterbundle

// Path to the asset bundle within Unity.
// Unity subfolders should match 1:1 with dat subfolders.
Asset_Prefix Assets/CoreMasterBundle

// Version 3 is Unity 2018.4 LTS. Older versions have shader consolidation enabled for␣
↪backwards compatibility.
Asset_Bundle_Version 3
```

Individual asset *.dats can set the following keys:

```
// Name of master bundle to load files from.
Master_Bundle_Override core.masterbundle

// If included, look for an individual *.unity3d asset bundle instead.
Exclude_From_Master_Bundle

// Path within master bundle to load files from.
// Used by notes to share a common object prefab.
Bundle_Override_Path /Objects/Medium/Furniture/Note
```

5.2.2 Tool Usage for Master Bundles

1. Follow *Tool Setup* instructions.
2. Open the tool from the Window > Unturned > Master Bundle Tool menu.
3. Select directories of assets in the Project window.
4. In the Inspector window tag them in any asset bundle.
5. Click the checkbox next to an asset bundle's name in the tool to mark it as a master bundle. This filters the list of asset bundles to show, and tracks an export path associated with it.
6. Click the … to choose a destination for the bundle file.
7. Click Export.
8. (optional) When redistributing the asset bundle the "multiplatform" toggle should be enabled. This ensures the appropriate shaders for each platform are included, and exports a ".hash" file so the server can validate client asset bundle integrity.

5.2.3 Motivations for Master Bundles

When upgrading to Unity 2017.4 LTS it became apparent that all asset bundles would have to be re-exported from Unity due to shader compatibility changes. This would be an incredible amount of files, so it was time to re-approach the *.content issue in a way that could quickly convert all existing content. This was handled by keeping the file hierarchy 1:1 and guessing the file extension for the by-name loading.

5.3 Individual Asset Bundles

Most official files have transitioned to the master bundle system, but some use individual asset bundles (*.unity3d). For example, the per-map road textures.

5.3.1 Tool Usage for Asset Bundles

1. Follow *Tool Setup* instructions.
2. Open the tool from the Window > Unturned > Bundle Tool menu.
3. Select individual assets or directories of assets in the Project window.
4. Click Grab to preview which assets will be exported.
5. Click Bundle to choose a destination for the asset bundle file.

5.3.2 Motivations for Asset Bundles

When beginning development of 3.0, it was key to support runtime loading of custom modded content. At the time files in asset bundles were loaded by name without extension, so each game type looked for specific names like "Item", "Object", "Animal", etc. The .unity3d extension was chosen for web browser compatibility. Obviously this system did not age well.

5.4 Content Bundles (*.content)

Deprecated since version 3.22.4.0.

This format was historically used by terrain, material palettes, and radio songs. After the April 23, 2021 patch (version 3.21.15.0) these assets can all use master bundles instead. As of the February 25, 2022 patch (version 3.22.4.0) any remaining support for content bundles has been removed. New references should use a master bundle name and relative path for the "Name" and "Path" properties.

5.4.1 Reusing Content Bundles

Although it is preferable to properly migrate older assets into master bundles, preexisting content bundles can be easily reused as a master bundle. Rename the *.content file to be *.masterbundle file instead. Then, add a corresponding MasterBundle.dat file as described in the file setup for master bundles.

CHAPTER

SIX

ASSET DEFINITIONS

Unturned **asset definitions** associate game data with Unity asset bundles. They are stored in `.dat` or `.asset` files.

For information about the file format please refer to *Data File Format*.

6.1 Header

Each asset has a common `GUID` and `Type` header:

GUID *GUID*: Unique ID used to link assets together. If left empty the game will prepend a newly generated GUID during startup.

Type *string*: Specific guides will list individual type names. This determines which keys the game will read. It can also be set to the fully qualified name of any class in any module.

Note: `Type` and `GUID` can either be specified in the root dictionary (default), or in a `Metadata` sub-dictionary. For example this is valid as well:

```
Metadata
{
	GUID 7e4b847061b64272b42ea8869fd053c7
	Type SDG.Unturned.Asset
}
```

If `GUID` is specified in the `Metadata` sub-dictionary the game cannot (as of 2023-04-13) automatically prepend a newly generated one during startup.

6.2 Body

The body contains any class properties. Individual asset type documentation elaborates on these.

ID *uint16*: 16-bit identifier. Unfortunately this ID must be unique within each category of assets (vehicles, items, animals, etc). Objects are the exception from this legacy restriction because they have been upgraded to fully use GUIDs.

Optionally the body properties can be placed into an `Asset` sub-dictionary. For example:

```
GUID [...]
Type [...]
Asset
{
	ID [...]
	Key1 Value
	Key2 Value
}
```

Is equivalent to:

```
GUID [...]
Type [...]
ID [...]
Key1 Value
Key2 Value
```

6.3 Unity Asset Bundles

Each Unturned asset is associated with a Unity asset bundle. If there is a master bundle in the file hierarchy that takes priority, otherwise a `.unity3d` bundle with the same name as the `.dat` file is used. There are several keys available to control the asset bundle:

Asset_Bundle_Version *int*: Indicates which version of Unity this `.unity3d` bundle was built for. When Unturned upgrades Unity versions it tries to maintain backwards compatibility based on this number. 1 is Unity 5.5, 2 is 2017.4 LTS and 3 is 2018.4 LTS.

Master_Bundle_Override *string*: Name of a master bundle to use rather than the `.unity3d` bundle or master bundle found in the hierarchy.

Exclude_From_Master_Bundle *string*: If this key exists the asset will look for a `.unity3d` bundle instead of the hierarchy.

Bundle_Override_Path *string*: Path within the master bundle to load rather than this asset's file path.

6.4 Localization

Each asset looks for a localization `.dat` file in the same directory based on the current language. For example: `English.dat` or `French.dat`.

6.5 Loading Order

When scanning a folder for assets the game checks in this order:

1. Is there a `.asset` file with the same name as the folder? e.g. Eaglefire.asset in the Eaglefire folder
2. Is there a `.dat` file with the same name as the folder? e.g. Eaglefire.dat in the Eaglefire folder
3. Is there an `Asset.dat` file?
4. Otherwise, load all files with the `.asset` extension in the folder.

CHAPTER

SEVEN

DATA FILE FORMAT

This article describes the syntax of Unturned's `.dat` and `.asset` files.

Each line is a key-value pair separated by a space. The key and/or value can optionally be in quotes. For example:

```
Key1 First value
"Key2 in quotes" Second value
Key3 "Third value"
```

Will be parsed as:

```
"Key1" = "First value"
"Key2 in quotes" = "Second value"
"Key3" = "Third value"
```

The only reason to quote a value is to enable comments on the same line. Quotation marks within a quoted key/value can be escaped with a \ backslash. For example `"a \"b\" c"` is parsed with quotation marks around `b`. Keys support quotes in case a space is required, but no keys in the vanilla game use spaces.

Note: Keys are case-insensitive. i.e., `Use_Cool_Option true` and `UsE_cOoL_oPtIoN true` are identical. Keys should be unique within their dictionary.

Acceptable values for a key will depend on their data type. Most—but not all—properties will use one of the *C# built-in types*.

7.1 Objects / Dictionaries

Each series of key-value pairs is a dictionary (sometimes called an object). The top level of the file is treated as a dictionary, and child dictionaries can be added with `{ }` curly braces. Adding `{` on the line after a key opens a dictionary, and the matching `}` closes it.

In this example `object1` is a child dictionary in the root dictionary, and `object2` is a grand-child:

```
object1
{
        object2
        {
                key value
        }
}
```

7.2 Arrays / Lists

Lists (sometimes called an array) can be added with `[ ]` square brackets. Adding `[` on the line after a key opens a list, and the matching `]` closes it.

In this example `values` is a list of strings:

```
values
[
	first value
	second value
	third value
]
```

Lists can also contain dictionaries as seen in this example:

```
List_Of_Objects
[
	{
		x 1
		y 2
	}
	{
		x 3
		y 4
	}
]
```

Note: Many older asset properties predate the addition of lists. In these cases arrays/lists are typically handled by a key specifying the number of items, and then appending the index number to each element's key. For example:

```
// Total number of elements in old-style list
Elements 2

// First element has an index of 0
Element_0 A

// Second element has an index of 1
Element_1 B
```

7.3 Comments

Lines starting with `//` are comments, which means they are excluded from parsing. Comments can be useful for adding helpful, explanatory notes inside an asset. Comments can also be added to the end of a line if the value is quoted.

For example these comments are valid:

```
// a comment
key1 value1
key2 "value2" // in-line comment
```

Whereas this comment will not be excluded from the value:

```
key value // this is not treated as a comment because the value is not in quotes
```

7.4 History

Prior to the 3.23.6.0 update there were two sets of custom Unturned syntax: “v1” for `.dat` files, and “v2” for `.asset` files. Assets using v1 syntax only supported key-value pairs, whereas v2 introduced dictionaries, lists, and required keys/values to be quoted.

This is why `{` and `[` must be on a new line, as existing v1 assets may have `{` or `[` as the first character of a value.

CHAPTER

EIGHT

ASSET VALIDATION

During startup the game runs fast basic health checks on assets while loading, but there are a variety of slower tests available. These can be enabled with the `-ValidateAssets` command-line flag. Errors are logged to the Client.log file, as well as to the Asset Errors menu.

Navmesh Readable: Object navmeshes should have the CPU Readable flag enabled in Unity. This is required for Recast to be able to generate the level navmesh.

Mesh Readable: Most non-navmesh meshes do not need to have the CPU Readable flag enabled in Unity. This is not currently enforced however, as the core content has lots of cases which still need cleaning up.

Missing Meshes: Mesh filters without a mesh, or mesh renderers without a mesh filter are found and logged.

Mesh Vertex Counts: Meshes with unusually high numbers of vertices are recommended to be optimized. Typically this is simply a matter of removing unused faces and vertices. Colliders have a lower recommended target because collision against complex meshes is slower than rendering complex meshes.

Missing Materials: Renderers without materials are found and logged. One exception to this is “DepthMask” renderers which are set by the game.

Material Counts: Renderers with high numbers of materials are recommended to be merged and simplified. Each individual material needs to be rendered separately (usually), so less materials is better. Good practice is to have one material for each render type on an object, i.e. a material for the opaque surfaces and a material for the transparent surfaces (if any).

Texture Readable: Most textures do not need to have the CPU Readable flag enabled in Unity. Disabling means a copy is not kept in RAM. One exception is shirt and pants textures which are layered on the CPU.

Texture NPOT: Most textures should have power-of-two dimensions, e.g. 1 x 2, 4 x 4, 64 x 32, etc. GPUs are best equipped for drawing these resolutions. Unity has import options for scaling up or down to the nearest power of two.

Audio Samples: Long audio clips with high frequencies are found and logged. Generally the clips are fairly small files.

CHAPTER

NINE

CURATED ITEMS

Community-created cosmetic items can be submitted for inclusion in the game through the curated workshop.

Accepted items are sold in the item store. The default revenue share is 25%, but for items associated with maps (e.g. the Elver Map Bundle) the revenue share is 50%.

9.1 Requirements

Some additional preparation is needed compared with regular gameplay items:

1. Following the style rules
2. Organizing your Unity project
3. Exporting a Unity package
4. Specifying Mythical effect placement

9.2 Style Rules

There are several rules for cosmetic items to help promote consistency with the Unturned's art style, seeing as they will be integrated directly into the game:

1. Please avoid super high-contrast colors. They can be painful to look at, especially when used in harsh lighting. Most of the official content has medium-intensity colors.
2. Following this, please don't go darker than #1e1e1e or brighter than #f0f0f0. Both extremes of brightness won't play nicely with lighting.
3. Flat forest/snow/desert/urban camouflages are reserved for in-game items like the ghillie suit. You are, however, allowed to incorporate patterned camouflage into your cosmetics.
4. Corners of models should not be beveled. Most objects should follow the ninety-degree edges. There isn't a limit on vertex/triangle/polygon count because anything matching the art style will naturally have a reasonable number.
5. Edges of clothing should have a slightly darker border one pixel wide. If you wear one of the official in-game items, you can see, for example, that the cuffs and waist of shirts have these.
6. Only use copyrighted content, trademarks, and intellectual property that belong to you.

More rules may be added in the future if necessary.

9.3 Unity Project Organization

Organizing your project into an export folder for your asset-bundled files (e.g. `Item.prefab`) and a sources folder for the imported files is greatly appreciated.

This makes it much easier to ensure only the necessary assets are included in the game. As an example, if a source `.blend` file were in the asset-bundled folder any unused/intermidiary meshes would accidentally get exported as well.

All of the base game's exported assets are in the `Assets/CoreMasterBundle` folder, and all of the source files are in the `Assets/Game/Sources` folder.

Cosmetic organization:

- Map-related cosmetics are in per-map folders and prefixed with the map name. For example, Arid's Arrowhead item export files are located in `Assets/CoreMasterBundle/Items/Arid/Arid_Arrowhead` and the source files are in `Assets/Game/Sources/Items/Arid`.
- Sets of items are in a per-outfit folder and prefixed with the outfit name. For example, the Cultist bundle's mask item export files are located in `Assets/CoreMasterBundle/Items/Outfits/Cultist/Cultist_Mask` and the source files are in `Assets/Game/Sources/Items/Outfits/Cultist`.
- Miscellaneous items are in the folder matching their type. For example, the Turtle Backpack exported files are located in `Assets/CoreMasterBundle/Items/Backpacks/Turtle_Backpack` and the source files are in `Assets/Game/Sources/Items/Backpacks/Turtle_Backpack`.

9.4 Exporting Unity Package

Since the assets for accepted cosmetics are included in the core asset bundle, a `.unitypackage` file is needed along with the regular `.dat` files. To export the package:

1. Select the folders containing your `Item.prefab` files. For example, if submitting the vanilla fedora you would select the `Assets/CoreMasterBundle/Items/Hats/Fedora` folder.
2. Right-click in the **Project** window
3. Click **Export Package…**
4. Ensure **Include dependencies** is checked to include the files that aren't directly placed in the asset bundles (meshes, materials, textures, etc).

Note: The Unity package is in *addition* to the regular asset `.dat` and `English.dat` files. Including the `.dat` files from your setup is useful for keeping the accepted version consistent. While not strictly necessary, including a name and description in the English text file is appreciated and will probably be used.

9.5 Mythical Effect Placement

If the item will have mythical effects the Item.prefab needs an "Effect" child transform. The orientation is rather unfortunate: +Z is the mythical's up direction and +Y is the mythical's forward direction.

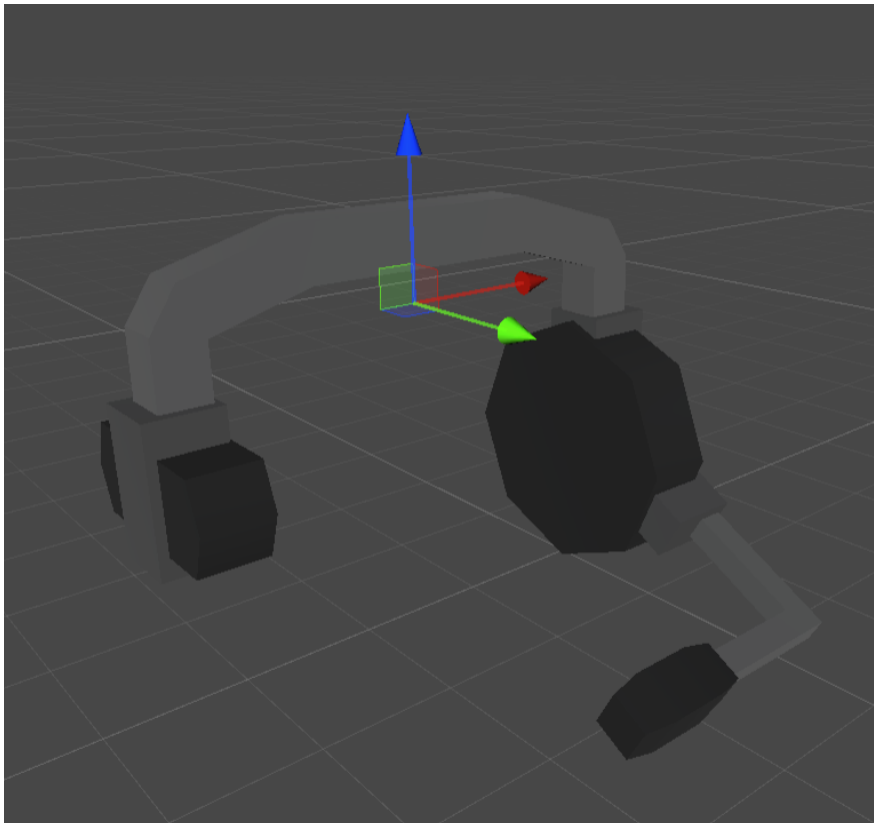


Fig. 1: Example “Effect” transform positioning and orientation.

CHAPTER

TEN

AIRDROP ASSETS

Overrides the care package model seen falling from the dropship, as well as the barricade spawned when it lands on the ground. Referenced by the *Level Asset*.

Type *string*: `SDG.Unturned.AirdropAsset`

Landed_Barricade *Asset Pointer*: Barricade item storage asset. Pivot point of the spawned barricade matches the pivot point of the care package as it hit the ground.

Carepackage_Prefab *Master Bundle Pointer*: Model to spawn falling.

CHAPTER

ELEVEN

ANIMAL ASSETS

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Animal`)

ID *uint16*: Must be a unique identifier.

11.1 Animal Properties

Health *uint16*: Total health value.

Regen *float*: How often health should be regenerated, in seconds. After the specified amount of time passes, the animal regains 1 health. Defaults to 10 seconds.

Damage *byte*: Damage dealt to the player per attack.

Behaviour *enum* (`Defense`, `Offense`, `Ignore`): AI behavior type. Defense AI will run away when alerted, Offense AI will attack when alerted, and Ignore AI will run away when attacked.

Speed_Run *float*: Running speed in m/s.

Speed_Walk *float*: Walking speed in m/s.

Horizontal_Attack_Range *float*: Maximum attack range on the horizontal plane. Defaults to 2.25 meters.

Horizontal_Vehicle_Attack_Range *float*: Maximum attack range on the horizontal plane, when the target is inside a vehicle. Defaults to 4.4 meters squared.

Vertical_Attack_Range *float*: Maximum vertical attack range. Defaults to 2 meters.

Attack_Interval *float*: Minimum seconds between attacks. Defaults to one second. If the attack duration is longer than the attack interval then the attack duration is used instead.

Roars *int*: Total number of roar sounds in Unity. A roar sound is played when the animal attacks.

Panics *int*: Total number of panic sounds in Unity. A panic sound is played when the animal is startled.

Attack_Anim_Variants *int*: Total number of attack animations in Unity. Defaults to 1.

Eat_Anim_Variants *int*: Total number of eat animations in Unity. Defaults to 1.

Glance_Anim_Variants *int*: Total number of glance animations in Unity. Defaults to 2.

Startle_Anim_Variants *int*: Total number of startle animations in Unity. Defaults to 1.

Should_Play_Anims_On_Dedicated_Server *bool*: If animations are needed on the server, such as due to having complicated triggers tied to the attack animations. Defaults to false.

11.2 Drops

Reward_ID *uint16*: ID of the item spawn table to use.

Reward_XP *uint*: Amount of experience to reward.

Reward_Min *byte*: Minimum amount of item drops to reward. Defaults to 3.

Reward_Max *byte*: Maximum amount of item drops to reward. Defaults to 4.

Meat *uint16*: ID of item to spawn when animal is killed. Deprecated in favor of Reward_ID.

Pelt *uint16*: ID of item to spawn when animal is killed. Deprecated in favor of Reward_ID.

11.3 Localization

Name *string*: Animal name in user interfaces.

CHAPTER

TWELVE

ANIMATION

Unturned's character rig is terrible – so using existing animations is recommended for your sanity.

12.1 Export

1. Ensure scene unit system is metric with unit scale set to 1.0 and length set to meters.
2. Select the Skeleton node.
3. File > Export > FBX
4. Selected Objects: True
5. Apply Scale: FBX Units Scale
6. Add Leaf Bones: False
7. Primary Bone Axis: +X
8. Secondary Bone Axis: -Y

Note that the Item.prefab from Unity is attached to the left or right hook with a local rotation of (0, 0, 90).

CHAPTER

THIRTEEN

CHARACTER MESH REPLACEMENT

The player's character mesh can be entirely replaced with a special *shirt item*. There's an example CharacterMeshReplacementTest item (ID 1522), as well as example source files in the ExampleAssets.unitypackage under the Shirts directory.

Two limitations are that it must be a shirt because only shirts are loaded for first person (1P) views, and the 1P model should only contain the arms because the rest of the body is not animated.

13.1 Properties Reference

- **Has_1P_Character_Mesh_Override**: true
- **Character_Mesh_3P_Override_LODs**: >0
- **Has_Character_Material_Override**: true
- **Hair_Visible**: true/false
- **Beard_Visible**: true/false

If `Has_1P_Character_Mesh_Override` is true then the game will try to load a prefab named "Character_Mesh_1P_Override_0". This should have a MeshFilter component with the first person arms replacement mesh.

If `Character_Mesh_3P_Override_LODs` is greater than zero then the game will try to load prefabs for each LOD index (e.g., Character_Mesh_3P_Override_0). These should have MeshFilter components for the third person replacement meshes.

If `Has_Character_Material_Override` is true then the game will try to load a material named "Character_Material_Override" to replace the 1P and 3P mesh materials. Without this, equipped shirt and pants textures will be used by default.

CHAPTER

FOURTEEN

CRAFTING BLACKLIST ASSETS

Prevents specific items or blueprints from being used while crafting. They are hidden from the item quick actions menu and recipe list.

Type *string*: `SDG.Unturned.CraftingBlacklistAsset`

Input_Items array of Item *Asset Pointers*: Any blueprints consuming these items cannot be crafted. For example (blacklisted items are highlighted):

```
Input_Items
[
	// Orange Hoodie
	"GUID" "67c76cdf16024bf68b6e5d14d4c617ab"

	// Individual items can also be enclosed in brackets { }
	{
		// Eaglefire
		GUID b03d581a5c1a490f995f8deba57b0f17
	}

	// Jeans
	dab78cc4d66645bfb8169be7c15cf876
	55c69817a31448b685c7f788ec7d2d0c
	bdae9d26ca704d729b2b0f34812d2a36
	67a6ec52e4b24ffd89f75ceee0eb5179
]
```

Output_Items array of Item *Asset Pointers*: Any blueprints generating these items cannot be crafted.

Blueprints array: Prevent specific individual blueprints from being crafted. Each entry has an `Item` *Asset Pointer* and `Blueprint` index. For example, to prevent the Chef Hat from being salvaged:

```
Blueprints
[
	{
		Item a6099002318e4d58b8e59d431bcf1b8a
		Blueprint 0
	}
]
```

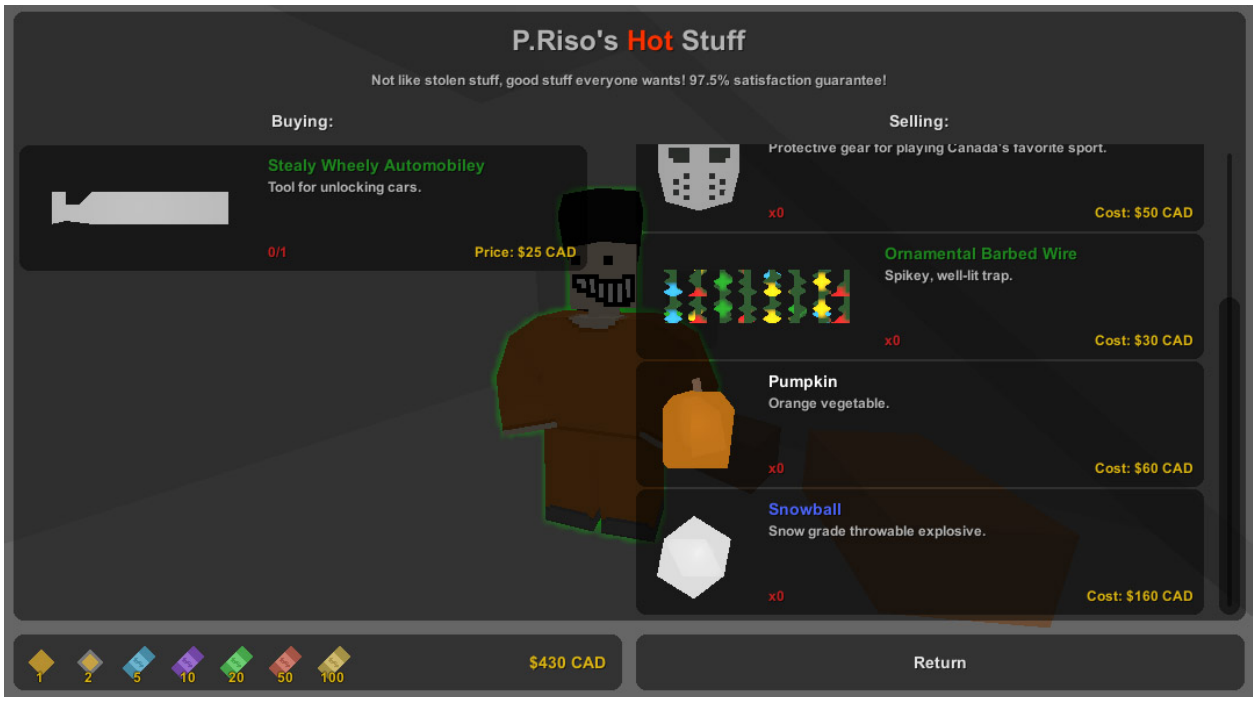
Allow_Core_Blueprints *bool*: Defaults to `true`. If `false`, blueprints from the vanilla/built-in items are not allowed.

CHAPTER

FIFTEEN

CURRENCY ASSETS

Any collection of items with different numeric values can be associated together in a **Currency** asset. NPCs can then automatically convert between the different items, and vendor menus can display information using the linked currency. This is intended to be useful beyond real-world currencies, e.g. bartering ammunition.



P.Riso's Hot Stuff

Not like stolen stuff, good stuff everyone wants! 97.5% satisfaction guarantee!

Buying:

Stealy Wheely Automobiley
Tool for unlocking cars.
0/1 Price: $25 CAD

Selling:

Protective gear for playing Canada's favorite sport.
x0 Cost: $50 CAD

Ornamental Barbed Wire
Spikey, well-lit trap.
x0 Cost: $30 CAD

Pumpkin
Orange vegetable.
x0 Cost: $60 CAD

Snowball
Snow grade throwable explosive.
x0 Cost: $160 CAD

$430 CAD

Fig. 1: P.Riso's Hot Stuff vendor.

15.1 Asset Setup

The currency asset defines how numbers are formatted, which items make up the currency, and their individual values. An example can be found at Bundles/Items/Supplies/CanadianCurrency.asset.

Type *string*: `SDG.Unturned.CurrencyAsset`

ValueFormat *string*: String to format numeric value into. For example "${0:N0} CAD" is the vanilla Canadian currency format.

DefaultConditionFormat *string*: If an NPC currency condition does not specify a formatting string this is used as the default. {0} is the total value held in the player's inventory, and {1} is the condition value. For example

"${0:N0}/{1:N0} CAD" is the vanilla Canadian currency format.

Entries: Array of items in the currency. Each has an **Item** GUID and **Value** integer. Optionally **Is_Visible_In_Vendor_Menu** bool can be false to hide the item from the NPC vendor currency list. For example these are the $10 and $20 notes in the Canadian currency:

```
{
        "Item"
        {
                "GUID" "b6b87dfad5f342dc91bbb2de950f56ee"
        }
        "Value" "10"
}
{
        "Item"
        {
                "GUID" "3b9847bb328d445495b64be9e5ea9400"
        }
        "Value" "20"
}
```

To link a vendor with a currency set the vendor asset's **Currency** to the currency asset's GUID. Vendors display all of the items sorted from lowest to highest value.

15.2 NPC Logic

Conditions can use the **Currency** type to require different total amounts in the player's inventory. Rewards can use the **Currency** type as well to grant amounts. Refer to *Conditions* documentation and *Rewards* documentation for documentation.

15.3 Testing

The built-in "give" command accepts currency GUIDs as an alternative to item IDs. For example the following grants $1,000 CAD to the local player:

```
/give 5150ca8f765d4a68bfe54912146da410/1000
```

CHAPTER

SIXTEEN

EFFECT ASSETS

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Effect`)

ID *uint16*: Must be a unique identifier.

16.1 General data

Blast *uint16* or *GUID*: ID or GUID of effect.

Lifetime *float*: Duration of the effect.

Lifetime_Spread *float*: Variation on the duration of the effect. A random value is chosen between the specified spread value, and the negative of that spread value. Default is 4 seconds.

Gore *bool*: Effect is hidden when gore is disabled.

Static *flag*: Disable randomized audio pitch change.

Randomize_Rotation *bool*: Rolls the effect around the hit axis. Defaults to true.

Spawn_On_Dedicated_Server *flag*: Spawn effect on server.

Relevant_Distance *float*: How far away players can be before an asset effect should not be sent to them, measured in meters. Players within this radius will be sent the effect in multiplayer.

Preload *byte*: Total number of the effect to pre-instantiate in the effect pool to reduce hitching when first used.

Is_Music *bool*: Placeholder to disable music when used in an ambiance volume if music option is disabled. Once the audio settings menu is separated out there will be a volume multiplier for music.

16.2 Camera shake

CameraShake_MagnitudeDegrees *float*: The amount of camera shake inflicted upon affected players, in degrees.

CameraShake_Radius *float*: Players within the radius around the effect are affected by other camera shake properties.

16.3 Splatters

Splatter *int*: Total number of splatter textures in Unity.

Splatters *int*: Total number of splatters to spawn.

Splatter_Lifetime *float*: Duration of the splatter.

Splatter_Lifetime_Spread *float*: Variation on the duration of each individual splatter after effect spawn. A random value is chosen between the specified spread value, and the negative of that spread value. Default is 1 second.

Splatter_Liquid *flag*: Splatters are visible regardless of effect graphics settings being disabled, and slightly changes the direction of each splatter.

Splatter_Temperature *enum* (`Acid`, `Burning`, `Warm`): Temperature status effect caused when standing in the effect.

Splatter_Preload *byte*: Total number of the splatter effects to pre-instantiate in the effect pool to reduce hitching when first used.

CHAPTER

SEVENTEEN

FOLIAGE ASSETS

There are sub-types of foliage asset for different uses, most notably instanced meshes (grass, pebbles) and resources (trees). Unlike the older system, tree baking cannot be configured directly within the level editor yet, but there are two benefits to separating baking settings from the trees themselves:

1. Different biomes or levels can use the same trees with different parameters. For example a dense forest material with less dense forest surrounding it, or using tree assets from a different map with custom configuration.
2. Eventually the resource system should be converted into a regular objects (this will be automatic) but most objects do not need foliage parameters.

17.1 FoliageResourceInfoAsset Properties Reference

Type *string*: `SDG.Unturned.FoliageResourceInfoAsset`

Resource *Asset Pointer*: actual tree to spawn.

Obstruction_Radius *float*: spawn position is invalid if a sphere with this radius overlaps anything.

Density *float*: this value is poorly named. One tree will try to spawn per this many square meters. For example a value of 4 will spawn approximately once per 2m x 2m area.

Min_Weight *float*: [0, 1] only spawn if landscape material weight is greater than this value.

Max_Weight *float*: [0, 1] only spawn if landscape material weight is less than this value.

Min_Angle *float*: [0, 90] degrees only spawn if surface angle is greater than this value. For example a boulder only spawning on slopes steeper than 45 degrees.

Max_Angle *float*: [0, 90] degrees only spawn if surface angle is less than this value. For example a tree not growing on slopes steeper than 30 degrees.

17.2 Upgrade Devkit Foliage from V1 to V2

Note: Maps with auto-converted terrain from the 3.22.8.0 update will have already been converted to V2.

V1 of devkit foliage saved each small, individual region into their own files, which made maps slow to copy, download, and install. V2 fixes this by storing pointers for each region into a single file, at the cost of RAM in the map editor.

Following the 3.22.20.0 update, maps using v1 foliage will automatically update to v2 the next time they are saved, without needing to use the `-SaveFoliageUsingV2` command-line argument. The older v1 foliage files are still kept in the map's Foliage directory as a backup. These v1 files can be manually removed after having converted to v2, in order to free up space.

CHAPTER

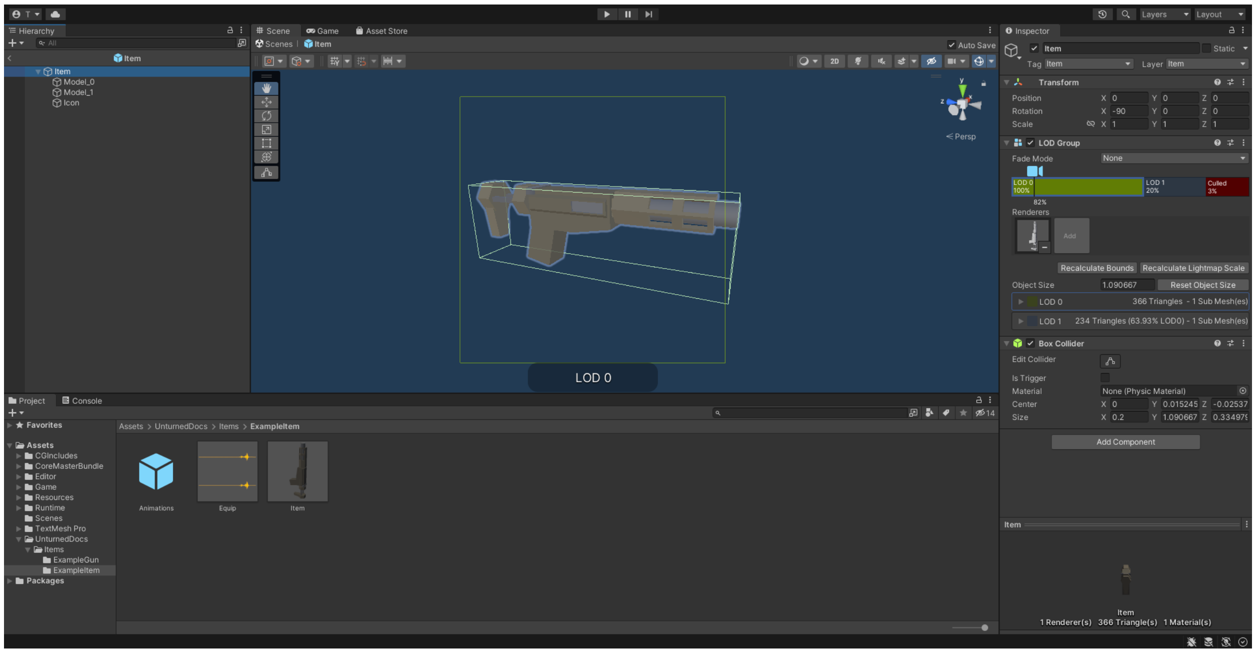
EIGHTEEN

ITEM ASSETS

18.1 Introduction to Items

Items in *Unturned* encompass anything that can be carried in a player's in-game inventory. All items will share certain properties, but each item type may have its own unique properties as well. Please refer to *Asset Definitions* and *Asset Bundles* for the full documentation regarding assets and asset bundles.

18.1.1 Unity Asset Bundle Contents



Item, Static unchecked
Tag: Item, Layer: Item

Transform
Position X 0 Y 0 Z 0
Rotation X -90 Y 0 Z 0
Scale X 1 Y 1 Z 1

LOD Group
Fade Mode: None
LOD 0 100%, LOD 1 20%, Culled 3%, 82%
Object Size: 1.090667
LOD 0: 366 Triangles - 1 Sub Mesh(es)
LOD 1: 234 Triangles (63.93% LOD0) - 1 Sub Mesh(es)

Box Collider
Is Trigger: unchecked
Material: None (Physic Material)
Center X 0 Y 0.015245 Z -0.02537
Size X 0.2 Y 1.090667 Z 0.334979

Fig. 1: An example of an item being set up in the Unity editor.

To get started, create a new folder for your custom item. The name of this folder will be relevant when further configuring your item after it has been exported from Unity.

Item (Prefab)

Inside this folder, create a new Prefab named "Item". This should be tagged as 4: Item, and layered as 13: Item. Open the "Item" Prefab.

Items can have multiple colliders including different types, but just attaching a Box Collider component to the root GameObject will usually suffice. It is recommended to use a minimum dimension of (0.2, 0.2, 0.2), because the large colliders are less likely to fall through a thin surface in a single physics tick.

If your item only has one LOD, you can attach Mesh Filter and Mesh Renderer components directly to the root GameObject. Configure these components as desired.

It is recommended to have multiple LODs for your item, so that less needs to be rendered when the item is far away. If your item should have multiple LODs, attach a LOD Group component to the root GameObject. Create a child GameObject for each LOD, named "Model_#" (e.g., "Model_0", "Model_1"). Attach the Mesh Filter and Mesh Renderer components to each one. Configure these components as desired.

Add a new child GameObject named "Icon" to the root GameObject. This will be used to draw an icon with an orthographic camera. By default, the game will automatically calculate the position and size of the camera – so the only thing that needs to be configured is its orientation. To test the orientation of your icon, temporarily attach a Camera component with its Projection property set to "Orthographic". When satisified, delete the Camera component.

Animations (Prefab)

For equippable items, a Prefab named "Animations" is required. The Prefab and the animations included can either be created from scratch, or they can be duplicated from the provided Unity packages.

If you have installed the ExampleAssets.unitypackage we provide, you can find the vanilla animations for most item types in the game. Prefabs can be found along the `CoreMasterBundle/Items` path, while the raw animation files can be found along `Game/Sources/Animations`.

To create the Prefab from scratch instead, add a new Prefab named "Animations" in your custom item's folder. Add an Animation component to the root GameObject of the "Animations" Prefab.

Every equippable item should have an animation named "Equip". If your weapon should be inspectable, it should also have an "Inspect" animation.

Equip (Audio Clip)

To have a sound play when the item is equipped, include an Audio Clip named "Equip" in your custom item's folder.

18.1.2 Game Data File

The `GUID`, `Type`, and `ID` properties are required by all item assets. Most item assets will *usually* want (or need) to include the `Rarity`, `Useable`, `Slot`, `Size_X`, and `Size_Y` properties as well, with only a few excepions.

Properties

Property Name	Type	Default Value
Allow_Manual_Drop	*bool*	`true`
Amount	*uint8*	`1`
Backward	*bool*	`false`
Bypass_Hash_Verification	*bool*	`false`
Bypass_ID_Limit	*flag*	
Can_Player_Equip	*bool*	See description
Can_Use_Underwater	*bool*	See description
Count_Max	*uint8*	`1`
Count_Min	*uint8*	`1`
Destroy_Item_Colliders	*bool*	`true`
Equipable_Movement_Speed_Multiplier	*float32*	`1`
EquipablePrefab	*Master Bundle Pointer*	
EquipAudioClip	*Master Bundle Pointer*	`Equip`
GUID	*GUID*	
ID	*uint16*	`0`
Ignore_TexRW	*flag*	
InspectAudioDef	*Master Bundle Pointer*	
Instantiated_Item_Name_Override	*string*	See description
InventoryAudio	*Master Bundle Pointer*	See description
Left_Handed_Characters_Mirror_Equipable	*bool*	`true`
Override_Show_Quality	*bool*	`false`
Pro	*flag*	
Procedurally_Animate_Inertia	*bool*	`true`
Quality_Max	*uint8*	`90`
Quality_Min	*uint8*	`10`
Rarity	*EItemRarity*	`Common`
Shared_Skin_Lookup_ID	*uint16*	See description
Should_Delete_At_Zero_Quality	*bool*	`false`
Should_Drop_On_Death	*bool*	`true`
Size_X	*uint8*	`1`
Size_Y	*uint8*	`1`
Size_Z	*float32*	`-1`
Size2_Z	*float32*	`-1`
Slot	*ESlotType*	`None`
Type	*EItemType*	
Use_Auto_Icon_Measurements	*bool*	`true`
Use_Auto_Stat_Descriptions	*bool*	`true`
Useable	*EUseableType*	`None`

EUseableType Enumeration

Named Value	Description
`None`	Does not correspond to any useable type.
`Clothing`	Corresponds to the "Clothing" useable type.
`Gun`	Corresponds to the "Gun" useable type.
`Consumeable`	Corresponds to the "Consumeable" useable type.
`Melee`	Corresponds to the "Melee" useable type.
`Fuel`	Corresponds to the "Fuel" useable type.
`Carjack`	Corresponds to the "Carjack" useable type.
`Barricade`	Corresponds to the "Barricade" useable type.
`Structure`	Corresponds to the "Structure" useable type.
`Throwable`	Corresponds to the "Throwable" useable type.
`Grower`	Corresponds to the "Grower" useable type.
`Optic`	Corresponds to the "Optic" useable type.
`Refill`	Corresponds to the "Refill" useable type.
`Fisher`	Corresponds to the "Fisher" useable type.
`Cloud`	Corresponds to the "Cloud" useable type.
`Arrest_Start`	Corresponds to the "Arrest_Start" useable type.
`Arrest_End`	Corresponds to the "Arrest_End" useable type.
`Detonator`	Corresponds to the "Detonator" useable type.
`Filter`	Corresponds to the "Filter" useable type.
`Carlockpick`	Corresponds to the "Carlockpick" useable type.

Property Descriptions

Allow_Manual_Drop bool `true`

Item can be manually dropped by the player.

Amount uint8 `1`

Maximum capacity for container-like items, such as ammunition boxes. Typically used with `Count_Min` and `Count_Max`.

Backward bool `false`

Set the item to be held in the character's non-dominant hand.

Bypass_Hash_Verification bool `false`

Disable hash verification check, and allow for mismatched files.

Bypass_ID_Limit flag

Allows for using an `ID` value within the range reserved for official content.

Can_Player_Equip bool

Item can be equipped by the player. If the `Useable` property has been set, then this defaults to `true`. Otherwise, this defaults to `false`.

Can_Use_Underwater bool

Item can be used while underwater. If the `Slot` property has *not* been set to `Primary`, then this defaults to `true`. Otherwise, this defaults to `false`.

Count_Min uint8 `1`

Minimum amount to generate, for container-like items. Typically used with `Count_Max` and `Amount`.

Count_Max uint8 `1`

Maximum amount to generate, for container-like items. Typically used with `Count_Min` and `Amount`.

Destroy_Item_Colliders bool `true`

If `false`, colliders are not destroyed when the "Item" Prefab is attached to the character. For example equipped vanilla guns do not have any colliders, but some mods (e.g., riot shields) may have relied on child colliders not being destroyed.

Equipable_Movement_Speed_Multiplier float32 `1`

Multiplies character movement speed while equipped in the player's hands. If a gun is equipped, then any gun attachment multipliers are combined as well.

EquipablePrefab Master Bundle Pointer

Overrides the model spawned when this item is equipped. For example, the "Equipable" Prefab could use an animated skinned mesh component while the regular "Item" Prefab only needs a static mesh component.

EquipAudioClip Master Bundle Pointer `Equip`

AudioClip to play when equipping.

GUID GUID

Refer to *GUID* documentation. Item assets are required to have this property.

Tip: If the GUID property has been omitted from the asset file, then the game will automatically attempt to assign a random (and unique) GUID during a successful load.

ID uint16 `0`

Must be a unique identifier. Item assets are required to have this property.

Ignore_TexRW flag

Read/writeable texture errors regarding this asset should be hidden from the error logs.

InspectAudioDef Master Bundle Pointer

AudioClip or OneShotAudioDefinition to play when item is inspected.

Instantiated_Item_Name_Override string

Name to use when instantiating "Item" Prefab. By default, the value of `ID` is used. Since Unity's built-in Animation component references GameObjects by name, this property can help share animations between items.

InventoryAudio Master Bundle Pointer

AudioClip or OneShotAudioDefinition to play when item is picked up, moved within the inventory, and dropped. Default value is dependent on the child asset.

Left_Handed_Characters_Mirror_Equipable bool `true`

If `false`, the equipped item model is mirrored to counteract the mirrored character.

Override_Show_Quality bool `false`

Override to forcefully show item quality.

Pro flag

This is a Steam Economy item.

Procedurally_Animate_Inertia bool `true`

Whether viewmodel should accumulate angular velocity from animations. Useful for low-quality older animations, but should probably be disabled for high-quality newer animations.

Quality_Max uint8 `90`

Maximum quality to generate. Typically used with `Quality_Min`.

Quality_Min uint8 `10`

Minimum quality to generate. Typically used with `Quality_Max`.

Rarity EItemRarity `Common`

Rarity of the item, as text shown in menus and colors used for highlights.

Shared_Skin_Lookup_ID uint16

Share skins with another item. Defaults to item's `ID`.

Should_Delete_At_Zero_Quality bool `false`

Item should be deleted when at 0% quality.

Should_Drop_On_Death bool `true`

Item should be dropped on death.

Size_X uint8 `1`

In slots, the total width of the inventory space (i.e., the number of columns).

Size_Y uint8 `1`

In slots, the total height of the inventory space (i.e., the number of rows).

Size_Z float32 `-1`

Manually specify orthogonal camera size for item icons. This directly corresponds to the value of a Camera component's Size property in Unity.

Size2_Z float32 `-1`

Orthogonal camera size for economy icons.

Slot ESlotType `None`

Which equipped item slot an item is valid to be equippable in. This is only relevant if your property has configured the `Useable` property.

- `None` restricts the useable item to hotkeys.
- `Primary` restricts the useable item to the primary slot.
- `Secondary` restricts the useable item to the primary or secondary slots.
- `Tertiary` is not implemented by this asset.
- `Any` has no restrictions on slots or hotkeying.

Type EItemType

Designates the item's class. Item assets are required to have this property.

Use_Auto_Icon_Measurements bool `true`

Automatically calculate axis-aligned item icon camera size from bounds.

Use_Auto_Stat_Descriptions bool `true`

If true, properties like damage, storage, health, etc. are appended to the description.

Useable EUseableType Enumeration `None`

Class for how to treat equippable items. This is often used with the `Slot` property, which determines which item slots an item is equippable in.

Blueprints and Actions

In addition to the properties already described, item assets can utilize properties for *crafting blueprints* and *context menu actions*.

18.1.3 Localization

Name *string*: Item name in user interfaces.

Description *Rich Text*: Item description in user interfaces.

18.2 Blueprints

Blueprints can be added to items. These function as "crafting recipes", which allow players to craft other items, or even modify the state of the current item. Blueprints are not restricted to affecting the item they have been added to, and a blueprint's inputs and outputs can consist entirely of unrelated items.

Context actions are able to reference blueprints. Depending on the type of blueprint added to the item, the game may automatically generate a corresponding context action as well.

18.2.1 Game Data File

The `Blueprints`, `Blueprint_#_Type`, `Blueprint_#_Supplies`, and `Blueprint_#_Supply_#_ID` properties are required by all blueprints. Blueprints also require that an output has been configured.

There are two methods available for configuring an output. When a blueprint only needs to output one item ID, the `Blueprint_#_Products` and `Blueprint_#_Product` properties can be used. Alternatively, blueprints can use the `Blueprint_#_Outputs`, `Blueprint_#_Output_#_ID`, and `Blueprint_#_Output_#_Amount` properties to output multiple, different item IDs.

It is very common that a blueprint will also use the `Blueprint_#_Build`, `Blueprint_#_Tool`, or `Blueprint_#_Skill` properties. Other properties for blueprints have more niche uses, and are less common.

Properties

Property Name	Type	Default Value
Blueprint_#_Build	*GUID* or *uint16*	
Blueprint_#_Level	*uint8*	`0`
Blueprint_#_Map	*string*	
Blueprint_#_Origin	*EItemOrigin*	`Craft`
Blueprint_#_Output_#_Amount	*uint8*	`0`
Blueprint_#_Output_#_ID	*uint16*	`0`
Blueprint_#_Output_#_Origin	*EItemOrigin*	`Craft`
Blueprint_#_Outputs	*uint8*	`0`
Blueprint_#_Product	*uint16*	See description
Blueprint_#_Products	*uint8*	`1`
Blueprint_#_Searchable	*bool*	`true`
Blueprint_#_Skill	*EBlueprintSkill*	`None`
Blueprint_#_State_Transfer	*flag*	
Blueprint_#_Supplies	*uint8*	`0`
Blueprint_#_Supply_#_Amount	*uint8*	`0`
Blueprint_#_Supply_#_Critical	*flag*	
Blueprint_#_Supply_#_ID	*uint16*	
Blueprint_#_Tool	*uint16*	`0`
Blueprint_#_Tool_Critical	*flag*	
Blueprint_#_Type	*EBlueprintType*	
Blueprints	*uint8*	`0`

EBlueprintType Enumeration

Named Value	Description
`Ammo`	Blueprint appears in the “Ammunition” tab.
`Apparel`	Blueprint appears in the “Apparel” tab.
`Barricade`	Blueprint appears in the “Barricades” tab.
`Furniture`	Blueprint appears in the “Furniture” tab.
`Gear`	Blueprint appears in the “Gear” tab.
`Repair`	Blueprint appears in the “Repair” tab.
`Structure`	Blueprint appears in the “Structures” tab.
`Supply`	Blueprint appears in the “Supplies” tab.
`Tool`	Blueprint appears in the “Tools” tab.
`Utilities`	Blueprint appears in the “Utilities” tab.

EBlueprintSkill Enumeration

Named Value	Description
`None`	No skill is required.
`Craft`	"Crafting" skill is required.
`Cook`	"Cooking" skill is required.
`Repair`	"Engineer" skill is required.

Property Descriptions

Blueprint_#_Build GUID or uint16

GUID or legacy ID of an audio effect to play upon crafting.

Blueprint_#_Level uint8 `0`

If the blueprint requires a skill, its level must be equal to this value. This property is used in conjunction with `Blueprint_#_Skill`.

Blueprint_#_Map string

Name of a map that this blueprint is restricted to. The blueprint will only be visible while on this map. For other maps, the blueprint is hidden from view.

Blueprint_#_Origin EItemOrigin `Craft`

Set the item origin. For example, setting the origin to `Admin` will cause items to spawn at full quality. This property requires `Blueprint_#_Product`.

Blueprint_#_Output_#_Amount uint8 `0`

Quantity of the product created. For example, a quantity value of `2` would create two of the item specified in `Blueprint_#_Output_#_ID`.

Blueprint_#_Output_#_ID uint16 `0`

Legacy ID of an item created as a product (i.e., an output that is provided after crafting the blueprint). This property requires `Blueprint_#_Outputs`.

Blueprint_#_Output_#_Origin EItemOrigin `Craft`

Set the item origin. For example, setting the origin to `Admin` will cause items to spawn at full quality. This property requires `Blueprint_#_Output_#_ID`.

Blueprint_#_Outputs uint8 `0`

Total number of `Blueprint_#_Output_#_ID` properties that have been configured.

Blueprint_#_Product uint16

Legacy ID of the item created as the product (i.e., an output that is provided after crafting the blueprint). To output multiple *different* items, refer to the `Blueprint_#_Outputs` and `Blueprint_#_Output_#_ID` properties instead.

When left unconfigured, this property will default to the value of the parent item's `ID` value.

Blueprint_#_Products uint8 `1`

Quantity of the product created. For example, a quantity value of `2` would create two of the item specified in `Blueprint_#_Product`. This property requires that `Blueprint_#_Product` has been set.

Blueprint_#_Searchable bool `true`

When `true`, this blueprint is visible in the search results even when the player lacks the required items. This property can be used to hide blueprints intended for debugging that are not acquirable through normal gameplay.

Blueprint_#_Skill EBlueprintSkill `None`

The player must have leveled the specified skill in order to craft this blueprint. When set to `Cook`, the player will also need to be next to a heat source (such as a lit Campfire). This property is used in conjunction with `Blueprint_#_Level`.

Blueprint_#_State_Transfer flag

Transfer the current state of any supplies to the product, when applicable. For example, some states that can be transferred include: amount (e.g., rounds in an ammunition box), quality percentage, selected firing mode, or fuel units (e.g., from a gas can).

Blueprint_#_Supplies uint8 `0`

Total number of `Blueprint_#_Supply_#_ID` properties that have been configured.

Blueprint_#_Supply_#_Amount uint8 `0`

Quantity of the supply required. For example, a quantity value of `2` would require two of the item specified in `Blueprint_#_Supply_#_ID`.

Blueprint_#_Supply_#_Critical flag

The blueprint is only visible while the player has this supply. This property requires `Blueprint_#_Supply_#_ID`.

Blueprint_#_Supply_#_ID uint16

Legacy ID of an item that is required as a supply (i.e., an input that is consumed when crafting the blueprint). This property requires `Blueprint_#_Supplies`.

Blueprint_#_Tool uint16 `0`

Legacy ID of an item that is required as a “tool” for this blueprint. This item is not consumed when the blueprint is crafted.

Blueprint_#_Tool_Critical flag

If the blueprint requires a tool, it will only be visible while the player has that tool. This property requires `Blueprint_#_Tool`.

Blueprint_#_Type EBlueprintType

This value determines which tab of the crafting menu that this blueprint appears under. All blueprints require that this has been configured.

Blueprints int `0`

Total number of blueprints. All blueprints require that this has been configured.

Conditions and Rewards

Blueprints can use quest conditions and rewards. A common usage is to make it so a blueprint is only available during a seasonal event. For more information, refer to the documentation for *Conditions* and *Rewards* respectively.

Blueprint conditions and rewards are prefixed with `Blueprint_#_`. For example, `Blueprint_0_Condition_0_Type Holiday`.

18.3 Actions

Context actions appear when a player right-clicks an item from their inventory menu. Some items may have set a of automatically-generated context actions, but additional context actions can be added to any item. The system currently supports adding additional *blueprint* actions.

Depending on an item's configuration, the game may automatically add context actions for various actions.

18.3.1 Game Data File

Properties

Property Name	Type	Default Value
Action_#_Blueprint_#_Index	*uint8*	`0`
Action_#_Blueprint_#_Link	*flag*	
Action_#_Blueprints	*uint8*	`0`
Action_#_Key	*string*	
Action_#_Source	*uint8*	`uint16`
Action_#_Text	*string*	
Action_#_Tooltip	*string*	
Action_#_Type	*EActionType*	
Actions	*uint8*	`0`

EActionType Enumeration

Named Value	Description
`Blueprint`	Action is linked a crafting blueprint.

Property Descriptions

Action_#_Blueprint_#_Index uint8 `0`

Index of the blueprint that action should perform.

Action_#_Blueprint_#_Link flag

Action should redirect to the associated blueprint listing in the crafting menu, rather than immediately craft the item.

Action_#_Blueprints uint8 `0`

Total number of blueprint indices.

Action_#_Key string

Translation key that should be used instead of a custom button name and tooltip. Valid translation keys and their localization can be found in the `PlayerDashboardInventory.dat` localization file.

Valid keys include: `Attachments`, `Craft_Bandage`, `Craft_Dressing`, `Craft_Rag`, `Craft_Seed`, `Dequip`, `Drop`, `Equip`, `Pickup`, `Refill`, `Repair`, `Salvage`, `Store`, and `Take`.

This property cannot be used in combination with `Action_#_Text` or `Action_#_Tooltip`. If set, the value of this property will always override any custom button name or tooltip that has been set.

Action_#_Source uint16 `0`

ID of the item to source actions from. Default source is the current item.

Action_#_Text string

Context button name. This property is usually used in combination with `Action_#_Tooltip`.

Action_#_Tooltip string

Context button tooltip. This property is usually used in combination with `Action_#_Text`.

Action_#_Type EActionType

Type of action to perform. Currently, only the `Blueprint` action type exists.

Actions int `0`

Total number of context actions.

18.3.2 Default Actions

Depending on the blueprints an item has, some blueprint actions may be automatically added to the item as well. These actions will be linked to the blueprint they were automatically generated from.

- When a blueprint only has one supply, with a supply ID of the item itself, an action using the `Salvage` localization key is generated.
- When a blueprint uses the `Repair` type, an action using the `Repair` localization key is generated.
- When a blueprint uses the `Refill` type, an action using the `Refill` localization key is generated.

18.4 Arrest End Assets

The ItemArrestEndAsset class is used for "releaser" items, which can remove a corresponding "*catcher*" item that is restraining a player. An example of a vanilla releaser is the Handcuffs Key.

18.4.1 Game Data File

Releasers inherit properties from the *ItemAsset* class. Any properties from parent classes that are required are listed in the table below.

Class	Property Name	Required Value
ItemAsset	*GUID*	
ItemAsset	*ID*	
ItemAsset	*Type*	`Arrest_End`
ItemAsset	*Useable*	`Arrest_End`

Properties

Property Name	Type	Default Value
Recover	*uint16*	`0`

Property Descriptions

Recover uint16 `0`

Legacy ID of a corresponding *catcher* item that can be unlocked with this item.

18.5 Arrest Start Assets

The ItemArrestStartAsset class is used for "catcher" items, which can restrain a player. Its sister item, the "*releaser*", can be used to unlock restraints. Some examples of vanilla catchers include the Handcuffs and Cable Tie.

18.5.1 Game Data File

Catchers inherit properties from the *ItemAsset* class. Any properties from parent classes that are required are listed in the table below.

Class	Property Name	Required Value
ItemAsset	*GUID*	
ItemAsset	*ID*	
ItemAsset	*Type*	`Arrest_Start`
ItemAsset	*Useable*	`Arrest_Start`

Properties

Property Name	Type	Default Value
Strength	*uint16*	`0`

Property Descriptions

Strength uint16 `0`

Number of times a player must lean in order to break free from their restraints.

18.6 Backpack Assets

Backpacks can be worn by players and zombies.

This inherits the *BagAsset* class.

18.6.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Backpack`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

18.6.2 Backpack Asset Properties

Backpacks have no unique asset properties. Refer to parent classes for additional properties.

18.7 Bag Assets

Clothing bags can be worn by players and zombies.

This inherits the *ClothingAsset* class.

18.7.1 Bag Asset Properties

Width *byte*: Number of columns (horizontal storage space). Defaults to 0.

Height *byte*: Number of rows (vertical storage space). Defaults to 0.

18.8 Barrel Assets

Barrel attachments are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

18.8.1 Game Data File

Barrel attachments inherit properties from the CaliberAsset class, which in turn inherits properties from the ItemAsset class. Properties that are required to be included are listed in the table below.

Class	Property Name	Required Value
ItemAsset	*GUID*	
ItemAsset	*ID*	
ItemAsset	*Type*	`Barrel`

Properties

Property Name	Type	Default Value
Ballistic_Drop	*float32*	1
Braked	*flag*	
Durability	*uint8*	`0`
Gunshot_Rolloff_Distance_Multiplier	*float32*	See description
Silenced	*flag*	
Volume	*float32*	1

Property Descriptions

Ballistic_Drop float32 `1`

Gravity acceleration multiplier for bullets in flight.

Braked flag

Muzzle flash should be hidden.

Durability uint8 `0`

Amount of quality lost after each firing of the ranged weapon. When this value is greater than `0`, the item always has a visible item quality shown.

Gunshot_Rolloff_Distance_Multiplier float32

Multiplier on gunshot rolloff distance. Defaults to `0.5` if `Silenced`, otherwise to `1`.

Silenced flag

Alerts should not be generated when firing.

Volume float32 1

Multiplier on gunfire sound volume. This is often used alongside with `Silenced`, but doing so is not required.

18.9 Barricade Assets

Barricades can be placed by players.

This inherits the *PlaceableAsset* class.

18.9.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Barricade`): When intending to use a child class, refer to that class's documentation instead for the proper enumerator to use.

Useable *enum* (`Barricade`)

Build *enum* (`Barrel_Rain`, `Barricade`, `Barricade_Wall`, `Beacon`, `Bed`, `Cage`, `Campfire`, `Charge`, `Claim`, `Clock`, `Door`, `Farm`, `Fortification`, `Freeform`, `Gate`, `Generator`, `Glass`, `Hatch`, `Ladder`, `Library`, `Mannequin`, `Note`, `Oil`, `Oven`, `Oxygenator`, `Safezone`, `Sentry_Freeform`, `Sentry`, `Shutter`, `Sign_Wall`, `Sign`, `Spike`, `Spot`, `Stereo`, `Storage_Wall`, `Storage`, `Tank`, `Torch`, `Vehicle`, `Wire`): Some values may not function properly without using a child class instead. When intending to use a child class, refer to that class's documentation instead for the proper enumerator to use.

ID *uint16*: Must be a unique identifier.

InventoryAudio *Master Bundle Pointer*: See *ItemAsset* for full documentation. Defaults to `Sounds/Inventory/Seeds.asset` if the name contains the word "Seed", to `Sounds/Inventory/SmallMetal.asset` if the name contains the word "Metal", to `Sounds/Inventory/LightMetalEquipment.asset` if either `Size_X` or `Size_Y` value is equal to 1, to `Sounds/Inventory/MediumMetalEquipment.asset` if either `Size_X` or `Size_Y` value is less than or equal to 2, or `Sounds/Inventory/HeavyMetalEquipment.asset` if none of the criteria is met.

18.9.2 Barricade Asset Properties

Allow_Collision_While_Animating *bool*: Whether or not animated interactables should have collision during their animation. If true, animated colliders are enabled while playing the animation even when a player is overlapping. Be wary when enabling this, as it can allow for physics-based exploits such as those involving doors. Defaults to false.

Allow_Placement_Inside_Clip_Volumes *bool*: If true, the barricade can be placed inside of player clip volumes. Defaults to false, except when using `Build Charge`.

Allow_Placement_On_Vehicle *bool*: If true, this barricade can be placed on vehicles. Defaults to true, except when using `Build Bed`, `Build Sentry`, or `Build Sentry_Freeform`.

Armor_Tier *enum* (`Low`, `High`): Barricade armor can either be low-tier or high-tier. Defaults to low-tier, except when the barricade's name contains the word "Metal". By default, barricades with low-tier armor take 100% of the damage they receive, while barricades with high-tier armor take 50% of the damage they receive. These multipliers can be configured in the gameplay config.

Bypass_Claim *flag*: Can be placed inside someone else's claimed area.

Bypass_Pickup_Ownership *bool*: If true, non-owners of the placed barricade can pick it up. Defaults to false, except when using `Build Charge`.

Can_Be_Damaged *bool*: If true, this barricade can be damaged. Defaults to true.

Eligible_For_Pooling *bool*: If true, this barricade is eligible for object pooling. Some barricades may not reset properly when pooling is enabled. Defaults to true, except when using `Build Beacon`.

Explosion *GUID* or *uint16*: GUID or legacy ID of *EffectAsset* to play when destroyed. When using `Build Vehicle`, this is instead the GUID or legacy ID of the vehicle that should be spawned.

Has_Clip_Prefab *bool*: Whether or not the barricade has a Clip.prefab. If the barricade should use the same prefab on the server as on the client, set to false. For example, most official content uses `Has_Clip_Prefab false`. Defaults to true.

Health *uint16*: Total health value. Defaults to 0.

Locked *flag*: Only the placed barricade's owner(s) can interact with it.

Offset *float*: In meters, the distance above the ground the barricade is placed.

PlacementAudioClip *Master Bundle Pointer*: AudioClip to play when the barricade is placed.

PlacementPreviewPrefab *Master Bundle Pointer*: Overrides the placement preview model spawned when this item is held.

Proof_Explosion *flag*: Immune to area-of-effect explosive damage.

Radius *float*: In meters, the radius around the barricade the must be clear in order for it to be placeable.

Range *float*: In meters, the maximum distance away the barricade can be placed from the player.

Salvage_Duration_Multiplier *float*: Multiplier on how long it takes to salvage this barricade. Setting this to a larger number will cause salvaging to take longer. Defaults to 1.

Unpickupable *flag*: Disables the ability to pick up a placed barricade. For example, the Horde Beacon uses this flag.

Unrepairable *flag*: Cannot be repaired by a *MeleeAsset* with the `Repair` flag. For example, the Blowtorch would not be able to repair this barricade.

Unsalvageable *flag*: Salvaging a damaged barricade yields no partial resources. For example, small glass plates use this flag.

Unsaveable *flag*: This barricade is excluded from being saved. For example, carepackages use this flag.

Use_Water_Height_Transparent_Sort *flag*: Useful for transparent barricades, such as glass.

Vulnerable *flag*: The barricade can be damaged by lower-power weapons that do not have the `Invulnerable` flag.

18.10 Beacon Assets

Beacons are a placeable zombie horde spawners. Placing the beacon will start a horde event, which can be completed by killing a certain number of zombies without letting the beacon be destroyed.

This inherits the *BarricadeAsset* class.

18.10.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Beacon`)

Useable *enum* (`Barricade`)

Build *enum* (`Beacon`)

ID *uint16*: Must be a unique identifier.

18.10.2 Beacon Asset Properties

Wave *uint16*: Number of zombies that must be killed to complete the beacon. If there are not enough zombies in the area for the horde event to be completed, zombies will respawn continuously. The final zombie spawned by a horde event is guaranteed to be a Mega Zombie. Defaults to 0.

Rewards *byte*: Number of items to drop upon successfully completing the beacon. When using `Enable_Participant_Scaling true`, the number of rewards will scale based on the number of participants. Defaults to 0.

Reward_ID *uint16*: Legacy ID of the spawn table to use for the rewarded items. Defaults to 0.

Enable_Participant_Scaling *bool*: Whether or not zombie health, and rewards dropped, should scale based on the number of players. Zombie health scales linearly, which can be represented by `(Initial Participants) × 1.5`. Reward scaling has diminishing returns, with an equivalent formula being `7 * sqrt(Initial Participants)`. Defaults to true.

18.11 Box Assets

Boxes are intended to be used as a part of the Steam Economy, rather than as in-game content. As such, none of its unique properties can be properly utilized by modders.

This inherits the *ItemAsset* class.

18.11.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Box`)

ID *uint16*: Must be a unique identifier.

18.11.2 Box Asset Properties

Generate *int32*: The itemdefid granted by opening this box, which is used to display the correct UI element after an unbox.

Destroy *int32*: The itemdefid removed by opening this box, which is used to display the correct UI element after an unbox.

Drops *int32*: Corresponds to the total number of items in the box, so that the correct number of UI elements are displayed when showing box contents.

Drop_# *int32*: The itemdefid of an item in the box, which is visually displayed as a UI element when showing box contents.

Item_Origin *enum* (`Unbox`, `Unwrap`): The localization key to use for for the unbox/unwrap menu button.

Probability_Model *enum* (`Equalized`, `Original`): UI elements regarding unbox probability chances are added depending on the specified enumerator.

Contains_Bonus_Items *bool*: When true, adds a UI element regarding bonus items.

18.12 Caliber Assets

The ItemCaliberAsset class is a base class that other classes are derived from. It is unusable on its own.

18.12.1 Game Data File

The ItemCaliberAsset class inherits properties from the *ItemAsset* class.

Properties

Property Name	Type	Default Value
Aiming_Movement_Speed_Multiplier	*float32*	1
Aiming_Recoil_Multiplier	*float32*	1
Aim_Duration_Multiplier	*float32*	1
Ballistic_Damage_Multiplier	*float32*	See description
Calibers	*uint8*	`0`
Caliber_#	*uint16*	`0`
Damage	*float32*	*deprecated*
Destroy_Attachment_Colliders	*bool*	`true`
Firerate	*uint8*	`0`
Paintable	*flag*	
Recoil_X	*float32*	1
Recoil_Y	*float32*	1
Shake	*float32*	1
Spread	*float32*	1
Sway	*float32*	1

Property Descriptions

Aiming_Movement_Speed_Multiplier float32 1

Multiplier on character movement speed while aiming down sights.

Aiming_Recoil_Multiplier float32 `1`

Multiplier on recoil magnitude while aiming down sights.

Aim_Duration_Multiplier float32 `1`

Multiplier on the value of *Aim_In_Duration* property available to the *ItemGunAsset* class.

Ballistic_Damage_Multiplier float32

Multiplier on damage. Defaults to the value of the `Damage` property, or `1` if both properties are unset.

Caliber_# uint16 `0`

Legacy ID of a caliber to check for attachment compatibility. This property is used in conjunction with `Calibers`, which determines how many instances of this property should be read by the game.

When this property is unset, it will default to `0`. When the `Magazine_Calibers` property is not greater than `0`, this property will default to the value of `Caliber`.

Calibers uint8 `0`

Set the length of the array containing the calibers for attachment compatibility. This property is used in conjunction with the `Caliber_#` property, and the value of `Calibers` should be equal to the number of instances of `Caliber_#`.

Damage float32

Deprecated since version 3.27.0.0: Use `Ballistic_Damage_Multiplier` instead.

Maintained for backwards compatibility. If both this property and `Ballistic_Damage_Multiplier` have been set, the latter's value is used.

Destroy_Attachment_Colliders bool `true`

When `false`, colliders are not destroyed when the attached ranged weapon's colliders are destroyed. This property exists for backwards compatibility with mods that may have relied on attachments having colliders, but using this property is not recommended.

Caution: Mods with complex colliders on their custom attachments are frequently reported as causing low performance issues for players. It is recommended that your custom attachments do not rely on having colliders.

Firerate uint8 `0`

The value of the attached ranged weapon's *Firerate* property is reduced by the value of this property. A larger decrease will allow for the ranged weapon to fire more often.

Paintable flag

When this flag is included, the attachment should be affected by Steam Economy skins that include support for skinning attachments.

Recoil_X float32 `1`

Multiplier on horizontal recoil.

Recoil_Y float32 `1`

Multiplier on vertical recoil.

Shake float32 `1`

Multiplier on shake.

Spread float32 1

Multiplier on bullet spread.

Sway float32 1

Multiplier on scope sway.

18.13 Charge Assets

Remote explosives can be placed and then remotely detonated with a *remote trigger*.

This inherits the *BarricadeAsset* class.

18.13.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Charge`)

Useable *enum* (`Barricade`)

Build *enum* (`Charge`)

ID *uint16*: Must be a unique identifier.

18.13.2 Charge Asset Properties

Animal_Damage *float*: Damage dealt to animals caught within the area-of-effect explosion.

Barricade_Damage *float*: Damage dealt to barricades caught within the area-of-effect explosion.

Explosion2 *uint16* or *GUID*: ID or GUID of effect to play upon detonation.

Explosion_Launch_Speed *float*: Launch speed of players caught within the area-of-effect explosion, in meters per second. Defaults to the value of `Player_Damage * 0.1`.

Object_Damage *float*: Damage dealt to objects caught within the area-of-effect explosion. Defaults to the value of `Resource_Damage`.

Player_Damage *float*: Damage dealt to players caught within the area-of-effect explosion.

Resource_Damage *float*: Damage dealt to resources caught within the area-of-effect explosion.

Structure_Damage *float*: Damage dealt to structures caught within the area-of-effect explosion.

Vehicle_Damage *float*: Damage dealt to vehicles caught within the area-of-effect explosion.

Range2 *float*: Radius of the damaging, area-of-effect explosion.

Zombie_Damage *float*: Damage dealt to zombies caught within the area-of-effect explosion.

18.14 Clothing Assets

Clothing can be worn by players and zombies. Clothing items always show quality.

This inherits the *ItemAsset* class.

18.14.1 Clothing Asset Properties

Armor *float*: Multiplier on damage received. Defaults to 1.

Armor_Explosion *float*: Multiplier on damage received from area-of-effect explosions. Defaults to the value used for Armor.

Destroy_Clothing_Colliders *bool*: If false, colliders are not destroyed when the clothing is attached to the character. For example equipped vanilla clothes do not have any colliders, but some mods (e.g., armor with hitbox) may have relied on child colliders not being destroyed. Defaults to true.

Falling_Damage_Multiplier *float*: Multiplier on damage received from falling. Defaults to 1.

Prevents_Falling_Broken_Bones *bool*: If true on any worn clothing item, bones never break when falling. Defaults to false.

Priority_Over_Cosmetic *bool*: If set, overrides the default cosmetic override behavior. By default, night vision goggles and headlamps take priority over cosmetics in the same slot. If true, the in-game item is shown rather than a cosmetic in the same slot.

Proof_Water *flag*: Specified if it should exhibit the waterproof property. Only applicable to backpacks and glasses. When waterproof glasses are worn, the player will not have their screen blurred while underwater. When a waterproof backpack and waterproof glasses are worn together, the player's oxygen will deplete at a greatly reduced rate when underwater.

Proof_Fire *flag*: Specified if it should exhibit the fireproof property. Only applicable to shirts and pants. When a fireproof shirt and fireproof pants are worn together, the player will be immune to fire damage.

Proof_Radiation *flag*: Specified if it should exhibit the radiation-proof property. Only applicable to pants, shirts, and masks. When a radiation-proof mask is worn, the player will not be damaged by standard deadzones. When radiation-proof pants, a radiation-proof shirt, and a radiation-proof mask are worn together, the player will not be damaged by full-suit deadzones. The protection only lasts for as long as the radiation-proof mask's item quality remains greater than 0%. The mask's quality will deplete over time while inside of a deadzone. *Radiation filters* can be used to replenish a radiation-proof mask's quality.

Skin_Override *string*: Optional name of a renderer that should use the player's skin material. For example, to create a miniature version of the player sitting on their shoulder.

Mirror_Left_Handed_Model *bool*: Clothing should be mirrored when the player is left-handed. Only applicable to vests, backpacks, masks, glasses, and hats. Defaults to true.

Movement_Speed_Multiplier *float*: Multiplier on movement speed. Defaults to 1.

Hair_Visible *bool*: Hair should be visible. Defaults to true.

Beard_Visible *bool*: Facial hair should be visible. Defaults to true.

Visible_On_Ragdoll *bool*: Should be visible on ragdolls. Defaults to true.

WearAudio *Master Bundle Pointer*: AudioClip or OneShotAudioDefinition to play upon wearing.

18.15 Cloud Assets

Parachutes can affect a player's gravity when held.

This inherits the *ItemAsset* class.

18.15.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Cloud`)

Useable *enum* (`Cloud`)

ID *uint16*: Must be a unique identifier.

18.15.2 Cloud Asset Properties

Gravity *float*: Multiplier on the influence of gravity.

18.16 Consumeable Assets

Consumable items are irreversibly consumed by the player on use, and directly affect a player's stats such as food or health.

This inherits the *WeaponAsset* class.

18.16.1 Consumeable Asset Properties

Aid *flag*: Specified if the item can be used on other players, via the "Secondary" action.

Bleeding *flag*: Specified if the item should remove the "Bleeding" status effect. Deprecated in favor of Bleeding_Modifier.

Bleeding_Modifier *enum* (`Cut`, `Heal`, `None`): Determines the effect the consumable has in relation to the "Bleeding" status effect.

Broken *flag*: Specified if the item should remove the "Broken Bones" status effect. Deprecated in favor of Bones_Modifier.

Bones_Modifier *enum* (`Break`, `Heal`, `None`): Determines the effect the consumable has in relation to the "Broken Bones" status effect.

ConsumeAudioClip *Master Bundle Pointer*: AudioClip to play when the consumeable is used.

Disinfectant *byte*: Amount of immunity restored.

Energy *byte*: Amount of stamina restored.

Experience *int*: Amount of experience added or removed.

Explosion *uint16* or *GUID*: ID or GUID of the explosion effect to play upon consumption.

Food *byte*: Amount of food restored. If the amount of food to restore is larger than the amount of water to restore, then food constrains water.

Health *byte*: Amount of health restored.

Item_Reward_Spawn_ID *uint16*: ID of the item spawn table to generate an item from upon consuming the consumable. The number of items generated is random, depending on the range defined by Min_Item_Rewards and Max_Item_Rewards.

Max_Item_Rewards *int*: Maximum number of items that can be generated from the spawn table specified by Item_Reward_Spawn_ID.

Min_Item_Rewards *int*: Minimum number of items that can be generated from the spawn table specified by Item_Reward_Spawn_ID.

Oxygen *sbyte*: Amount of oxygen restored or depleted.

Should_Delete_After_Use *bool*: Boolean for if the item should be deleted after being consumed. Defaults to true.

Virus *byte*: Amount of immunity depleted.

Vision *uint*: Length of hallucinations, in seconds. The length does not stack when consuming multiple hallucinogenics. Instead, the timer is reset to the longer value.

Warmth *uint*: Amount of warmth added.

Water *byte*: Amount of water restored. If the amount of water to restore is less than the amount of food to restore, then water is constrained by food.

18.16.2 Rewards

Consumables can use quest rewards. A common usage is to create consumables with multiple (but still limited) uses, by placing a new item in the player's inventory after consuming the original. Alternatively, consuming a consumable may be required to complete a quest. Refer to *Rewards* documentation for additional documentation.

These rewards are prefixed with `Quest_`. For example, `Quest_Rewards 1`.

18.17 Detonator Assets

Remote triggers can be used to detonate *remote explosives*.

This inherits the *ItemAsset* class.

18.17.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Detonator`)

Useable *enum* (`Detonator`)

ID *uint16*: Must be a unique identifier.

18.17.2 Detonator Asset Properties

Remote triggers have no unique asset properties. Refer to *item asset documentation* for additional properties.

18.18 Farm Assets

Farms (localized as "plants") are a placeable seeds capable of growing into harvestable crops. When a seed is planted, it will grow over time until eventually harvestable. Growing can be finished immediately by either rainfall, or by using a *growth supplement* on the plant. A fully-grown crop can be harvested, which deals 2 damage to the crop. A crop can be harvested until it has 0 health remaining.

This inherits the *BarricadeAsset* class.

18.18.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Farm`)

Useable *enum* (`Barricade`)

Build *enum* (`Farm`)

ID *uint16*: Must be a unique identifier.

18.18.2 Farm Asset Properties

Affected_By_Agriculture_Skill *bool*: If true, the amount of crops acquired when harvesting the plant is affected by the Agriculture skill. Defaults to true.

Allow_Fertilizer *bool*: If true, allows the player to use fertilizer to fully grow the plant. Defaults to true.

Grow *ushort*: Legacy ID of the item to spawn when harvested.

Grow_SpawnTable *GUID*: GUID of a spawntable from which to spawn an item when harvested.

Growth *uint*: In seconds, how long before the crop is fully grown.

Harvest_Reward_Experience *uint*: The amount of experience gained upon harvesting. Defaults to 1.

Ignore_Soil_Restrictions *bool*: If false, only allow placement on Soil Materials. If true, allow placement anywhere. Default to false.

Rain_Affects_Growth *bool*: If true, the plant will fully finish growing after rainy weather. Defaults to true.

Harvest_Rewards: NPC reward list granted when harvesting the grown plant. For more information, please refer to the *Rewards* documentation.

Tip: The `Health` property from the parent ItemAsset class can be configured to allow for harvesting a crop multiple times. A plant can be harvested a number of items equal to `Health / 2`. For example, a plant with 10 health can be harvested up to 5 times.

18.19 Filter Assets

Radiation filters can be used to replenish the quality of radiation-proof *masks*.

This inherits the *ItemAsset* class.

18.19.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Filter`)

Useable *enum* (`Filter`)

ID *uint16*: Must be a unique identifier.

18.19.2 Filter Asset Properties

Radiation filters have no unique asset properties. Refer to parent classes for additional properties.

18.20 Fisher Assets

Fishers (localized as "fishing poles") are useables that allow for catching fish.

This inherits the *ItemAsset* class.

18.20.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Fisher`)

Useable *enum* (`Fisher`)

ID *uint16*: Must be a unique identifier.

18.20.2 Fisher Asset Properties

Reward_ID *uint16*: Legacy ID of the spawn table a reward should be generated from upon successfully catching something with the fishing pole.

18.21 Food Assets

Food is irreversibly consumed by the player on use, and directly affect a player's stats such as food or health.

This inherits the *ConsumeableAsset* class.

18.21.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Food`)

Useable *enum* (`Consumeable`)

ID *uint16*: Must be a unique identifier.

18.21.2 Food Asset Properties

Food have no unique asset properties. Refer to parent classes for additional properties.

18.22 Fuel Assets

Fuel canisters are useables able to siphon, store, and deposit fuel.

This inherits the *ItemAsset* class.

18.22.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Fuel`)

Useable *enum* (`Fuel`)

ID *uint16*: Must be a unique identifier.

18.22.2 Fuel Asset Properties

Always_Spawn_Full *bool*: If true, this item will always spawn filled at full capacity.

Delete_After_Filling_Target *bool*: If true, this item is removed from the player's inventory after adding fuel to target.

Fuel *uint16*: Maximum units of fuel that can be stored in the fuel canister.

18.23 Gear Assets

Clothing gear can be worn by players and zombies. The inherited Hair_Visible and Beard_Visible properties default to the gear asset's corresponding Hair and Beard flag properties.

This inherits the *ClothingAsset* class.

18.23.1 Gear Asset Properties

Hair *flag*: Specified if hair should be visible.

Beard *flag*: Specified if facial hair should be visible.

Hair_Override *string*: When this property is set, the game will look for a child Mesh Renderer component in Unity that has the same name as this property's value. If a matching Mesh Renderer is found, its material will be changed to the character's hair material. This property is used by certain cosmetics that entirely cover the character's hair, so that the player's selected hair color can still be used for customization.

18.24 Generator Assets

Generators are a placeable power sources. Players can deposit fuel into generators with a *fuel canister*.

This inherits the *BarricadeAsset* class.

18.24.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Generator`)

Useable *enum* (`Barricade`)

Build *enum* (`Generator`)

ID *uint16*: Must be a unique identifier.

18.24.2 Generator Asset Properties

Capacity *uint16*: Maximum units of fuel that can be stored in the generator. Defaults to 0.

Wirerange *float* [0, 256]: In meters, the radius around the generator that is provided electricity.

Burn *float*: How many seconds it takes to burn one unit of fuel.

18.25 Glasses Assets

Glasses can be worn by players and zombies.

This inherits the *GearAsset* class.

18.25.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Glasses`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

18.25.2 Glasses Asset Properties

Blindfold *flag*: Specified if glasses should blacken the player's screen.

Nightvision_Allowed_In_ThirdPerson *bool*: If `true`, nightvision works in third-person, not just first-person. Defaults to `false` for backwards compatibility. Vanilla nightvision has this set to true.

Nightvision_Color *color*: Overrides the default color when using `Vision Military`. This property supports using legacy color parsing.

Nightvision_Fog_Intensity *float*: Intensity of fog while nightvision is active.

Vision *enum* (`None`, `Military`, `Civilian`, `Headlamp`): Type of unique lighting vision effect to use. Defaults to `None`. When intending to assign a custom nightvision color via the `Nightvision_Color` property, it is recommended to use the `Military` enumerator.

18.26 Grip Assets

Grip attachments are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

18.26.1 Game Data File

Grip attachments inherit properties from the CaliberAsset class, which in turn inherits properties from the ItemAsset class. Properties that are required to be included are listed in the table below.

Class	Property Name	Required Value
ItemAsset	*GUID*	
ItemAsset	*ID*	
ItemAsset	*Type*	`Grip`

Properties

Property Name	Type	Default Value
Bipod	*flag*	

Property Descriptions

Bipod flag

Stat-changing properties should only take effect while prone.

18.27 Grower Assets

Growth supplements can be used to instantly finish growing a *plant*.

This inherits the *ItemAsset* class.

18.27.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Grower`)

Useable *enum* (`Grower`)

ID *uint16*: Must be a unique identifier.

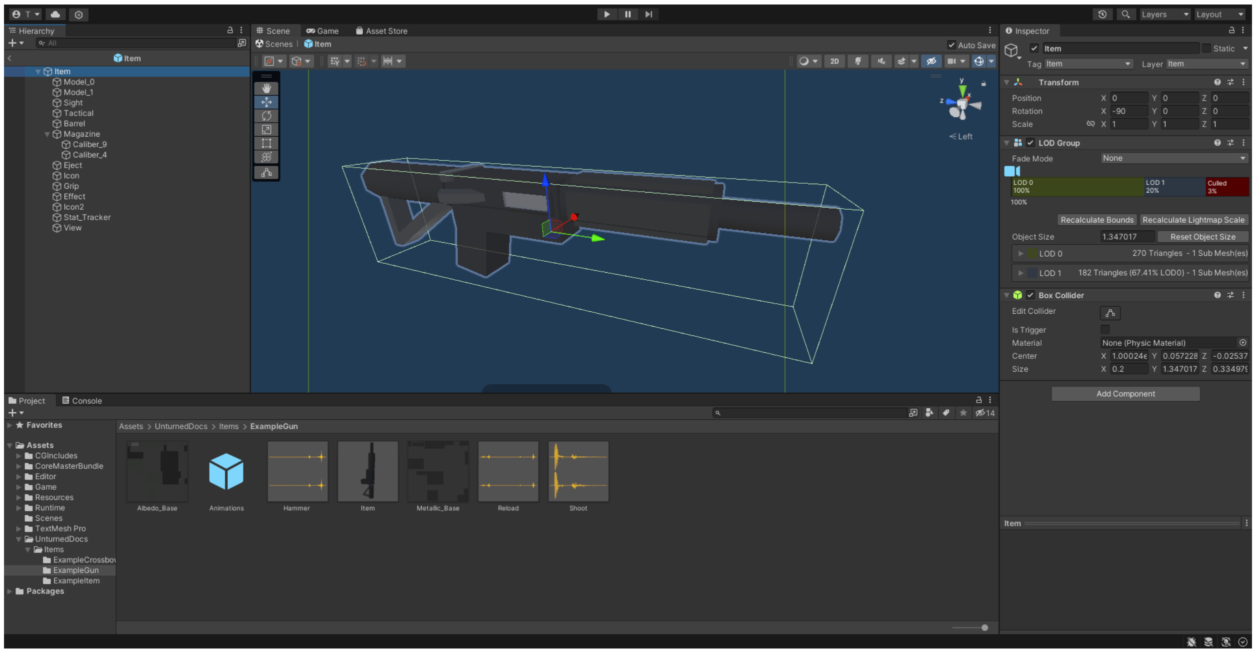
18.27.2 Grower Asset Properties

Growth supplements have no unique asset properties. Refer to *item asset documentation* for additional properties.

18.28 Gun Assets

The ItemGunAsset class is used for ranged weapons (or "guns"), which can be used by players to deal damage. Some examples of vanilla ranged weapons include the Eaglefire and Crossbow.

18.28.1 Unity Asset Bundle Contents



```
Item
  Model_0
  Model_1
  Sight
  Tactical
  Barrel
  Magazine
    Caliber_9
    Caliber_4
  Eject
  Icon
  Grip
  Effect
  Icon2
  Stat_Tracker
  View
```

```
Item                      Static
Tag Item                  Layer Item
Transform
  Position   X 0      Y 0      Z 0
  Rotation   X -90    Y 0      Z 0
  Scale      X 1      Y 1      Z 1
LOD Group
  Fade Mode  None
  LOD 0 100%   LOD 1 20%   Culled 3%
  Object Size  1.347017
  LOD 0   270 Triangles - 1 Sub Mesh(es)
  LOD 1   182 Triangles (67.41% LOD0) - 1 Sub Mesh(es)
Box Collider
  Is Trigger
  Material   None (Physic Material)
  Center     X 1.00024e  Y 0.057228  Z -0.02537
  Size       X 0.2       Y 1.347017  Z 0.33497
```

Fig. 2: An example of a gun being set up in the Unity editor.

To get started, either follow the steps to begin creating a custom item from the *introduction*, or duplicate the contents of a prepackaged example asset.

Item (Prefab)

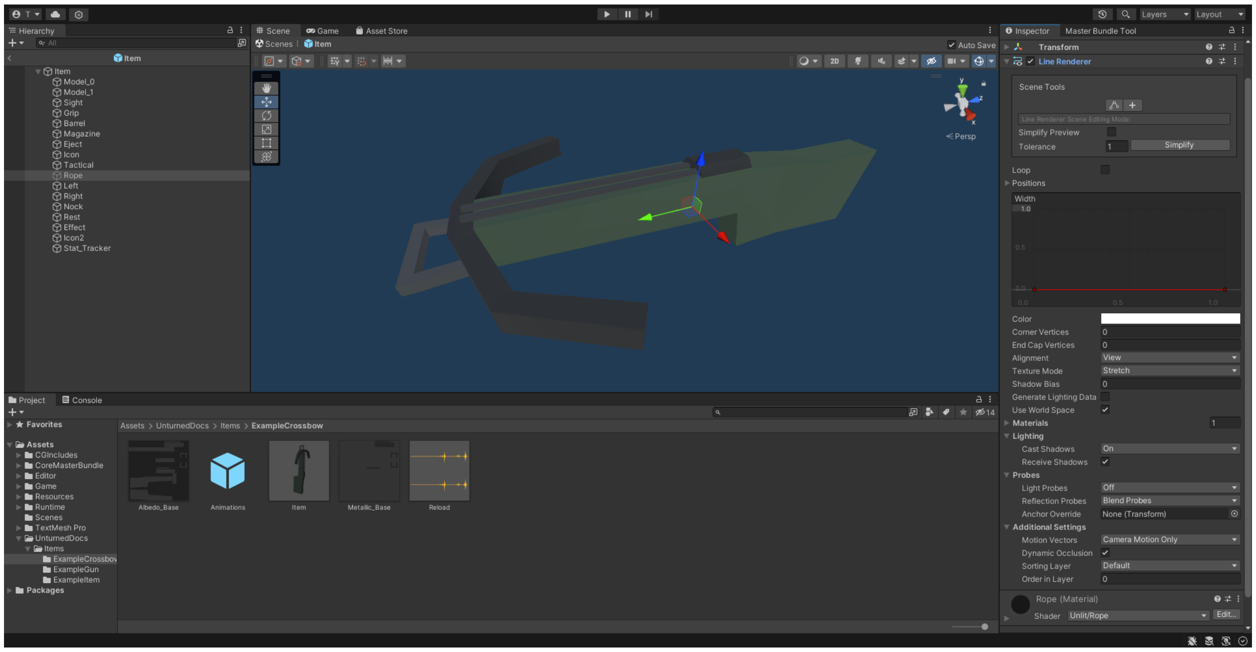
Open the "Item" Prefab, and add six child GameObjects named "Barrel", "Grip", "Sight", "Tactical", "Magazine", and "Eject". Most custom guns will want to have these six child GameObjects, although they are not strictly required.

The "Barrel", "Grip", "Sight", "Tactical", and "Magazine" GameObjects will determine the location of attachments on your gun. The "Sight" GameObject also determines where the camera will be positioned when aiming down sights. Shells are emitted from the "Eject" GameObject.

If an "View" GameObject is added, the camera will use its position when aiming down sights if a sight attachment has not been attached to the gun.

When a gun can accept more than one type of magazine caliber, it may be desirable to have the position of the magazine attachment depend on its caliber ID. Add a child to the "Magazine" GameObject, named "Caliber_#". For example, adding "Caliber_1" would cause magazine attachments using caliber ID 1 to use that position instead of the "Magazine" GameObject's position.

Additional Setup for Bows



Simplify Preview
Tolerance 1
Loop
Positions
Width 1.0
Color
Corner Vertices 0
End Cap Vertices 0
Alignment View
Texture Mode Stretch
Shadow Bias 0
Generate Lighting Data
Use World Space ✓
Materials 1
Lighting
Cast Shadows On
Receive Shadows ✓
Probes
Light Probes Off
Reflection Probes Blend Probes
Anchor Override None (Transform)
Additional Settings
Motion Vectors Camera Motion Only
Dynamic Occlusion ✓
Sorting Layer Default
Order in Layer 0
Rope (Material)
Shader Unlit/Rope

Fig. 3: An example of a crossbow being set up in the Unity editor.

Bows require additional GameObjects to simulate the drawing of the bowstring. Note that bowstrings are only simulated from the first-person perspective.

Add a new child GameObject named "Rope", and set its state to inactive. The "Rope" GameObject should include a Line Renderer component. Vanilla bowstrings use a custom Material named "Rope" with the Unlit-Rope Shader, but this is not required.

Add two child GameObjects named "Left" and "Right". These GameObjects will determine the end points of the bowstring. If a third GameObject named "Rest" is included, it will be used as the middle point of the bowstring when aiming down sights.

Including a fourth GameObject named “Nock” will allow the bow to be fired without aiming down sights. Additionally, the “Rest” GameObject will act as a middle point when not aiming down sights, and the “Nock” GameObject will act as a middle point while aiming down sights.

Additional Setup for Economy Items

There are several child GameObjects that can be added related to skins. Custom items are ineligible to receive skins, so there is usually no reason to add these to the Prefab.

If an item has an “Icon2” GameObject included, its position and orientation will be used when generating icons of skins on this item. A GameObject named “Stat_Tracker” determines the location where stat trackers will appear on the gun, while a GameObject named “Effect” will determine the position of mythical effects on the gun.

Animations (Prefab)

In addition to animations used by any equippable item, guns have an additional set of animations that they can use.

Adding animations named “Aim_Start” and “Aim_Stop” will cause an animation to be played whenever the player starts or stops aiming down sights. Animations named “Attach_Start” and “Attach_Stop” will play when an attachment is attached or unattached to the gun. The “Sprint_Start” and “Sprint_Stop” animations play when the player starts and stops sprinting. The “Reload” animation is played when reloading the gun.

The “Hammer” animation is played under certain conditions where it would make sense to manually eject a cartridge from the gun. For example: after reloading an gun that had an empty magazine, or after firing a single-shot weapon (such as a bolt-action rifle or pump-action shotgun).

If a gun is configured to use the gun jamming feature, the “UnjamChamber” animation will play when a jam occurs.

Audio Clips

In addition to the Audio Clips that can be included for equippable items, guns have an additional set of audio clips they can use.

If an Audio Clip named “Shoot” is included, it will play after the gun is fired. Including Audio Clips named “Reload” and “Hammer” will cause audio to play after reloading and hammering the gun, respectively.

An “Aim” Audio Clip can be included to have audio play after aiming down sights. For example, a longbow might want to have an the sound of the bow being drawn play. Miniguns can also include an Audio Clip named “Minigun” to have audio play while revving the minigun.

If a gun is configured to use the gun jamming feature, the “ChamberJammed” Audio Clip will play when a jam occurs.

18.28.2 Game Data File

Ranged weapons inherit properties from the *ItemWeaponAsset* class. Any properties from parent classes that are required—or highly recommended—are listed in the table below.

Class	Property Name	Required Value
ItemAsset	*GUID*	
ItemAsset	*ID*	
ItemAsset	*Slot*	
ItemAsset	*Type*	`Gun`
ItemAsset	*Useable*	`Gun`
ItemWeaponAsset	*Range*	

Additionally, all ranged weapons require that the `Action` property has been configured. Note that ranged weapons will always show a quality value.

Properties

Ranged weapons have a significant number of properties. To make navigating these easier, they have been categorized into one of several property tables. Many of these tables contain similar properties that are often used together.

Table 2: Uncategorized

Property Name	Type	Default Value
Aim_In_Duration	*float32*	`0.2`
Aiming_Movement_Speed_Multiplier	*float32*	See description
Alert_Radius	*float32*	`48`
Can_Aim_During_Sprint	*bool*	`false`
Gunshot_Rolloff_Distance	*float32*	See description
Range_Rangefinder	*float32*	See description
Scale_Aim_Animation_Speed	*bool*	`true`

Table 3: Calibers

Property Name	Type	Default Value
Attachment_Caliber_#	*uint16*	See description
Attachment_Calibers	*int32*	See description
Caliber	*uint16*	`0`
Magazine_Caliber_#	*uint16*	See description
Magazine_Calibers	*int32*	See description
Requires_NonZero_Attachment_Caliber	*bool*	`false`

Table 4: Damage

Property Name	Type	Default Value
Damage_Falloff_Max_Range	*float32*	`1`
Damage_Falloff_Multiplier	*float32*	`1`
Damage_Falloff_Range	*float32*	`1`
Instakill_Headshots	*bool*	`false`

Table 5: Effects

Property Name	Type	Default Value
Explosion	*GUID* or *uint16*	`0`
Muzzle	*GUID* or *uint16*	`0`
Shell	*GUID* or *uint16*	See description

Table 6: Firing Mechanism

Property Name	Type	Default Value
Action	*EAction*	
Auto	*flag*	
Bursts	*int32*	`0`
Fire_Delay_Seconds	*int32*	`0`
Firerate	*uint8*	`0`
Safety	*flag*	
Semi	*flag*	

Table 7: Hook Attachments

Property Name	Type	Default Value
Barrel	*uint16*	`0`
Grip	*uint16*	`0`
Sight	*uint16*	`0`
Tactical	*uint16*	`0`
Hook_Barrel	*flag*	
Hook_Grip	*flag*	
Hook_Sight	*flag*	
Hook_Tactical	*flag*	

Table 8: Jamming

Property Name	Type	Default Value
Can_Ever_Jam	*flag*	
Jam_Quality_Threshold	*float32*	`0.4`
Jam_Max_Chance	*float32*	`0.1`
Unjam_Chamber_Anim	*string*	`UnjamChamber`

Table 9: Magazine Attachments

Property Name	Type	Default Value
Allow_Magazine_Change	*bool*	`true`
Ammo_Max	*uint8*	`0`
Ammo_Min	*uint8*	`0`
Ammo_Per_Shot	*uint8*	`1`
Delete_Empty_Magazines	*flag*	*deprecated*
Hammer_Time	*float32*	`1`
Infinite_Ammo	*bool*	`false`
Magazine	*uint16*	`0`
Magazine_Replacement_#_ID	*uint16*	`0`
Magazine_Replacement_#_Map	*string*	
Magazine_Replacements	*int32*	`0`
Reload_Time	*float32*	`1`
Replace	*float32*	`1`
Should_Delete_Empty_Magazines	*bool*	See description
Unplace	*float32*	`0`

Table 10: Projectiles (Ballistic System)

Property Name	Type	Default Value
Ballistic_Drop	*float32*	*deprecated*
Ballistic_Steps	*uint8*	See description
Ballistic_Travel	*float32*	See description
Bullet_Gravity_Multiplier	*float32*	`4`

Table 11: Projectiles (Physics System)

Property Name	Type	Default Value
Ballistic_Force	*float32*	`0.002`
Projectile_Explosion_Launch_Speed	*float32*	See description
Projectile_Lifespan	*float32*	`30`
Projectile_Penetrate_Buildables	*flag*	

Table 12: Recoil

Property Name	Type	Default Value
Aiming_Recoil_Multiplier	*float32*	`1`
Recoil_Crouch	*float32*	`0.85`
Recoil_Max_X	*float32*	`0`
Recoil_Max_Y	*float32*	`0`
Recoil_Min_X	*float32*	`0`
Recoil_Min_Y	*float32*	`0`
Recoil_Prone	*float32*	`0.7`
Recoil_Sprint	*float32*	`1.25`
Recover_X	*float32*	`0`
Recover_Y	*float32*	`0`

Table 13: Shake

Property Name	Type	Default Value
Shake_Max_X	*float32*	`0`
Shake_Min_X	*float32*	`0`
Shake_Max_Y	*float32*	`0`
Shake_Min_Y	*float32*	`0`
Shake_Max_Z	*float32*	`0`
Shake_Min_Z	*float32*	`0`

Table 14: Spread

Property Name	Type	Default Value
Spread_Aim	*float32*	`0`
Spread_Angle_Degrees	*float32*	`0`
Spread_Crouch	*float32*	`0.85`
Spread_Hip	*float32*	*deprecated*
Spread_Prone	*float32*	`0.7`
Spread_Sprint	*float32*	`1.25`

EAction Enumeration

Named Value	Description
`Trigger`	Corresponds to the “Trigger” action. Uses the ballistic projectile system.
`Bolt`	Corresponds to the “Bolt” action. Uses the ballistic projectile system.
`Pump`	Corresponds to the “Pump” action. Uses the ballistic projectile system.
`Rail`	Corresponds to the “Rail” action. Uses the ballistic projectile system.
`String`	Corresponds to the “String” action. Uses the ballistic projectile system.
`Break`	Corresponds to the “Break” action. Uses the ballistic projectile system.
`Rocket`	Corresponds to the “Rocket” action. Uses the physics projectile system.
`Minigun`	Corresponds to the “Minigun” action. Uses the ballistic projectile system.

Property Descriptions

Action EAction

The value of this property determines how the weapon functions when used, including whether it uses *ballistic projectiles*, or *physics projectiles*. Different properties are available to the weapon depending on the value of this property.

Although most action mechanisms utilize ballistic projectiles, the `Rocket` action mechanism uses physics projectiles instead. Additionally, any projectiles from these weapons (e.g., the Rocket Launcher) are explosive.

To fire a weapon with the `String` action mechanism, a player must be aiming down sights – unless a “Nock” GameObject has been added during its Unity setup.

Aim_In_Duration float32 `0.2`

How long it takes to fully aim down sights, in seconds.

Aiming_Movement_Speed_Multiplier float32

Multiplier on the player's movement speed while aiming down sights. Defaults to `0.75` when `Can_Aim_During_Sprint` is `false`. Otherwise, defaults to `1`.

Aiming_Recoil_Multiplier float32 `1`

Multiplier on recoil magnitude while aiming down sights.

Alert_Radius float32 `48`

The radius of the alert generated by ranged weapons when they are fired. Zombies or animals caught within this radius are alerted. This radius is measured in meters.

Allow_Magazine_Change bool `true`

When `false`, the magazine cannot be removed, replaced, or reloaded. This functions similar to a few other properties, such as `Hook_Barrel` or `Hook_Grip` when determing valid hook attachment slots.

Ammo_Max uint8 `0`

Maximum amount of ammo to randomly generate in the magazine attachment that was attached to the weapon by default.

Ammo_Min uint8 `0`

Minimum amount of ammo to randomly generate in the magazine attachment that was attached to the weapon by default.

Ammo_Per_Shot uint8 1

Number of ammunition consumed per shot. For example, a value of 3 would consume three ammo every time the weapon is fired, while a value of 0 would allow for the weapon to have infinite ammo.

Attachment_Caliber_# uint16

Legacy ID of a caliber to check for hook attachment compatibility. This property is used in conjunction with Attachment_Calibers, which determines how many instances of this property should be read by the game.

When this property is unset, it will default to 0. When the Attachment_Calibers property is not greater than 0, this property will default to the value of any Magazine_Caliber_# properties.

For example, a valid configuration for a ranged weapon's calibers could be:

```
Attachment_Calibers 2
Attachment_Caliber_0 1
Attachment_Caliber_1 9

Magazine_Calibers 3
Magazine_Caliber_0 1
Magazine_Caliber_1 4
Magazine_Caliber_2 9
```

This would allow the ranged weapon to use hook attachments with caliber IDs of 1 or 9, and to use magazine attachments with caliber IDs of 1, 4, or 9.

Attachment_Calibers int32

Set the length of the array containing the calibers for hook attachment compatibility. This property is used in conjunction with the Attachment_Caliber_# property, and the value of Attachment_Calibers should be equal to the number of instances of Attachment_Caliber_#.

When this property is not greater than 0 – it will default to the value of Magazine_Calibers, and the Attachment_Caliber_# property can no longer be customized.

To use this property, Magazine_Calibers must be configured.

Auto flag

The weapon has an automatic firing mode.

Ballistic_Drop float32

Deprecated since version 3.23.7.0: Use `Bullet_Gravity_Multiplier` instead.

Existing values are automatically converted if `Bullet_Gravity_Multiplier` has not been configured. The conversion is logged during *Asset Validation*.

Ballistic_Force float32 `0.002`

The amount of force that should be applied to the *physics projectile*, measured in Newtons. It may be helpful to read Unity's Rigidbody.AddForce documentation to better understand physics projectiles.

Properties used by physics projectiles (such as `Ballistic_Force`) cannot be used alongside properties intended for ballistic projectiles (such as `Ballistic_Travel` or `Bullet_Gravity_Multiplier`).

Ballistic_Steps uint8

Lifespan of *ballistic projectiles*. A higher value relative to `Ballistic_Travel` will result in less muzzle velocity. Must be a value greater than `0`.

Defaults to `Range ÷ Ballistic_Travel`, rounded up to the nearest integer.

To avoid a mismatch between the weapon's max range and its manual ballistic range, it is recommend to only configure `Ballistic_Steps` or `Ballistic_Travel` (or neither) – no both.

Ballistic_Travel float32

Travel speed of *ballistic projectiles*. A higher value relative to `Ballistic_Steps` will result in more muzzle velocity. Must be a value greater than `0.1`.

Defaults to `10`. If `Ballistic_Steps` is specified and greater than `0`, and `Ballistic_Travel` is not specified, then `Ballistic_Travel` defaults to `Range ÷ Ballistic_Steps`.

To avoid a mismatch between the weapon's max range and its manual ballistic range, it is recommend to only configure `Ballistic_Travel` or `Ballistic_Steps` (or neither) – no both.

Barrel uint16 `0`

Legacy ID of a barrel attachment that should be attached by default. The `Hook_Barrel` flag is not required to use this property.

Bullet_Gravity_Multiplier float32 `4`

Multiplier for gravity's acceleration. This property is available to *ballistic projectile* weapons. Setting this value to `1` allows for more realistic bullet drop.

Note: This defaults to `4` because *Unturned*'s maximum engagement distance is rather short, but this distance may be raised in the future if/when network improvements are made to the game. Gravity defaults to 9.81 m/s^2, or can be configured in the *Level Config*.

Bursts int32 `0`

When a value greater than `0` is provided, the weapon has a burst firing mode. A number of shots equal to this value is fired when using this mode.

Caliber uint16 `0`

Legacy ID of the caliber to check for hook attachment and magazine attachment compatibility. To add compatibility for multiple calibers, or to configure hook attachment and magazine attachment compatibility separately, use the `Magazine_Calibers` and `Attachment_Calibers` properties instead.

Can_Aim_During_Sprint bool `false`

When `true`, the player can sprint while aiming down sights.

Can_Ever_Jam flag

When this flag is included, the weapon can jam. Weapons have a chance of jamming once their quality drops below a certain threshold. Starting from the initial threshold, the chance of jamming on each shot is blended between between 0% and a specified max chance.

The "ChamberJammed" Audio Clip is played when a jam occurs, as well as the animation "UnjamChamber" if present.

For an example, refer to `.../Guns/Cobra_Jam/Cobra_Jam.dat` in the game files.

Damage_Falloff_Max_Range float32 `1`

Percentage of maximum range where damage stops decreasing. For example, a max falloff range value of `0.6` with a range of `200` means damage stops dropping off after 120 meters.

Damage_Falloff_Multiplier float32 `1`

Percentage of damage to apply at maximum range. For example, a falloff multiplier value of `0.25` with a damage value of `40` means 10 damage will be dealt at maximum range.

Damage_Falloff_Range float32 `1`

Percentage of maximum range where damage begins decreasing. For example, a falloff range value of `0.3` with a range value of `200` means damage begins dropping off after 60 meters.

Delete_Empty_Magazines flag

Deprecated since version 3.30.3.0: Use `Should_Delete_Empty_Magazines` instead.

When this flag is included, the attached magazine attachment is deleted when fully depleted.

Explosion GUID or uint16

GUID or legacy ID of the effect that should be used for explosions caused by `Action Rocket` projectiles.

Fire_Delay_Seconds int32 `0`

Delay before the weapon is actually fired, in seconds.

Firerate uint8 `0`

The value of this property affects the minimum number of ticks between the firing of consecutive shots. A higher `Firerate` value will cause the weapon to have a slower rate of a fire. The weapon's rate of fire can be calculated with `50 ÷ (Firerate + 1)`, as the rounds per second.

Grip uint16 `0`

Legacy ID of a grip attachment that should be attached by default. The `Hook_Grip` flag is not required to use this property.

Gunshot_Rolloff_Distance float32

Distance over which the gunshot audio rolls off until it is completely inaudible, in meters. Defaults to `16` when using `Action String`; defaults to `64` when using `Action Rocket`; otherwise, defaults to `512`.

Hammer_Time float32 `1`

Multiplier on the time it takes to pull back the hammer a ranged weapon after firing. This does not affect the actual animation speed, but the cooldown before the player can perform other actions (such as shooting) again. Values less than `1` have no effect.

Hook_Barrel flag

When this flag is included, the ranged weapon has a barrel attachment slot.

Hook_Grip flag

When this flag is included, the ranged weapon has a grip attachment slot.

Hook_Sight flag

When this flag is included, the ranged weapon has a sight attachment slot.

Hook_Tactical flag

When this flag is included, the ranged weapon has a tactical attachment slot.

Infinite_Ammo bool `false`

When `true`, ammunition is not depleted from the magazine attachment. This allows for the weapon to have infinite ammo, so long as a magazine attachment with a number of rounds remaining equal to `Ammo_Per_Shot` is attached.

Instakill_Headshots bool `false`

If `true`, a player that is headshot with this weapon is instantly killed. This does not affect zombies, unless the world's difficulty configuration has the `Weapons_Use_Player_Damage` setting enabled.

Jam_Max_Chance float32 `0.1`

Decimal-to-percent chance for jamming to occur. This property requires `Can_Ever_Jam`.

Jam_Quality_Threshold float32 `0.4`

The maximum threshold for when jamming can occur. This value is a decimal-to-percent representation of the item's quality value. For example, a threshold of `0.4` allows jamming to start occuring at 40% item quality. This property requires `Can_Ever_Jam`.

Magazine uint16 `0`

Legacy ID of a magazine attachment that should be attached by default.

Magazine_Caliber_# uint16

Legacy ID of a caliber to check for magazine attachment compatibility. This property is used in conjunction with `Magazine_Calibers`, which determines how many instances of this property should be read by the game.

When this property is unset, it will default to `0`. When the `Magazine_Calibers` property is not greater than `0`, this property will default to the value of `Caliber`.

Magazine_Calibers int32

Set the length of the array containing the calibers for magazine attachment compatibility. This property is used in conjunction with the `Magazine_Caliber_#` property, and the value of `Magazine_Calibers` should be equal to the number of instances of `Magazine_Caliber_#`.

When this property is not greater than `0` – it will default to `1`, and the `Magazine_Caliber_#` property can no longer be customized.

This property is often used alongside `Attachment_Calibers`, but this is optional.

Magazine_Replacement_#_ID uint16 `0`

Legacy ID of a magazine attachment that should be used as an alternative default when certain condition(s) are met. This property is used in conjunction with `Magazine_Replacements`, which determines how many instances of this property should be read by the game.

Magazine_Replacement_#_Map string

This value should be the name of a map. When the weapon spawns on this map, this condition has been met. This property requires `Magazine_Replacement_#_ID`.

Magazine_Replacements int `0`

`Magazine_Replacements` and its related properties are used to add alternative magazine attachments that should be used as the weapon's default when certain condition(s) are met.

This value sets the length of the array containing any alternative default magazine attachments. This property is used in conjunction with the `Magazine_Replacement_#_ID` property, and the value of `Magazine_Replacements` should be equal to the number of instances of `Magazine_Replacement_#_ID`.

Muzzle GUID or uint16

GUID or legacy ID of the effect to play after shooting. This is emitted from the ranged weapon's "Barrel" GameObject.

Projectile_Explosion_Launch_Speed float32

Players caught within the area-of-effect explosion caused by a *physics projectile* weapon are launched at this speed. For example, this can be used to create velocity-related items like "rocket-jumping" mods. Defaults to `Player_Damage × 0.1`.

Projectile_Lifespan float32 `30`

Lifespan of *physics projectiles*, in seconds. After this much time elapses, the projectile despawns.

Projectile_Penetrate_Buildables flag

The area-of-effect explosions caused by *physics projectiles* penetrate through buildables when this flag is set.

Range_Rangefinder float32

Overrides the maximum distance displayed when using a "Rangefinder" tactical attachment on this weapon. For example, it may be useful to set this property when using `Action Rocket`, as explosive projectiles use `Range` to determine the explosion radius rather than the maximum range of the weapon. Defaults to the value of the `Range` property.

Recoil_Crouch float32 `0.85`

Multiplier on camera recoil while crouched.

Recoil_Max_X float32 `0`

Maximum horizontal camera recoil in degrees. This property is used in conjunction with `Recoil_Min_Y`.

Recoil_Max_Y float32 `0`

Maximum vertical camera recoil in degrees. This property is used in conjunction with `Recoil_Min_X`.

Recoil_Min_X float32 `0`

Minimum horizontal camera recoil in degrees. This property is used in conjunction with `Recoil_Max_X`.

Recoil_Min_Y float32 `0`

Minimum vertical camera recoil in degrees. This property is used in conjunction with `Recoil_Max_Y`.

Recoil_Prone float32 `0.7`

Multiplier on camera recoil while prone.

Recoil_Sprint float32 `1.25`

Multiplier on camera recoil while sprinting. This property is not relevant unless `Can_Aim_During_Sprint` has been set to `true`.

Recover_X float32 `0`

Multiplier on camera degrees to be counter-animated horizontally over the next 250 milliseconds.

Recover_Y float32 `0`

Multiplier on camera degrees to be counter-animated vertically over the next 250 milliseconds.

Reload_Time float32 `1`

Multiplier on time it takes to finish reloading the ranged weapon. This does not affect the actual animation speed, but the cooldown before the player can perform other actions (such as shooting) again. Values less than 1 have no effect.

Replace float32 `1`

Multiplier of the reload animation length before the magazine is respawned. This does not affect the actual animation speed, but the cooldown before the player can perform other actions (such as shooting) again. Values less than `0.01` have no effect.

Requires_NonZero_Attachment_Caliber bool `false`

If `true`, attachments must specify at least one non-zero (`0`) caliber ID to be compatible. For example, this can be used to make most vanilla attachments (like the Tactical Laser, Dot Sight, and Vertical Grip) incompatible with this weapon.

Safety flag

The weapon has a safety firing mode.

Scale_Aim_Animation_Speed bool `true`

When true, the length of the "Aim_Start" and "Aim_Stop" animations are scaled to match `Aim_In_Duration` (with modifiers).

Semi flag

The weapon has a semi-automatic firing mode.

Shake_Max_X float32 `0`

Maximum -axis model shake caused from firing the weapon.

Shake_Min_X float32 `0`

Minimum -axis model shake caused from firing the weapon.

Shake_Max_Y float32 `0`

Maximum -axis model shake caused from firing the weapon.

Shake_Min_Y float32 `0`

Minimum -axis model shake caused from firing the weapon.

Shake_Max_Z float32 `0`

Maximum -axis model shake caused from firing the weapon.

Shake_Min_Z float32 `0`

Minimum -axis model shake caused from firing the weapon.

Shell GUID or uint16

GUID or legacy ID of the effect to play after shooting, emitted from the ranged weapon's "Eject" GameObject. Defaults to `33` when using either `Action Pump` or `Action Break`; defaults to `1` when using any other `Action` key-value pair except for `Action Rail`; otherwise, defaults to `0`.

Should_Delete_Empty_Magazines bool

Overrides how empty magazines are handled by the action item mode. When set to `true`, empty magazine attachments are deleted when completely emptied. The default behavior depends on the configuration of the `Action` property.

Defaults to `true` when using one of the following `Action` enumerators: `Break`, `Pump`, `Rail`, `Rocket`, or `String`. Otherwise, defaults to `false`.

Sight uint16 `0`

Legacy ID of a sight attachment that should be attached by default. The `Hook_Sight` flag is not required to use this property.

Spread_Aim float32 `0`

Multiplier on the bullet spread while aiming down sights. This is multiplied by the `Spread_Angle_Degrees` value.

Spread_Angle_Degrees float32 `0`

Bullet angle of deviation away from the aiming direction. For example, `15` means the shot could hit up to 15 degrees away from the center of the crosshair, whereas `0` will always hit the center of the crosshair. All other spread values are multipliers for this.

Spread_Crouch float32 `0.85`

Multiplier on the bullet spread while crouched.

Spread_Hip float32

Deprecated since version 3.22.20.0: Use `Spread_Angle_Degrees` instead.

Maintained for backwards compatibility. Running the game with the `-ValidateAssets` *launch option* will log the equivalent `Spread_Angle_Degrees` value.

Spread_Prone float32 `0.7`

Multiplier on the bullet spread while prone.

Spread_Sprint float32 `1.25`

Multiplier on the bullet spread while sprinting.

Tactical uint16 `0`

Legacy ID of a tactical attachment that should be attached by default. The `Hook_Tactical` flag is not required to use this property.

Turret flag

This weapon should be treated as a vehicular turret. This flag affects the player's first-person viewmodel while the weapon is held.

Unjam_Chamber_Anim string `UnjamChamber`

Name of an animation clip to play when unjamming the weapon. This property requires `Can_Ever_Jam`.

Unplace float32 `0`

Multiplier of the reload animation length before the magazine is despawned. This does not affect the actual animation speed, but the cooldown before the player can perform other actions (such as shooting) again.

18.28.3 NPC Rewards

Gun assets can use quest rewards. For example, every time the ranged weapon is fired an item could be spawned in the player's inventory. Alternatively, shooting the ranged weapon may be required to complete a quest. For more information, refer to the *Rewards* documentation.

These rewards are prefixed with `Shoot_Quest_`. For example, `Shoot_Quest_Rewards 1`.

18.28.4 Understanding Projectile Systems

All ranged weapons utilize one of two projectile systems: the *ballistic projectile system*, or the *physics projectile system*. This is determined based on the *Action* the weapon has been configured to use, although most weapons use the ballistic projectile system.

Ballistic projectiles use a deterministic simulation. Their travel time, bullet drop, and other characteristics can be configured with properties such as *Ballistic_Travel* and *Bullet_Gravity_Multiplier*. When the ballistics game mechanic is disabled, these weapons function as hitscan instead.

Physics projectiles use Unity's physics simulation. Unlike ballistic projectiles, these are not deterministic. Additionally, physics projectiles cause area-of-effect explosions upon impact. The characteristics of physics projectiles can be configured with properties such as *Ballistic_Force* and *Projectile_Explosion_Launch_Speed*.

18.29 Hat Assets

Hats can be worn by players and zombies.

This inherits the *GearAsset* class.

18.29.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Hat`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

18.29.2 Hat Asset Properties

Hats have no unique asset properties. Refer to parent classes for additional properties.

18.30 Key Assets

Keys are intended to be used as a part of the Steam Economy, rather than as in-game content. As such, none of its unique properties can be properly utilized by modders.

This inherits the *ItemAsset* class.

18.30.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Key`)

ID *uint16*: Must be a unique identifier.

18.30.2 Key Asset Properties

Exchange_With_Target_Item *flag*: Adds UI elements for using this Steam Economy item on another Steam Economy item.

18.31 Library Assets

Libraries are placeable storage containers for experience points.

This inherits the *BarricadeAsset* class.

18.31.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Library`)

Useable *enum* (`Barricade`)

Build *enum* (`Library`)

ID *uint16*: Must be a unique identifier.

18.31.2 Library Asset Properties

Capacity *uint32*: Maximum amount of experience points that can be stored.

Tax *byte*: Percentage of the deposit that is taxed. Defaults to 0.

18.32 Magazine Assets

Magazine attachments are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

18.32.1 Game Data File

Magazine attachments inherit properties from the CaliberAsset class, which in turn inherits properties from the ItemAsset class. Properties that are required to be included are listed in the table below.

Class	Property Name	Required Value
ItemAsset	*GUID*	
ItemAsset	*ID*	
ItemAsset	*Type*	`Magazine`

Properties

Property Name	Type	Default Value
Animal_Damage	*float32*	`0`
Barricade_Damage	*float32*	`0`
Delete_Empty	*flag*	
Explosion	*GUID* or *uint16*	`0`
Explosion_Launch_Speed	*float32*	See description
Explosive	*flag*	
Impact	*GUID* or *uint16*	`0`
Object_Damage	*float32*	See description
Pellets	*uint8*	`1`
Player_Damage	*float32*	`0`
Projectile_Blast_Radius_Multiplier	*float32*	`1`
Projectile_Damage_Multiplier	*float32*	`1`
Projectile_Launch_Force_Multiplier	*float32*	`1`
Range	*float32*	`0`
Resource_Damage	*float32*	`0`
Should_Fill_After_Detach	*bool*	`false`
Spawn_Explosion_On_Dedicated_Server	*flag*	
Speed	*float32*	`1`
Structure_Damage	*float32*	`0`
Stuck	*uint8*	`0`
Tracer	*GUID* or *uint16*	`0`
Vehicle_Damage	*float32*	`0`
Zombie_Damage	*float32*	`0`

Property Descriptions

Animal_Damage float32 `0`

Damage dealt to animals caught within the area-of-effect explosion of a magazine attachment using the `Explosive` flag.

Barricade_Damage float32 `0`

Damage dealt to barricades caught within the area-of-effect explosion of a magazine attachment using the `Explosive` flag.

Delete_Empty flag

The magazine attachment should be deleted when it is fully depleted.

Explosion GUID or uint16 `0`

GUID or legacy ID of the effect that should be used for explosions caused by magazine attachment using the `Explosive` flag.

Explosion_Launch_Speed float32

Players caught within the area-of-effect explosion caused by projectiles when using the `Explosive` property are launched at this speed, in meters per second. Defaults to the resulting value of `Player_Damage * 0.1`.

Explosive flag

When this flag is included, the projectile fired from a ballistics projectile weapon will cause an area-of-effect explosion. This is typically used alongside the `Range` property.

Impact GUID or uint16 `0`

GUID or legacy ID of the effect that should be play on impact.

Object_Damage float32

Damage dealt to players caught within the area-of-effect explosion of a magazine attachment using the `Explosive` flag. Defaults to the value of the `Resource_Damage` property.

Pellets uint8 `1`

Number of bullet rays shot.

Player_Damage float32 `0`

Damage dealt to players caught within the area-of-effect explosion of a magazine attachment using the `Explosive` flag.

Projectile_Blast_Radius_Multiplier float32 `1`

Multiplier on the blast radius of the explosive projectiles fired from physics projectile weapons.

Projectile_Damage_Multiplier float32 `1`

Multiplier on the damage dealt by the explosive projectiles fired from physics projectile weapons.

Projectile_Launch_Force_Multiplier float32 `1`

Multiplier on the launch force applied to the explosive projectiles fired from physics projectile weapons.

Range float32 `0`

In meters, the radius of the area-of-effect explosion caused by a projectile when a magazine attachment is using the `Explosive` flag.

Resource_Damage float32 `0`

Damage dealt to resources caught within the area-of-effect explosion of a magazine attachment using the `Explosive` flag.

Should_Fill_After_Detach bool `false`

Ammunition should be fully refilled after the magazine attachment is detached from a ranged weapon.

Spawn_Explosion_On_Dedicated_Server flag

When using the `Explosion` property, spawn the explosion effect on the server.

Speed float32 `1`

Multiplier on reload speed.

Structure_Damage float32 `0`

Damage dealt to structures caught within the area-of-effect explosion of a magazine attachment using the `Explosive` flag.

Stuck uint8 `0`

The amount of quality that should be lost after the projectile hits something. When this value is greater than `0`, the item will have a visible quality value. This property is typically used with *ranged weapons* utilizing the `Action String` key-value pair, such as a crossbow.

Tracer GUID or uint16 `0`

GUID or legacy ID of the effect that should be used for bullet tracers.

Vehicle_Damage float32 `0`

Damage dealt to vehicles caught within the area-of-effect explosion of a magazine attachment using the `Explosive` flag.

Zombie_Damage float32 `0`

Damage dealt to zombies caught within the area-of-effect explosion of a magazine attachment using the `Explosive` flag.

18.33 Map Assets

Maps and compasses provide the player with additional UI information for as long as they are in the player's inventory. They can neither be held nor equipped.

This inherits the *ItemAsset* class.

18.33.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Map`, `Compass`)

ID *uint16*: Must be a unique identifier.

18.33.2 Map Asset Properties

Enables_Map *flag*: Provides access to a satellite map display.

Enables_Chart *flag*: Provides access to a chart map display.

Enables_Compass *flag*: Provides a compass HUD, and the ability to set visible waypoints on the map display.

18.34 Mask Assets

Masks can be worn by players and zombies.

This inherits the *GearAsset* class.

18.34.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Mask`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

18.34.2 Mask Asset Properties

Earpiece *flag*: Specified if mask allows for listening on communications by walkie-talkie.

18.35 Medical Assets

Medicine is irreversibly consumed by the player on use, and directly affect a player's stats such as health or immunity.

This inherits the *ConsumeableAsset* class.

18.35.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Medical`)

Useable *enum* (`Consumeable`)

ID *uint16*: Must be a unique identifier.

18.35.2 Medical Asset Properties

Medicine have no unique asset properties. Refer to parent classes for additional properties.

18.36 Melee Assets

Melee weapons can be used as a source of damage. Melee weapons always show quality.

This inherits the *WeaponAsset* class.

18.36.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Melee`)

Useable *enum* (`Melee`)

Slot *enum* (`Primary`, `Secondary`, `Tertiary`, `Any`)

ID *uint16*: Must be a unique identifier.

18.36.2 Melee Asset Properties

Alert_Radius *float*: The radius where zombies and animals should be alerted when attacking, measured in meters. Defaults to 8.

AttackAudioClip *Master Bundle Pointer*: AudioClip to play when attacking.

ImpactAudioDef *Master Bundle Pointer*: AudioClip or OneShotAudioDefinition to play upon impact.

Light *flag*: Provides a toggleable flashlight, and allows for using *PlayerSpotLightConfig* properties.

Repair *flag*: Repairs barricades, structures, and vehicles.

Repeated *flag*: The melee weapon's strong attack is disabled, and its weak attack will deal damage continuously.

Stamina *byte*: Amount of stamina depleted with each attack. Defaults to 0.

Strength *float*: Multiplier on the damage dealt by strong attacks.

Strong *float*: Multiplier for the strong attack animation length, for when to apply damage. Defaults to 0.33.

Weak *float*: Multiplier for the weak attack animation length, for when to apply damage. Defaults to 0.5.

18.37 Oil Pump Assets

Oil pumps are placeables capable of creating fuel. When powered, oil pumps generate fuel over time.

This inherits the *BarricadeAsset* class.

18.37.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Oil_Pump`)

Useable *enum* (`Barricade`)

Build *enum* (`Oil`)

ID *uint16*: Must be a unique identifier.

18.37.2 Oil Pump Asset Properties

Fuel_Capacity *uint16*: Maximum units of fuel that can be stored in the oil pump. Defaults to 0.

18.38 Optic Assets

Optics can modify a player's view.

This inherits the *ItemAsset* class.

18.38.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Optic`)

Useable *enum* (`Optic`)

ID *uint16*: Must be a unique identifier.

18.38.2 Sight Asset Properties

Zoom *float*: Multiplicative amount of zoom. Defaults to 1.

18.39 Pants Assets

Pants can be worn by players and zombies.

This inherits the *BagAsset* class.

18.39.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Pants`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

18.39.2 Pants Asset Properties

Pants have no unique asset properties. Refer to parent classes for additional properties.

18.40 Placeable Assets

Placeables are able to be placed by players.

This inherits the *ItemAsset* class.

18.40.1 Placeable Asset Properties

Item_Dropped_On_Destroy *Asset Pointer*: Spawn table for items dropped when destroyed.

Min_Items_Dropped_On_Destroy *int*: Minimum number of items to drop when destroyed. Defaults to 0.

Max_Items_Dropped_On_Destroy *int*: Maximum number of items to drop when destroyed. Defaults to 0.

SalvageItem *Asset Pointer*: Item added when picking up below 100% health. Defaults to a random item used in the placeable's blueprints.

18.41 Refill Assets

Refills (localized as “water canisters”) are useables able to siphon, store, and deposit water. Players can also drink from water canisters in order to restore their status bars. Water canisters have four potential states: empty, salty, dirty, or clean.

This inherits the *ItemAsset* class.

18.41.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Refill`)

Useable *enum* (`Refill`)

ID *uint16*: Must be a unique identifier.

18.41.2 Refill Asset Properties

ConsumeAudioClip *Master Bundle Pointer*: AudioClip to play when the consumeable is used.

Clean_Food *float*: Amount of food restored when drinking clean water.

Clean_Health *float*: Amount of health restored when drinking clean water.

Clean_Oxygen *float*: Amount of oxygen restored when drinking clean water.

Clean_Stamina *float*: Amount of stamina restored when drinking clean water.

Clean_Virus *float*: Amount of immunity depleted when drinking clean water.

Clean_Water *float*: Amount of water restored when drinking clean water.

Dirty_Food *float*: Amount of food restored when drinking dirty water. Defaults to the result of `Clean_Food * 0.6`.

Dirty_Health *float*: Amount of health restored when drinking dirty water. Defaults to the result of `Clean_Health * 0.6`.

Dirty_Oxygen *float*: Amount of oxygen restored when drinking dirty water. Defaults to the result of `Clean_Oxygen * 0.6`.

Dirty_Stamina *float*: Amount of stamina restored when drinking dirty water. Defaults to the result of `Clean_Stamina * 0.6`.

Dirty_Virus *float*: Amount of immunity depleted when drinking dirty water. Defaults to the result of `Clean_Virus * -0.399999976`.

Dirty_Water *float*: Amount of water restored when drinking dirty water. Defaults to the result of `Clean_Water * 0.6`.

Salty_Food *float*: Amount of food restored when drinking salty water. Defaults to the result of `Clean_Food * 0.25`.

Salty_Health *float*: Amount of health restored when drinking salty water. Defaults to the result of `Clean_Health * 0.25`.

Salty_Oxygen *float*: Amount of oxygen restored when drinking salty water. Defaults to the result of `Clean_Oxygen * 0.25`.

Salty_Stamina *float*: Amount of stamina restored when drinking salty water. Defaults to the result of `Clean_Stamina * 0.25`.

Salty_Virus *float*: Amount of immunity depleted when drinking salty water. Defaults to the result of `Clean_Virus * -0.75`.

Salty_Water *float*: Amount of water restored when drinking salty water. Defaults to the result of `Clean_Water * 0.25`.

Deprecated since version 3.20.9.0: **Water** *byte*: Deprecated in favor of `Clean_Water`. When this property is used, its value is assigned to `Clean_Water` instead.

18.42 Sentry Assets

Sentries (localized as "robotic turrets") are placeables that can automatically detect, track, and attack target under certain conditions. Storing a ranged weapon inside a sentry allows it to use that weapon when attacking target.

This inherits the *StorageAsset* class.

18.42.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Sentry`)

Useable *enum* (`Barricade`)

Build *enum* (`Sentry`, `Sentry_Freeform`)

ID *uint16*: Must be a unique identifier.

18.42.2 Sentry Asset Properties

Detection_Radius *float*: Radius for initially detecting targets, in meters. Defaults to 48.

Mode *enum* (`Friendly`, `Neutral`, `Hostile`): The sentry's "Mode" determines what the sentry considers a valid target. Defaults to `Mode Neutral`.

Infinite_Ammo *bool*: Whether or not the magazine attachments in the stored ranged weapon should be depleted during use. If true, the ranged weapon has infinite ammo and any attached magazine attachment will not be depleted during use. Defaults to false.

Infinite_Quality *bool*: Whether or not the stored ranged weapon should degrade during use. If true, the ranged weapon's quality will not degrade during use. Defaults to false.

Requires_Power *bool*: Whether or not the sentry requires power from a generator. If true, the sentry must be powered in order for it to detect, track, and attack targets. Defaults to true.

Target_Acquired_Effect *Asset Pointer*: The audio effect played when a target is detected. Defaults to ab5f0056b54545c8a051159659da8bea.

Target_Lost_Effect *Asset Pointer*: The audio effect played when a target is no longer detected. Defaults to 288b98b718084699ba3653c592e57803.

Target_Loss_Radius *float*: Radius for continuing to track targets after they have left the initial detection radius, in meters. Defaults to `Detection_Radius * 1.2f` (i.e., 20% higher than `Detection_Radius`).

18.43 Shirt Assets

Shirts can be worn by players and zombies.

This inherits the *BagAsset* class.

18.43.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Shirt`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

18.43.2 Shirt Asset Properties

Ignore_Hand *flag*: Specified if shirt should ignore a player's left-handed setting.

18.43.3 Body Mesh Replacements

For the full documentation, refer to the *Character Mesh Replacement* documentation.

Has_1P_Character_Mesh_Override *bool*: A prefab named "Character_Mesh_1P_Override_0" should be loaded. Defaults to false.

Character_Mesh_3P_Override_LODs *uint16*: Number of prefabs to load for each LOD index. Defaults to 0.

Has_Character_Material_Override *bool*: A material named "Character_Material_Override" should be loaded to replace the first-person and third-person mesh materials. Defaults to false.

18.44 Sight Assets

Sight attachments are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

18.44.1 Game Data File

Sight attachments inherit properties from the CaliberAsset class, which in turn inherits properties from the ItemAsset class. Properties that are required to be included are listed in the table below.

Class	Property Name	Required Value
ItemAsset	*GUID*	
ItemAsset	*ID*	
ItemAsset	*Type*	`Sight`

Properties

Property Name	Type	Default Value
DistanceMarkers	*list of DistanceMarker*	
Holographic	*flag*	
Nightvision_Color	*color*	See description
Nightvision_Fog_Intensity	*float32*	See description
Offset_Scope_Overlay_By_One_Texel	*bool*	`false`
Vision	*ELightingVision*	`None`
Zoom	*float32*	`1`
ThirdPerson_Zoom	*float32*	`1.25`
Zoom_Using_Eyes	*bool*	`false`

DistanceMarker Dictionary

Property Name	Type	Default Value
Distance	*float32*	`0`
LineOffset	*float32*	`0`
LineWidth	*float32*	`0.05`
Side	*ESide*	`Right`
HasLabel	*bool*	`true`
Color	*color*	`black`

ESide Enumeration

Named Value	Description
`Left`	Marking extends to the left from the center.
`Right`	Marking extends to the right from the center.

Property Descriptions

DistanceMarkers list of DistanceMarker

This property is a list of *DistanceMarker dictionaries*. It can be used to add visible (and accurate) distance markers to the scope that account for the weapon's bullet drop.

Holographic flag

This sight should be holographic.

Nightvision_Color color

Override the default nightvision color. To configure this property, the `Vision` property must be set to `Military`. This property supports using legacy color parsing. When not overridden, the default nightivision color will depend on the value of the *Vision* property.

Nightvision_Fog_Intensity float32

Configure the intensity of fog while nightvision is active. When this property has not been configured, the default fog intensity will depend on the value of the *Vision* property.

Offset_Scope_Overlay_By_One_Texel bool `false`

If `true`, the 2D scope texture will be scaled up slightly to center the pixel that would otherwise be left of center. For example, when enabled with a 512×512 texture the pixel at 255×255 will be centered on the display.

Vision ELightingVision `None`

Set a unique lighting vision effect to use. The value of this property may effect the default values of other properties. The `Headlamp` enumerator is not supported by this property.

Zoom float32 `1`

Multiplicative amount of zoom. This value must be equal to or greater than 1.

ThirdPerson_Zoom float32 `1.25`

Zoom factor while in the third-person perspective. This value must be equal to or greater than 1.

Zoom_Using_Eyes bool `false`

Whether the main camera field of view should zoom without a scope overlay.

DistanceMarker Dictionary Descriptions

Distance float32 `0`

Meters between the player and a hypothethical target.

LineOffset float32 `0`

Distance between center line and start of horizontal line marker.

Display-related properties like `LineOffset` are a percentage (represented as a decimal value from 0 to 1). For example, `0.25` would be 25%.

LineWidth float32 `0.05`

Length of horizontal line marker.

Display-related properties like `LineWidth` are a percentage (represented as a decimal value from 0 to 1). For example, `0.25` would be 25%.

Side ESide `Right`

Direction the horizontal line and text expand in.

HasLabel bool `true`

If true, a label with `Distance` text is shown next to the horizontal line marker.

Color color `black`

Override the color of the horizontal line and text.

18.45 Storage Assets

Storages (localized as "item storages") are placeables used to store items.

This inherits the *BarricadeAsset* class.

18.45.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Storage`)

Useable *enum* (`Barricade`)

Build *enum* (`Storage`, `Storage_Wall`)

ID *uint16*: Must be a unique identifier.

18.45.2 Storage Asset Properties

Display *flag*: If specified, the first item in the storage will be visibly displayed.

Should_Close_When_Outside_Range *bool*: Whether or not the storage should automatically close when the player is outside of the interaction range. Defaults to false.

Storage_X *byte*: Number of columns (horizontal storage space). Defaults to 0.

Storage_Y *byte*: Number of rows (vertical storage space). Defaults to 0.

18.46 Structure Assets

Structures can be placed by players. Some structure pieces require another structure piece in order to be placed.

This inherits the *PlaceableAsset* class.

18.46.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Structure`): When intending to use a child class, refer to that class's documentation instead for the proper enumerator to use.

Useable *enum* (`Structure`)

Construct *enum* (`Floor`, `Floor_Poly`, `Pillar`, `Post`, `Rampart`, `Roof`, `Roof_Poly`, `Wall`): Determines how this structure can be placed, and how other structure pieces can snap to it.

ID *uint16*: Must be a unique identifier.

InventoryAudio *Master Bundle Pointer*: See *ItemAsset* for full documentation. Defaults to `Sounds/Inventory/SmallMetal.asset` if the name contains the word "Metal", to `Sounds/Inventory/LightMetalEquipment.asset` if either `Size_X` or `Size_Y` value is equal to 1, to `Sounds/Inventory/MediumMetalEquipment.asset` if either `Size_X` or `Size_Y` value is less than or equal to 2, or `Sounds/Inventory/HeavyMetalEquipment.asset` if none of the criteria is met.

18.46.2 Structure Asset Properties

Armor_Tier *enum* (`Low`, `High`): Armor is a multiplier on damage received. A structure's armor tier can either be low-tier or high-tier. By default, structures with low-tier armor take 100% of the damage they receive, while structures with high-tier armor take 50% of the damage they receive. These multipliers can be configured in the gameplay config. Defaults to low-tier, except when the structure's name contains the word "Metal" or "Brick".

Can_Be_Damaged *bool*: If true, this structure can be damaged. Defaults to true.

Eligible_For_Pooling *bool*: If true, this structure is eligible for object pooling. Some structures may not reset properly when pooling is enabled. Defaults to true.

Explosion *GUID* or *uint16*: GUID or legacy ID of *EffectAsset* to play when destroyed.

Foliage_Cut_Radius *float*: In meters, the radius around the structure where foliage is removed. Defaults to 6.

Has_Clip_Prefab *bool*: Whether or not the structure has a Clip.prefab. If the structure should use the same prefab on the server as on the client, set to false. For example, most official content uses `Has_Clip_Prefab false`. Defaults to true.

Health *uint16*: Total health value. Defaults to 0.

PlacementAudioClip *Master Bundle Pointer*: AudioClip to play when the structure is placed.

PlacementPreviewPrefab *Master Bundle Pointer*: Overrides the placement preview model spawned when this item is held.

Proof_Explosion *flag*: Immune to area-of-effect explosive damage.

Range *float*: In meters, the maximum distance away the structure can be placed from the player.

Salvage_Duration_Multiplier *float*: Multiplier on how long it takes to salvage this structure. Setting this to a larger number will cause salvaging to take longer. Defaults to 1.

Terrain_Test_Height *float*: Length of the raycast downward from the pivot to check if the floor is above terrain. This is the maximum distance a floor can be placed above terrain, in meters. Defaults to 10.

Unpickupable *flag*: Disables the ability to pick up a placed structure.

Unrepairable *flag*: Cannot be repaired by a *MeleeAsset* with the `Repair` flag. For example, the Blowtorch would not be able to repair this structure.

Unsalvageable *flag*: Salvaging a damaged structure yields no partial resources.

Unsaveable *flag*: This structure is excluded from being saved.

Vulnerable *flag*: The structure can be damaged by lower-power weapons that do not have the `Invulnerable` flag.

Requires_Pillars *bool*: Whether or not a valid wall placement requires pillars. If true, two pillars are required for a valid placement. Defaults to true.

18.47 Supply Assets

Crafting supplies are items primarily intended to be used as ingredients in crafting blueprints. They can neither be held nor equipped.

This inherits the *ItemAsset* class.

18.47.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Supply`)

ID *uint16*: Must be a unique identifier.

18.47.2 Supply Asset Properties

Crafting supplies have no unique asset properties. Refer to *item asset documentation* for additional properties.

18.48 Tactical Assets

Tactical attachments are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

18.48.1 Game Data File

Tactical attachments inherit properties from the CaliberAsset class, which in turn inherits properties from the ItemAsset class. Properties that are required to be included are listed in the table below.

Class	Property Name	Required Value
ItemAsset	*GUID*	
ItemAsset	*ID*	
ItemAsset	*Type*	`Tactical`

Properties

Property Name	Type	Default Value
Laser	*flag*	
Laser_Color	*color*	`#FF0000`
Light	*flag*	
Melee	*flag*	
Rangefinder	*flag*	

Property Descriptions

Laser flag

Provides a toggleable laser.

Laser_Color color `#FF0000`

Override the default red color with the specified value. This property supports using legacy color parsing.

Light flag

Provides a toggleable flashlight, and allows for using *PlayerSpotLightConfig* properties.

Melee flag

Provides the ability to perform a melee attack. This attack does 40 damage, and is not configurable.

Rangefinder flag

Provides a toggleable rangefinder.

18.49 Tank Assets

Tanks (localized as "liquid storages") are placeables used to store water or fuel. Players can siphon from, or deposit into, a liquid storage with certain items. Fuel tanks require a *fuel canister*, while water tanks require a *water canister*.

This inherits the *BarricadeAsset* class.

18.49.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Tank`)

Useable *enum* (`Barricade`)

Build *enum* (`Tank`)

ID *uint16*: Must be a unique identifier.

18.49.2 Tank Asset Properties

Resource *uint16*: Maximum units of liquid that can be stored in the tank. One unit of water is the equivalent of one usage of a water canister. Defaults to 0.

Source *enum* (`Fuel`, `None`, `Water`): Type of liquid that can be stored in the tank.

18.50 Throwable Assets

Throwables can be thrown by players. Throwables cannot be used in any safezones that disallow weapons.

This inherits the *WeaponAsset* class.

18.50.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Throwable`)

Useable *enum* (`Throwable`)

ID *uint16*: Must be a unique identifier.

18.50.2 Throwable Asset Properties

Boost_Throw_Force_Multiplier *float*: A multiplier on the amount of throwing force when the player has the "Olympic" random boost. Defaults to 1.4.

Explode_On_Impact *flag*: Specified if the throwable should immediately detonate upon impact. Robotic turrets using `Mode Friendly` will target players holding a throwable that has this flag.

Explosion *uint16* or *GUID*: ID or GUID of explosion effect to play upon detonation.

Explosion_Launch_Speed *float*: Velocity at which players are launched by area-of-effect explosions. Defaults to `Player_Damage × 0.1`.

Explosive *flag*: Specified if the throwable should have an area-of-effect explosion. Robotic turrets using `Mode Friendly` will target players holding a throwable that has this flag.

Flash *flag*: Specified if the throwable should cause a flashbang effect for players caught within the area-of-effect. Robotic turrets using `Mode Friendly` will target players holding a throwable that has this flag.

Fuse_Length *float*: A timer, in seconds, for the fuse length. Defaults to 180 seconds. If the throwable has the `Explosive` flag or the `Flash` flag, then it defaults to 2.5 seconds instead.

Sticky *flag*: Specified if the throwable should stick to the environment, barricades, and structures.

Strong_Throw_Force *float*: The amount of force throwables are thrown with when performing a strong throw, measured in Newtons. Defaults to 1100.

Weak_Throw_Force *float*: The amount of force throwables are thrown with when performing a weak throw, measured in Newtons. Defaults to 600.

18.51 Tire Assets

Tires (localized as "tools") are useables that allow for adding and removing tires from vehicles.

This inherits the *VehicleRepairToolAsset* class.

18.51.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Tire`)

Useable *enum* (`Tire`)

ID *uint16*: Must be a unique identifier.

18.51.2 Tire Asset Properties

Mode *enum* (`Add`, `Remove`): How the usable should interact with tires. `Mode Add` will consume the item to add a tire to the vehicle. `Mode Remove` allows the usable to remove tires, adding the corresponding item to the player's inventory.

18.52 Tool Assets

Tools are a type of useable. The specific function of a tool significantly depends on the `Useable` property.

This inherits the *ItemAsset* class.

18.52.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Tool`): When intending to use a child class, refer to that class's documentation instead for the proper enumerator to use.

Useable *enum* (`Carjack`, `Carlockpick`, `Housing_Planner`, `Walkie_Talkie`): When using the `Carjack` enumerator, the tool can be used on vehicles to launch them upwards into the air. When using `Carlockpick`, the tool can be used once on any locked vehicle in order to forcefully unlock it. When using `Housing_Planner`, the tool can be used to quickly access *structure pieces* from the player's own inventory. When using `Walkie_Talkie`, the tool can be used to have long-distance voice chat communications with other players.

ID *uint16*: Must be a unique identifier.

18.52.2 Tool Asset Properties

Tools have no unique asset properties. Instead, refer to `Useable` for relevant configuration options. Refer to parent classes for additional properties.

18.53 Trap Assets

Traps are placeable damage sources.

This inherits the *BarricadeAsset* class.

18.53.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Trap`)

Useable *enum* (`Barricade`)

Build *enum* (`Spike`, `Wire`)

ID *uint16*: Must be a unique identifier.

18.53.2 Trap Asset Properties

Animal_Damage *float*: Damage dealt to animals caught within the area-of-effect explosion.

Barricade_Damage *float*: Damage dealt to barricades caught within the area-of-effect explosion.

Broken *flag*: Players who trigger the trap will be inflicted with the Broken Bones status effect.

Damage_Tires *flag*: This trap can pop the tires of vehicles that drive over it.

Explosion_Launch_Speed *float*: Launch speed of players caught within the area-of-effect explosion, in meters per second. Defaults to the value of `Player_Damage * 0.1`.

Explosion2 *uint16* or *GUID*: ID or GUID of effect to play upon detonation.

Explosive *flag*: Specified if the trap should have an area-of-effect explosion when triggered.

Object_Damage *float*: Damage dealt to objects caught within the area-of-effect explosion. Defaults to the value of `Resource_Damage`.

Player_Damage *float*: Damage dealt to players caught within the area-of-effect explosion.

Range2 *float*: In meters, the radius of the damaging, area-of-effect explosion.

Resource_Damage *float*: Damage dealt to resources caught within the area-of-effect explosion.

Structure_Damage *float*: Damage dealt to structures caught within the area-of-effect explosion.

Trap_Cooldown *float*: In seconds, the time until trap is active again.

Trap_Setup_Delay *float*: In seconds, delay before a trap becomes active after being placed. Defaults to 0.25 seconds.

Vehicle_Damage *float*: Damage dealt to vehicles caught within the area-of-effect explosion.

18.54 Vehicle Repair Tool Assets

Vehicle repair tools (localized as “tools”) are useables for replacing vehicle batteries.

This inherits the *ToolAsset* class.

18.54.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Vehicle_Repair_Tool`)

Useable *enum* (`Battery_Vehicle`)

ID *uint16*: Must be a unique identifier.

18.54.2 Vehicle Repair Tool Asset Properties

Vehicle repair tools have no unique asset properties. Refer to parent classes for additional properties.

18.55 Vest Assets

Vests can be worn by players and zombies.

This inherits the *BagAsset* class.

18.55.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Vest`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

18.55.2 Vest Asset Properties

Vests have no unique asset properties. Refer to parent classes for additional properties.

18.56 Water Assets

Drinks are irreversibly consumed by the player on use, and directly affect a player's stats such as water or stamina.

This inherits the *ConsumeableAsset* class.

18.56.1 Item Asset Properties

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Water`)

Useable *enum* (`Consumeable`)

ID *uint16*: Must be a unique identifier.

18.56.2 Water Asset Properties

Drinks have no unique asset properties. Refer to parent classes for additional properties.

18.57 Weapon Assets

Weapon assets function as a source of damage. The functional implementation of properties may differ slightly between assets.

This inherits the *ItemAsset* class.

18.57.1 Weapon Asset Properties

Allow_Flesh_Fx *bool*: Boolean for if special effects should occur when damaging flesh. Defaults to true.

Durability *float32*: Probability of quality loss upon the weapon being used, as a decimal.

Range *float32*: The maximum distance in meters before damage is no longer possible. For ballistic projectile ranged weapons, this is the maximum distance a projectile may travel. For melee weapons, this is the maximum swinging distance. For physics projectile ranged weapons, this is the radius of the explosion.

Wear *uint8*: Increment to degrade quality by. Defaults to 1.

Player Damage

Bypass_Allowed_To_Damage_Player *bool*: Boolean for if the weapon should bypass the requirements for being allowed to damage other players. Typically, a weapon cannot damage another player if the server is set to PvE, or if the target player is a part of the same group and friendly fire is disabled. Defaults to false.

Player_Damage *float32*: Amount of damage that should be dealt to player entities, prior to modifiers such as limb multipliers.

Player_Leg_Multiplier *float32*: Multiplier on damage targeted against a player's legs. Limb multipliers are not utilized by explosive weapons.

Player_Arm_Multiplier *float32*: Multiplier on damage targeted against a player's arms. Limb multipliers are not utilized by explosive weapons.

Player_Spine_Multiplier *float32*: Multiplier on damage targeted against a player's torso. Limb multipliers are not utilized by explosive weapons.

Player_Skull_Multiplier *float32*: Multiplier on damage targeted against a player's head. Limb multipliers are not utilized by explosive weapons.

Player_Damage_Bleeding *enum* (`Always`, `Default`, `Heal`, `Never`): Determines the effect the weapon has in relation to the "Bleeding" status effect. When using "Always", the Bleeding status effect will always be applied on hit. When using "Default", the Bleeding status effect will only be applied if the necessary damage threshold is met. When using "Heal", anyone hit by the weapon will have their Bleeding status effect removed. When using "Never", the Bleeding status effect is never applied by this weapon. Defaults to "Default" enumerator.

Player_Damage_Bones *enum* (`Always`, `Heal`, `None`): Determines the effect the weapon has in relation to the "Broken Bones" status effect. When using "Always", the Broken Bones status effect will always be applied on hit. When using "Heal", anyone hit by the weapon will have their Broken Bones status effect removed. When using "Never", the Broken Bones status effect is never applied by this weapon. Defaults to the "None" enumerator.

Player_Damage_Food *float32*: Amount of degradation dealt to a targeted player's food. Positive values are beneficial (increasing food level), and negative values are detrimental (decreasing food level). Negative values are blocked in the same situations damage is blocked (e.g., in safezones or shortly after respawns).

Player_Damage_Water *float32*: Amount of degradation dealt to a targeted player's water. Positive values are beneficial (increasing water level), and negative values are detrimental (decreasing water level). Negative values are blocked in the same situations damage is blocked (e.g., in safezones or shortly after respawns).

Player_Damage_Virus *float32*: Amount of degradation dealt to a targeted player's immunity. Positive values are beneficial (increasing immunity level), and negative values are detrimental (decreasing immunity level). Negative values are blocked in the same situations damage is blocked (e.g., in safezones or shortly after respawns).

Player_Damage_Hallucination *float32*: Length of hallucinations inflicted onto a targeted player, in seconds. Positive values are detrimental (increasing hallucination duration), and negative values are beneficial (decreasing hallucination duration). Positive values are blocked in the same situations damage is blocked (e.g., in safezones or shortly after respawns).

Zombie Damage

Zombie_Damage *float32*: Amount of damage that should be dealt to zombie entities, prior to modifiers such as limb multipliers.

Zombie_Leg_Multiplier *float32*: Multiplier on damage targeted against a zombie's legs. Limb multipliers are not utilized by explosive weapons.

Zombie_Arm_Multiplier *float32*: Multiplier on damage targeted against a zombie's arms. Limb multipliers are not utilized by explosive weapons.

Zombie_Spine_Multiplier *float32*: Multiplier on damage targeted against a zombie's torso. Limb multipliers are not utilized by explosive weapons.

Zombie_Skull_Multiplier *float32*: Multiplier on damage targeted against a zombie's head. Limb multipliers are not utilized by explosive weapons.

Stun_Zombie_Always *flag*: Specified if a zombie should always be stunned when targeted by the weapon.

Stun_Zombie_Never *flag*: Specified if a zombie should never be stunned when targeted by the weapon.

Animal Damage

Animal_Damage *float32*: Amount of damage that should be dealt to animal entities, prior to modifiers such as limb multipliers.

Animal_Leg_Multiplier *float32*: Multiplier on damage targeted against a animal's limbs. Limb multipliers are not utilized by explosive weapons.

Animal_Spine_Multiplier *float32*: Multiplier on damage targeted against a animal's torso. Limb multipliers are not utilized by explosive weapons.

Animal_Skull_Multiplier *float32*: Multiplier on damage targeted against a animal's head. Limb multipliers are not utilized by explosive weapons.

Construct Damage

BladeID *uint8*: Weapon can damage any resources or objects that have a matching BladeID. Deprecated in favor of BladeIDs and BladeID_#.

BladeIDs *int32*: Total number of unique BladeID_# values.

BladeID_# *uint8*: Weapon can damage any resources or objects that have a matching BladeID_# value.

Barricade_Damage *float32*: Amount of damage that should be dealt to barricades, prior to modifiers.

Structure_Damage *float32*: Amount of damage that should be dealt to structures, prior to modifiers.

Vehicle_Damage *float32*: Amount of damage that should be dealt to vehicles, prior to modifiers.

Resource_Damage *float32*: Amount of damage that should be dealt to resources, prior to modifiers.

Object_Damage *float32*: Amount of damage that should be dealt to objects, prior to modifiers. Defaults to the value used by Resource_Damage.

Invulnerable *flag*: Specified if damage should affect objects, structures, barricades, and vehicles that are considered invulnerable to low-power weaponry. Not applicable to explosive weapons, which will always ignore invulnerability.

CHAPTER

NINETEEN

LAYERS

Upfront: obviously Unturned makes poor use of Unity's Layers. This document exists as much for my personal reference as yours. My only defense is that these layers are entrenched from the earliest versions back in 2013, when I was 15 or 16.

19.1 Overview

Built-in Layers

- **0 Default**
- **1 TransparentFX**
- **2 Ignore Raycast**
- **4 Water**: ocean and water tiles.
- **5 UI**: menus with *uGUI glazier* as well as plugin custom menus.

User Layers

- **8 Logic**: Clickable overlays like the position, rotation and scale handles. Editor debug visuals that can be seen through walls are on this layer.
- **9 Player**: Character capsule (not body hitboxes). Exists for all players server-side, but only the local player client-side.
- **10 Enemy**: Player body hitboxes.
- **11 Viewmodel**: Local first-person arms and weapon.
- **12 Debris**: Typically small simulated objects like ragdolls, grenades, falling tree trunks, destroyed structures, and fragmented objects.
- **13 Item**: Dropped interactable items.
- **14 Resource**: Trees and boulders. Barricades attached to vehicles are moved to this layer.
- **15 Large**: Large props placed in the level editor.
- **16 Medium**: Medium props placed in the level editor.
- **17 Small**: Small props without collision placed in the level editor.
- **18 Sky**: Distant effects without collision like the clouds and stars.
- **19 Environment**: Roads, grass and pebbles.
- **20 Ground**: Landscape / terrain.

- **21 Clip**: Invisible collision.
- **22 Navmesh**: Invisible zombie-only collision. Navmesh graphs are generated from this collision, but the collision is also loaded on the server to help push zombies around.
- **23 Entity**: Zombie and animal body hitboxes.
- **24 Agent**: Zombie and animal character capsules (not body hitboxes).
- **25 Ladder**: Invisible climbable trigger.
- **26 Vehicle**: All vehicle colliders.
- **27 Barricade**: Barricade item placed in the world. Barricades attached to vehicles are moved to the Resource layer.
- **28 Structure**: Structure item placed in the world.
- **29 Tire**: Wheel colliders. Allows wheels to mask what they collide with.
- **30 Trap**: Typically trigger colliders including rocket launcher projectiles and kill volumes.
- **31 Ground2**: No longer used after old maps were converted to terrain tiles. Previously this was for out-of-bounds terrain. Reserved for future use.

19.2 Layer Collision Matrix

Note that these comments do **NOT** apply to collision queries like raycasts, spherecasts, etc.

No physics collision:

- **Default**
- **TransparentFX**
- **Ignore Raycast**
- **Water**
- **UI**
- **Logic**
- **Enemy**
- **Viewmodel**
- **Small**
- **Sky**
- **Environment**
- **Ground**
- **Entity**
- **Ladder**
- **Ground2**

Has physics collision:

- **Player**: Character controller layer is used by Unity as the underlying query mask.
- **Debris**

- **Item**
- **Resource**
- **Large**
- **Medium**
- **Environment**
- **Ground**
- **Clip**: Collides with Player and Vehicle for its original purpose. Makeshift vehicles have invisible colliders on this layer to expand their simulation size without affecting barricade placement, so Clip also collides with some of the same layers as Vehicle.
- **Navmesh**
- **Agent**: Character controller layer is used by Unity as the underlying query mask.
- **Vehicle**
- **Barricade**
- **Structure**
- **Tire**: Wheel collider layer is used by Unity as the underlying query mask.
- **Trap**

CHAPTER

TWENTY

LEVEL ASSETS

Each map can be associated with a **Level Asset**. These assets contain gameplay information not necessary for the main menus. Refer to *Level Config* for information on linking a level asset to a map.

For examples check the `Assets/Levels` directory.

Type *string*: `SDG.Unturned.LevelAsset`

Dropship *Master Bundle Pointer*: Overrides the model seen flying over the map when a care package is dropped.

Airdrop *Asset Pointer*: Asset pointer to an *Airdrop Asset*. Overrides the falling care package model.

Crafting_Blacklists array of *Asset Pointers*: Asset pointers to *Crafting Blacklist(s)*. Prevents specific items or blueprints from being used while crafting in the level.

Min_Stealth_Radius *float*: Player stealth skill level cannot reduce minimum detection distance below this value.

Weather_Types *array*: Determines which weather can occur naturally. Refer to schedulable weather properties. If weather is using legacy weather the default rain and snow will be included.

Perpetual_Weather_Asset *Asset Pointer*: Asset pointer to a *Weather Asset*. Overrides weather scheduling.

Global_Weather_Mask *u32 Mask*: Fallback weather mask while player is not inside an ambience volume. Defaults to 0xFFFFFFFF.

Skills *array*: Overrides skill default and max levels. Refer to skill rule properties.

Enable_Admin_Faster_Salvage_Duration *bool*: By default, players in singleplayer and admins in multiplayer have a faster salvage time.

Has_Clouds *bool*: Disables clouds in skybox when false. Defaults to true.

Loading_Screen_Music *array*: Randomly selected. Refer to music properties.

Should_Animate_Background_Image *bool*: If true, the background image moves left/right with loading progress. Defaults to false because maps have important information on the loading screen.

20.1 Schedulable Weather Properties

Asset *Asset Pointer*: Points to a *Weather Asset*.

Min_Frequency *float*: When chosen to be the next scheduled weather event, minimum number of in-game days before it will start.

Max_Frequency *float*: When chosen to be the next scheduled weather event, maximum number of in-game days before it will start.

Min_Duration *float*: Minimum number of in-game days before the weather event will end.

Max_Duration *float*: Maximum number of in-game days before the weather event will end.

20.2 Skill Rule Properties

Id *string*: Name of skill, for example Sharpshooter.

Default_Level *int*: Skill level when player spawns. Note server config Spawn_With_Max_Skills takes priority.

Max_Unlockable_Level *int*: Maximum skill level attainable through gameplay. Higher levels are hidden in the skills menu.

Cost_Multiplier *float*: multiplier for XP upgrade cost.

20.3 Music Properties

Loop *Master Bundle Pointer*: looping audio clip played until loading finishes.

Outro *Master Bundle Pointer*: audio clip played once loading finishes.

CHAPTER

TWENTYONE

MATERIAL PALETTE ASSETS

The `MaterialPaletteAsset` type allows an object to have multiple potential materials that it can use. A random material from the material palette is chosen every time the object is spawned in the level editor. In the level editor, material palettes can also be manually assigned to a selected object.

21.1 Metadata

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *string*: `SDG.Unturned.MaterialPaletteAsset`

21.2 Material Palette Properties

Materials *array* of *Master Bundle Pointer dictionaries*: Each dictionary in the list should point to a material bundled in Unity.

```
"Asset"
{
	"Materials"
	[
		{
			"Name" "core.masterbundle"
			"Path" "Objects/Material_Palettes/House/House_00.mat"
		}
		{
			"Name" "core.masterbundle"
			"Path" "Objects/Material_Palettes/House/House_01.mat"
		}
	]
}
```

CHAPTER TWENTYTWO

MOD HOOKS

22.1 Overview

Script Components can be added to Game Objects in Unity and exported in Asset Bundles *IF* they match a script in the base game code. These intentionally exportable scripts are referred to as **Mod Hooks**. They can be imported into a Unity project from the Project.unitypackage, and added to game objects inside the Unturned components menu. Each script makes several Events available which can drive other component properties like visibility or play an animation.

Each script documents its purpose and members within its *.cs file.

Originally proposed and coined by VitaxaRusModding in this GitHub issue: Link

22.2 Event Listeners

22.2.1 Activation Event Hook

Events when a component or game object are enabled and disabled. Useful for extending toggleable actions in the base game.

22.2.2 Binary Random Component

When triggered will invoke one of two events depending on percentage probability. For example with a probability of 0.05 the OnTrue event will be invoked 5% of the time, and OnFalse will be invoked the remaining 95% of times.

22.2.3 Collision Event Hook

Events for player overlaps with a trigger collider. Primarily useful for server-side objects as collisions are not triggered by other players client-side, but this limitation may be resolved in the future.

22.2.4 Destroy Event Hook

Event when a component or game object is removed from the scene.

22.2.5 Interactable Object Binary State Event Hook (IOBS)

(IOBS for short) are any prop placed from the level editor which can have F pressed on them to open, close, turn on/off, etc. This hook can be added to any GameObject within an IOBS to trigger events during state changes, and even control the IOBS from client and server side.

22.2.6 Interactable Object Quest Event Hook

This hook can be added to any GameObject within a Dropper, Note, or Quest Interactable Object. Its event is triggered when the corresponding interactable is successfully used. Note that the event is only triggered on the authority side (i.e., the server or singleplayer) and not on the client.

22.2.7 NPC Global Event Hook

Event triggered when corresponding *NPC Event reward* type is triggered. For example, when any NPC Event with ID "Fireworks" is broadcast all of the components with event ID "Fireworks" will have their corresponding Unity event triggered as well, in this case perhaps to spawn a fireworks effect.

22.2.8 Text Chat Event Hook

Event when a text chat message passes certain filters such as channel, within a radius, and containing a secret phrase. Only fired on the server.

22.2.9 Timer Event Hook

Allows events to set or cancel a timer, and triggers an event when the timer expires.

22.2.10 Useable Gun Event Hook

Events for EquipableItem prefab. Supersedes VehicleTurretEventHook. These events are fired on server and client.

22.2.11 Vehicle Event Hook

Events for driver entering and exiting the vehicle. These events are fired on server and client.

22.2.12 Vehicle Turret Event Hook

Events for Turret_# GameObjects in the vehicle when guns are used. These events are fired on server and client.

22.2.13 Weather Event Hook

Events for day, night, full moon, and weather. These events are fired on server and client.

22.2.14 Custom Weather Event Hook

Events for a specific custom *Weather Asset*. Any map can have an unlimited number of weather types and weather listeners.

22.3 Event Instigators

22.3.1 Client Text Chat Messenger

Allows Unity events to request a text chat message be sent on behalf of the client. For example, to execute a command.

The `UnityEvents.Allow_Client_Messages` and/or `UnityEvents.Allow_Client_Commands` settings must be enabled in the server `Config.json` file before these can be triggered. This ensures hosts are aware of their usage. Singleplayer defaults to enabled.

22.3.2 Server Text Chat Messenger

Allows Unity events to broadcast messages from the server. Icons and rich text are optional. Can also execute commands that are not available (yet) to NPCs like changing the weather or triggering an airdrop.

The `UnityEvents.Allow_Server_Messages` and/or `UnityEvents.Allow_Server_Commands` settings must be enabled in the server `Config.json` file before these can be triggered. This ensures hosts are aware of their usage. Singleplayer defaults to enabled.

22.3.3 Effect Spawner

Allows Unity events to spawn effect assets. When the `AuthorityOnly` field is enabled only the server will spawn effects and replicate them to clients.

22.3.4 NPC Global Event Messenger

Allows Unity events to broadcast Event NPC rewards. The `NPC Global Event Hook` can then listen for these events.

22.4 Misc

22.4.1 Fall Damage Override

Allows any game object to override the fall damage when a character lands on it or one of its descendants.

CHAPTER

TWENTYTHREE

NPC ASSETS

23.1 Introduction to NPCs

Modders can create interactable NPC characters, with a customized appearance. A dialogue box can be made to appear to the player when interacted with, which can display potential responses that can chain into more dialogue options. Dialogue can lead to special interactions, such as quests or vendors. Additionally, NPC interactions can have a set of conditions that dictate what is available to the player, and rewards for performing various actions such as turning in a quest.

23.1.1 Localization

There are additional text formatting features available for NPC localization files.

<color=*enum*></color>: Use a rarity color as the font color. Valid rarities are: `common`, `uncommon`, `rare`, `epic`, `legendary`, `mythical`, `gold`, `red`, `orange`, `yellow`, `green`, `blue`, and `purple`. Alternatively, specify a six-digit hexadecimal number representing RGB color.

<name_npc>: Insert the NPC character's name.

<name_char>: Insert the player character's name.

<br>: New line.

<pause>: Pause dialogue for 0.5 seconds before continuing.

23.2 NPC Character Assets

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`NPC`)

ID *uint16*: Must be a unique identifier.

PlayerKnowsNameFlagID *uint16*: If non-zero, NPC name is shown as ??? until bool flag is true. For example if set to 20 the NPC name is ??? until a Flag_Bool reward with ID 20 is set to true.

23.2.1 Clothing

Shirt *uint16* or *GUID*: ID or GUID of shirt to wear.

Pants *uint16* or *GUID*: ID or GUID of pants to wear.

Hat *uint16* or *GUID*: ID or GUID of hat to wear.

Backpack *uint16* or *GUID*: ID or GUID of backpack to wear.

Vest *uint16* or *GUID*: ID or GUID of vest to wear.

Mask *uint16* or *GUID*: ID or GUID of mask to wear.

Glasses *uint16* or *GUID*: ID or GUID of glasses to wear.

Holiday outfits

NPC characters can have event-specific outfits, which will only appear during the assigned seasonal event.

Has_Halloween_Outfit *flag*: Specified if event-specific clothing should be worn during the Halloween event.

Halloween_Shirt *uint16* or *GUID*: ID or GUID of shirt to wear during the Halloween event.

Halloween_Pants *uint16* or *GUID*: ID or GUID of pants to wear during the Halloween event.

Halloween_Hat *uint16* or *GUID*: ID or GUID of hat to wear during the Halloween event.

Halloween_Backpack *uint16* or *GUID*: ID or GUID of backpack to wear during the Halloween event.

Halloween_Vest *uint16* or *GUID*: ID or GUID of vest to wear during the Halloween event.

Halloween_Mask *uint16* or *GUID*: ID or GUID of mask to wear during the Halloween event.

Halloween_Glasses *uint16* or *GUID*: ID or GUID of glasses to wear during the Halloween event.

Has_Christmas_Outfit *flag*: Specified if event-specific clothing should be worn during the Festive event.

Christmas_Shirt *uint16* or *GUID*: ID or GUID of shirt to wear.

Christmas_Pants *uint16* or *GUID*: ID or GUID of pants to wear.

Christmas_Hat *uint16* or *GUID*: ID or GUID of hat to wear.

Christmas_Backpack *uint16* or *GUID*: ID or GUID of backpack to wear.

Christmas_Vest *uint16* or *GUID*: ID or GUID of vest to wear.

Christmas_Mask *uint16* or *GUID*: ID or GUID of mask to wear.

Christmas_Glasses *uint16* or *GUID*: ID or GUID of glasses to wear.

23.2.2 Appearance

While in the Appearance menu in-game, modders can press Page Down to copy the player's current appearance to clipboard.

Face *int*: Index of face image.

Hair *int*: Index of hair mesh.

Beard *int*: Index of beard mesh.

Color_Skin *hex triplet*: Six-digit hexadecimal number representing RGB color.

Color_Hair *hex triplet*: Six-digit hexadecimal number representing RGB color.

Backward *flag*: Specified if character is left-handed.

23.2.3 Pose

Primary *uint16* or *GUID*: ID or GUID of the weapon carried on the character's back, parallel to the spine.

Secondary *uint16* or *GUID*: ID or GUID of the weapon carried on the character's hip, perpendicular to the spine.

Tertiary *uint16* or *GUID*: ID or GUID of a non-weapon item to carry.

Equipped *enum* (`Primary`, `Secondary`, `Tertiary`): The item in the specified slot will be held in the character's hands, rather than carried.

Dialogue *uint16* or *GUID*: ID or GUID of the dialogue asset to open when interacted with.

Pose *enum* (`Asleep`, `Crouch`, `Passive`, `Prone`, `Rest`, `Sit`, `Stand`, `Surrender`, `Under_Arrest`): Idle animation.

Pose_Head_Offset *float*: Offset of the NPC's head from their body, in meters. Positive numbers offset it forward, while negative numbers offset it backward. Defaults to 0.1.

Pose_Lean *float*: How far the NPC leans left or right, as a number from -1 to 1. Positive numbers learn to the NPC's left, while negative numbers lean to the NPC's right. Defaults to 0.

Pose_Pitch *float*: How far the NPC leans forward or backward, in degrees. Numbers greater than 90 lean forward, while numbers less than 90 lean backward. Defaults to 90.

23.2.4 Conditions

An NPC character can be made to only appear while certain *conditions* are met by the player.

23.2.5 Localization

Name *string*: Object name in level editors.

Character *string*: Character name displayed when interacted with.

23.3 Conditions

Conditions can be held by NPC assets, interactable objects, and item blueprints. The specific property prefix may differ between asset types. For example, quests may use "Conditions" while blueprints use "Blueprint_#_Conditions".

Conditions *byte*: Total number of conditions.

Condition_#_Type *enum* (`Compare_Flags`, `Date_Counter`, `Flag_Bool`, `Flag_Short`, `Currency`, `Experience`, `Item`, `Kills_Animal`, `Kills_Horde`, `Kills_Object`, `Kills_Player`, `Kills_Tree`, `Kills_Resource`, `Player_Life_Food`, `Player_Life_Health`, `Player_Life_Virus`, `Player_Life_Water`, `Quest`, `Reputation`, `Skillset`, `Holiday`, `Time_Of_Day`, `Weather_Blend_Alpha`, `Weather_Status`)

Condition_#_Reset *flag*: Set back to equivalent of 0 when completed.

Condition_#_Logic *enum* (`Less_Than`, `Less_Than_Or_Equal_To`, `Equal`, `Not_Equal`, `Greater_Than_Or_Equal_To`, `Greater_Than`): Compare current state to target state.

Condition_#_UI_Requirements *string*: Comma-separated condition indices. If set, only show this condition in the UI when the conditions with these indices are met. For example, a condition with "1, 2" will only be shown when conditions 1 and 2 have been completed.

23.3.1 Flags

Compare_Flags

Compare left-hand flag A, to right-hand flag B.

Condition_#_Type *enum* (`Compare_Flags`)

Condition_#_A_ID *uint16*: Left-hand flag ID.

Condition_#_Allow_A_Unset *bool*: Pass condition flag onto player, if they do not have the flag already.

Condition_#_B_ID *uint16*: Right-hand flag ID.

Condition_#_Allow_B_Unset *bool*: Pass condition if player does not have the flag yet.

Date_Counter

Every in-game morning, the world's "date counter" is incremented. In a fresh save it starts at zero. This condition takes the remainder of the date counter divided by `Divisor` and compares it with `Value` according to `Logic`.

For example, an in-game event can be configured to occur every 4th and 5th day by setting `Divisor` to 5, `Value` to 3, and `Logic` to `Greater_Than_Or_Equal_To`.

Condition_#_Type *enum* (`Date_Counter`)

Condition_#_Value *int64*: Number to compare the remainder with.

Condition_#_Divisor *int64*: Number to divide the world date counter by.

Flag_Bool

Boolean flag condition.

Condition_#_Type *enum* (`Flag_Bool`)

Condition_#_ID *uint16*: ID of flag to check.

Condition_#_Value *bool*: Target value, as a boolean.

Condition_#_Allow_Unset *flag*: Pass condition if player does not have the flag yet.

Flag_Short

Short flag condition.

Condition_#_Type *enum* (`Flag_Short`)

Condition_#_ID *uint16*: ID of flag to check.

Condition_#_Value *int*: Target value for the flag, as a 16-bit signed integer.

Condition_#_Allow_Unset *flag*: Pass condition if player does not have the flag yet.

23.3.2 Player

Currency

Refer to *Currency* documentation.

Condition_#_Type *enum* (`Currency`)

Condition_#_GUID *string*: GUID of currency asset.

Condition_#_Value *int*: Target value, in terms of currency.

Experience

Condition_#_Type *enum* (`Experience`)

Condition_#_Value *int*: Target value, in terms of experience.

Item

Condition_#_Type *enum* (`Item`)

Condition_#_ID *uint16*: ID of item to search player's inventory for.

Condition_#_Amount *int*: Quantity of the item required.

Kills_Animal

Condition_#_Type *enum* (`Kills_Animal`)

Condition_#_ID *uint16*: ID of short flag to track stat.

Condition_#_Value *int*: Target value, in terms of animal kills.

Condition_#_Animal *uint16*: ID of animal required.

Kills_Horde

Condition_#_Type *enum* (`Kills_Horde`)

Condition_#_ID *uint16*: ID of short flag to track stat.

Condition_#_Value *int*: Target value, in terms of beacons completed.

Condition_#_Nav *byte*: Index of the navmesh that beacons should be completed in, seen as visible in the level editor.

Kills_Object

Condition_#_Type *enum* (`Kills_Object`)

Condition_#_ID *uint16*: ID of short flag to track stat.

Condition_#_Value *int*: Target value, in terms of object destructions.

Condition_#_Object *string*: GUID of object required.

Condition_#_Nav *byte*: Index of the navmesh that objects should be destroyed in, seen as visible in the level editor.

Kills_Player

Condition_#_Type *enum* (`Kills_Player`)

Condition_#_ID *uint16*: ID of short flag to track stat.

Condition_#_Value *int*: Target value, in terms of player kills.

Kills_Tree

Condition_#_Type *enum* (`Kills_Tree`)

Condition_#_ID *uint16*: ID of short flag to track stat.

Condition_#_Value *int*: Target value, in terms of resource destructions.

Condition_#_Tree *string*: GUID of resource required.

Kills_Zombie

Condition_#_Type *enum* (`Kills_Resource`)

Condition_#_ID *uint16*: ID of short flag to track stat.

Condition_#_Value *int*: Target value, in terms of zombies killed.

Condition_#_Zombie *enum* (`Acid`, `Boss_All`, `Boss_Electric`, `Boss_Elver_Stomper`, `Boss_Fire`, `Boss_Magma`, `Boss_Nuclear`, `Boss_Spirit`, `Boss_Wind`, `Burner`, `Crawler`, `DL_Blue_Volatile`, `DL_Red_Volatile`, `Flanker_Friendly`, `Flanker_Stalk`, `Mega`, `None`, `Normal`, `Spirit`, `Sprinter`): Type of zombie required.

Condition_#_Spawn_Quantity *int*: Number of zombies to spawn. Defaults to 1.

Condition_#_Nav *byte*: Index of the navmesh that zombies should be killed in, seen as visible in the level editor.

Condition_#_Radius *float*: Radius around players that zombies should be killed within, in meters. When a navmesh is unset and a radius is not specified, the radius defaults to 512 meters and is used for the condition.

Condition_#_MinRadius *float*: Zombies must be killed at least this many meters away from the player.

Condition_#_Spawn *flag*: Specified if the zombie type should be forcefully generated upon entering the area, which will then be deleted upon leaving the area.

Condition_#_LevelTableOverride *int*: Unique ID of a zombie type shown in the level editor. If set, the zombie spawned will use that type. Defaults to -1.

Player_Life_Food

Condition_#_Type *enum* (`Player_Life_Food`)

Condition_#_Value *int*: Target value, in terms of the player's current food.

Player_Life_Health

Condition_#_Type *enum* (`Player_Life_Health`)

Condition_#_Value *int*: Target value, in terms of the player's current health.

Player_Life_Virus

Condition_#_Type *enum* (`Player_Life_Virus`)

Condition_#_Value *int*: Target value, in terms of the player's current immunity.

Player_Life_Water

Condition_#_Type *enum* (`Player_Life_Water`)

Condition_#_Value *int*: Target value, in terms of the player's current water.

Quest

Condition_#_Type *enum* (`Quest`)

Condition_#_ID *uint16*: ID of quest to check for.

Condition_#_Status *enum* (`None`, `Active`, `Ready`, `Completed`): Current state of the quest.

Condition_#_Ignore_NPC *flag*: Player does not need to be talking to an NPC within 20 meters for the quest to be completable and turned in.

Reputation

Condition_#_Type *enum* (`Reputation`)

Condition_#_Value *int*: Target value, in terms of reputation.

Skillset

Condition_#_Type *enum* (`Skillset`)

Condition_#_Value *enum* (`Army`, `Camp`, `Chef`, `Farm`, `Fire`, `Fish`, `Medic`, `None`, `Police`, `Thief`, `Work`): Target value, as the skillset. For example, this condition could be used to offer unique questlines, dialogue, or blueprints depending on the player's chosen skillset.

23.3.3 World

Holiday

Condition_#_Type *enum* (`Holiday`)

Condition_#_Value *enum* (*ENPCHoliday*): Target value, as the holiday.

Is_Full_Moon

Condition_#_Type *enum* (`Is_Full_Moon`)

Condition_#_Value *bool*: If true the condition passes when the full moon is up, otherwise if false the condition passes when the full moon is **not** up.

Time_Of_Day

Condition_#_Type *enum* (`Time_Of_Day`)

Condition_#_Second *int*: Second of a 24-hour clock (military time) to compare against, where 43,200 is noon and 86,400 is a full day.

Weather_Blend_Alpha

The weather blend alpha condition compares the current intensity to a target value. For example, an NPC could sell umbrellas while rain is greater than 50% (0.5) blended in. This condition is supported by visibility, but is more expensive for visibility than the state condition because each listening object is updated when the intensity changes by 1% (0.01).

Condition_#_Type *enum* (`Weather_Blend_Alpha`)

Condition_#_GUID *string*: GUID of weather required.

Condition_#_Value *float* [0, 1]: Target value, as the weather intensity blend.

Weather_Status

The weather status condition tests the state of the global weather. This condition is supported by visibility.

Condition_#_Type *enum* (`Weather_Status`)

Condition_#_GUID *string*: GUID of weather required.

Condition_#_Value *enum* (`Active`, `Fully_Transitioned_In`, `Fully_Transitioned_Out`, `Transitioning`, `Transitioning_In`, `Transitioning_Out`): Target value, as the weather status.

23.3.4 Localization

Condition_#: Name of the condition as it appears in user interfaces.

23.4 Dialogue Assets

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Dialogue`)

ID *uint16*: Must be a unique identifier. Values less than 2,000 are reserved for official content.

23.4.1 Messages

Properties pertaining to dialogue performed by the NPC. Dialogue can utilize conditions and rewards. Messages that meet all of their conditions will be shown, and can grant rewards when the message is shown. These are prefixed with `Message_#_`. For example, `Message_0_Condition_0_Type Flag_Bool`. For more information, refer to the documentation for *Conditions* and *Rewards* respectively.

Messages *int32*: Total number of possible messages.

Message_#_Pages *byte*: Total number of pages the message has.

Message_#_Responses *byte*: Total number of responses to be shown when this message is shown. If 0, then all messages are automatically a candidate to be shown. Defaults to 0.

Message_#_Response_# *byte*: Index of the response to show.

Message_#_Prev *uint16* or *GUID*: ID or GUID of dialogue to return to if there are no responses available for this message. Defaults to 0.

Message_#_FaceOverride *byte*: Optional index of face image to use when this message is opened. Face is reset to character's default when unspecified or when dialogue is closed.

23.4.2 Responses

Properties pertaining to dialogue available to the player. Dialogue can utilize conditions and rewards. Responses are only visible when conditions are met, and can grant rewards when selected. These are prefixed with *Response_#_*. For example, *Response_0_Reward_0_Type Quest*. For more information, refer to the documentation for *Conditions* and *Rewards* respectively.

Responses *byte*: Total number of possible responses.

Response_#_Messages *byte*: Total number of messages to only show this response for. If 0, then it is shown for all messages. Defaults to 0.

Response_#_Message_# *uint16*: Index of the message to show for.

Response_#_Dialogue *uint16* or *GUID*: ID or GUID of the dialogue to open when selected.

Response_#_Quest *uint16* or *GUID*: ID or GUID of the quest to preview when selected.

Response_#_Vendor *uint16* or *GUID*: ID or GUID of the vendor to open when selected.

23.4.3 Localization

Message_#_Page_# *Rich Text*: Text shown for the corresponding message page.

Response_# *Rich Text*: Text shown for the corresponding response option.

23.5 Quest Assets

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Quest`)

ID *uint16*: Must be a unique identifier.

23.5.1 Conditions and Rewards

Quests can be turned in when conditions are met, and players can receive rewards for turning quests in. For more information, refer to the documentation for *Conditions* and *Rewards* respectively.

23.5.2 Localization

Name *string*: Quest name in user interfaces.

Description *Rich Text*: Quest description in user interfaces.

23.6 Rewards

Rewards can be granted by NPC assets, interactable objects, and item blueprints. The specific property prefix may differ between asset types. For example, quests may use “Rewards” while consumables use “Quest_Rewards”.

Rewards *byte*: Total number of rewards.

Reward_#_Type *enum* (`Flag_Bool`, `Flag_Math`, `Flag_Short`, `Flag_Short_Random`, `Achievement`, `Currency`, `Cutscene_Mode`, `Event`, `Experience`, `Item`, `Item_Random`, `Hint`, `Player_Life_Food`, `Player_Life_Health`, `Player_Life_Virus`, `Player_Life_Water`, `Player_Spawnpoint`, `Quest`, `Reputation`, `Rewards_List_Asset`, `Teleport`, `Vehicle`)

Reward_#_GrantDelaySeconds *float*: If set, the reward will be queued for the specified number of seconds before being granted to the player. When the player dies any pending rewards are cancelled. Defaults to `-1`.

23.6.1 Flags

Flag_Bool

Reward_#_Type *enum* (`Flag_Bool`)

Reward_#_ID *uint16*: ID of flag to set.

Reward_#_Value *bool*: Set flag to Boolean value, of either “True” or “False”.

Flag_Math

Reward_#_Type *enum* (`Flag_Math`)

Reward_#_A_ID *uint16*: ID of flag to apply math to.

Reward_#_B_ID *uint16*: ID of flag containing value to be applied mathematically. If not specified then *B_Value* is used instead.

Reward_#_B_Value *int16*: default value to be applied mathematically if flag B has not been set on the player or if *B_ID* is zero.

Reward_#_Operation *enum* (`Addition`, `Assign`, `Division`, `Modulo`, `Multiplication`, `Subtraction`): For example, using the Addition operation would set A to the value of A + B.

Flag_Short

Reward_#_Type *enum* (`Flag_Short`)

Reward_#_ID *uint16*: ID of flag to modify.

Reward_#_Value *int*: Modify flag's current value with this short value.

Reward_#_Modification *enum* (`Assign`, `Decrement`, `Increment`): Set value, subtract value, or add value.

Flag_Short_Random

Reward_#_Type *enum* (`Flag_Short_Random`)

Reward_#_ID *uint16*: ID of flag to modify.

Reward_#_Min_Value *int*: Minimum short value to modify flag's current value by.

Reward_#_Max_Value *int*: Maximum short value to modify flag's current value by.

Reward_#_Modification *enum* (`Assign`, `Decrement`, `Increment`): Set value, subtract value, or add value.

23.6.2 Non-flags

Achievement

Reward_#_Type *enum* (`Achievement`)

Reward_#_ID *string*: ID of achievement to grant. Only specific achievements can be granted as a reward.

Currency

Refer to *Currency* documentation.

Reward_#_Type *enum* (`Currency`)

Reward_#_GUID *string*: GUID of currency asset.

Reward_#_Value *int*: Amount of currency to reward.

Cutscene Mode

Not as exciting as it sounds. While active, the first-person viewmodel is hidden and certain item actions like shooting are disabled. It's saved/loaded, but resets on death.

Reward_#_Type *enum* (`Cutscene_Mode`)

Reward_#_Value *bool*: Whether cutscene mode should currently be active.

Event

Reward_#_Type *enum* (`Event`)

Reward_#_ID *string*: ID of event to broadcast. This can be used by c# plugins with the `NPCEventManager` class, or Unity events with the *NPC Global Event component*. For example, when an event with ID "Fireworks" is broadcast all of the components with event ID "Fireworks" will have their corresponding Unity event triggered as well, in this case perhaps to spawn a fireworks effect.

Experience

Reward_#_Type *enum* (`Experience`)

Reward_#_Value *int*: Amount of experience to reward.

Item

Reward_#_Type *enum* (`Item`)

Reward_#_ID *uint16*: ID of item to reward.

Reward_#_Amount *int*: Amount of item to reward.

Reward_#_Auto_Equip *bool*: Item should be automatically equipped by the player.

Reward_#_Ammo *byte*: Override for the amount of ammuntion that should be loaded in the item reward.

Reward_#_Barrel *uint16*: Override for the barrel attachment that should be attached to the item reward.

Reward_#_Grip *uint16*: Override for the grip attachment that should be attached to the item reward.

Reward_#_Magazine *uint16*: Override for the magazine attachment that should be attached to the item reward.

Reward_#_Origin *EItemOrigin*: Set the item origin. For example, setting the origin to `Admin` will cause items to spawn at full quality. Defaults to *Craft*.

Reward_#_Sight *uint16*: Override for the sight attachment that should be attached to the item reward.

Reward_#_Tactical *uint16*: Override for the tactical attachment that should be attached to the item reward.

Item_Random

Reward_#_Type *enum* (`Item_Random`)

Reward_#_ID *uint16*: ID of spawn table that the random item reward should come from.

Reward_#_Amount *int*: Amount of item to reward.

Reward_#_Auto_Equip *flag*: Item should be automatically equipped by the player.

Reward_#_Origin *EItemOrigin*: Set the item origin. For example, setting the origin to `Admin` will cause items to spawn at full quality. Defaults to *Craft*.

Hint

Reward_#_Type *enum* (`Hint`)

Reward_#_Text *Rich Text*: Text to display as a hint.

Reward_#_Duration *float*: Duration of the hint, in seconds. Defaults to 2 seconds.

Player Life Food

Reward_#_Type *enum* (`Player_Life_Food`)

Reward_#_Value *int*: Amount of food to add. Can be negative to decrease food.

Player Life Health

Reward_#_Type *enum* (`Player_Life_Health`)

Reward_#_Value *int*: Amount of health to add. Can be negative to decrease health.

Player Life Virus

Reward_#_Type *enum* (`Player_Life_Virus`)

Reward_#_Value *int*: Amount of virus to add. Can be negative to decrease virus level.

Player Life Water

Reward_#_Type *enum* (`Player_Life_Water`)

Reward_#_Value *int*: Amount of water to add. Can be negative to decrease water.

Player Spawnpoint

Reward_#_Type *enum* (`Player_Spawnpoint`)

Reward_#_ID *string* Override the player's default spawn location, using the spawnpoint name set in the Devkit level editor or a map location node name. For example, `Liberator_Jet`. Saved and loaded between sessions. If empty, the override is removed and the default spawns are used. The `SetNpcSpawnId` admin command is useful for testing this.

Hint: On the Buak map, the player can talk with Kira to claim a room in the Factory using this reward type.

Quest

Reward_#_Type *enum* (`Quest`)

Reward_#_ID *uint16*: Quest ID to give as a reward.

Reputation

Reward_#_Type *enum* (`Reputation`)

Reward_#_Value *int*: Amount of reputation to reward.

Rewards List Asset

Reward_#_Type *enum* (`Rewards_List_Asset`)

Reward_#_GUID *Asset Pointer*: *Rewards List* to grant directly, or *Spawn Table* to resolve into one.

Teleport

Reward_#_Type *enum* (`Teleport`)

Reward_#_Spawnpoint *string*: Location to teleport the player to as a reward, using the spawnpoint name as set in the Devkit level editor. For example, `Liberator_Jet`.

Vehicle

Reward_#_Type *enum* (`Vehicle`)

Reward_#_ID : ID of Vehicle to be given.

Reward_#_Spawnpoint *string*: Location to spawn the vehicle in as a reward, using the spawnpoint name as set in the Devkit level editor. For example, `Liberator_Jet`.

23.6.3 Localization

Reward_#: Name of the reward as it appears in user interfaces.

23.7 Rewards List Assets

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`RewardsList`)

The *Rewards_List_Asset* NPC reward type can either grant a rewards list asset directly, or a *Spawn Table Asset* which resolves to the final asset. This could be used, for example, to randomly select one of several rewards list assets which then grants the player a gun along with related ammo items.

A *Rewards List Volume* placed in the level editor can also reference a rewards list asset, and will grant the rewards if the conditions are met when a player enters the volume.

Conditions must be met to grant the rewards. For more information, refer to the documentation for *Conditions* and *Rewards* respectively.

23.8 Vendor Assets

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Vendor`)

ID *uint16*: Must be a unique identifier.

23.8.1 Buying

Properties pertaining to items that the vendor is willing to buy from players. Vendors can set conditions for the items they are buying. These conditions are prefixed with `Buying_#_`. For example, `Buying_0_Conditions 1`. For more information, refer to the documentation for *Conditions* and *Rewards* respectively.

Buying *byte*: Total number items being bought by the vendor.

Buying_#_ID *uint16*: ID of item to buy from the player.

Buying_#_Cost *uint32*: Amount of currency to pay the player. Defaults to experience points as the currency, unless the Currency property has been set.

23.8.2 Selling

Properties pertaining to items or vehicles that the vendor is willing to sell to players. Vendors can set conditions for the items/vehicles they are selling. These conditions are prefixed with `Selling_#_`. For example, `Selling_0_Conditions 1`. For more information, refer to the documentation for *Conditions* and *Rewards* respectively.

Selling *byte*: Total number of items/vehicles being sold by the vendor.

Selling_#_Type *enum* (`Item`, `Vehicle`): Type of asset being sold.

Selling_#_ID *uint16*: ID of item/vehicle to sell to the player.

Selling_#_Cost *uint32*: Amount of currency to pay the vendor. Defaults to experience points as the currency, unless the Currency property has been set.

Selling_#_Spawnpoint *string*: Location to spawn the purchased vehicle, using the spawnpoint name as set in the Devkit level editor. For example, `Liberator_Jet`.

23.8.3 Other Properties

Disable_Sorting *flag*: Disable vendor sorting.

Currency *string*: GUID of the *currency asset* to use as currency instead of experience points.

FaceOverride *byte*: Optional index of face image to use when this vendor is opened. Face is reset to character's default when unspecified or when a new message is opened.

23.8.4 Localization

Name *string*: Vendor name in user interfaces.

Description *Rich Text*: Vendor description in user interfaces.

CHAPTER

TWENTYFOUR

OBJECT ASSETS

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Small`, `Medium`, `Large`, `NPC`, `Decal`): The object's type is used for sorting, pathfinding, collision, and culling. Small objects are used for clutter and decoration, medium objects fill out the layout, and large objects make up the level. When using the `NPC` enumerator, refer to the documentation for *NPC Character Assets* as well.

ID *uint16*: Must be a unique identifier.

24.1 Object Properties

Add_Kill_Triggers *bool*: If true, this adds a kill volume inside the object that will kill players who clip inside it. This is helpful for preventing out-of-bounds exploits, such as players trying to build bases inside boulder objects.

Add_Night_Light_Script *flag*: Adds a script to the object that looks for a transform named "Light". During the night, the light will be turned on. During the day, the light will be turned off.

Allow_Structures *flag*: Structures can be built on top of this object. For example, a fake grass object may want to use this property.

Causes_Fall_Damage *bool*: Whether or not players should take fall damage when landing on this object. Defaults to true.

Chart *enum* (*EObjectChart*): When an object obstructs the terrain, it can appear on a map's chart view. The pixel sampled for the object's color on the chart view can be set or overridden with this property. By default, `Type Large` objects use the same pixel as the `Large` enumerator, and `Type Medium` objects use the same pixel as the `Medium` enumerator. Defaults to `Ignore` when using `Type NPC` or `Type Decal`. Otherwise, defaults to `None`.

Christmas_Redirect *GUID*: GUID of the object that should appear during the Festive holiday.

Collision_Important *flag*: Prevent collision from being disabled. When using `Type Large`, this flag is automatically included.

Exclude_From_Level_Batching *bool*: Exclude this object from *level batching*. This property may be helpful when using elaborate setups with Unity Event components. Defaults to true when using `Type Decal` or `Type NPC`.

Foliage *GUID*: GUID of a *foliage asset*. This property is useful for objects such as fake grass, which may want foliage to bake on top of the object similar to terrain materials.

Fuel *flag*: Fuel can be siphoned from this object. Deprecated in favor of `Interactability Fuel`.

Halloween_Redirect *GUID*: GUID of the object that should appear during the Halloween holiday.

Has_Clip_Prefab *bool*: Whether or not the object has a Clip.prefab. If the object should use the same prefab on the server as on the client, set to false. For example, most official content uses `Has_Clip_Prefab false`. Defaults to true.

Holiday_Restriction *enum* (*ENPCHoliday*): If a valid value is specified, then this object will only be visible during the corresponding holiday. The specified holiday will be appended to the object's user-friendly name. Defaults to `None`.

Is_Gore *bool*: Whether or not this object should be visible if the player has disabled "Show Blood Splatters".

Landmark_Quality *enum* (`Off`, `Low`, `Medium`, `High`, `Ultra`): The value that the "Landmarks" graphical setting must be set to in order to see a low detail model of this object from far away distances. Defaults to `Low`.

Material_Palette *GUID*: GUID of the *Material Palette Asset* that should be used by the object.

Refill *flag*: Water can be siphoned from this object. Deprecated in favor of `Interactability Water`.

Snowshoe *flag*: This object should not leave a footprint when baking materials.

Soft *flag*: Vehicles should not take damage when colliding with this object.

Use_Water_Height_Transparent_Sort *flag*: Useful for transparent objects, such as glass.

24.1.1 Decals

Decal_Alpha *flag*: This flag should be set if the decal has a transparent texture. Requires `Type Decal`.

Decal_X *float*: Override the scale of the decal, on the -axis. Requires `Type Decal`.

Decal_Y *float*: Override the scale of the decal, on the -axis. Requires `Type Decal`.

Decal_LOD_Bias *float*: Multiplier for the LOD's switching distance. Defaults to 1. Requires `Type Decal`.

24.1.2 Interior Culling

Exclude_From_Culling_Volumes *bool*: If set to true, this object will not be managed by culling volumes. For example, the aerospace facility on the Germany map is excluded from culling volumes, so that manually-placed culling volumes can hide large objects like shipping containers without accidentally hiding the giant aerospace facility itself.

LOD *enum* (`None`, `Mesh`, `Area`): How interior culling should be determined. Using the `Mesh` enumerator will use the mesh bounds to determine what is inside the object. For concave objects, you can use the `Area` enumerator instead and add multiple Occlusion Area components for the interior volumes.

LOD_Bias *float*: Multiplier on the threshold distance for interior culling. Requires that `LOD` has been set.

LOD_Center_X float: Offset for the culling volume's local position, on the -axis. Requires that `LOD` has been set.

LOD_Center_Y float: Offset for the culling volume's local position, on the -axis. Requires that `LOD` has been set.

LOD_Center_Z float: Offset for the culling volume's local position, on the -axis. Requires that `LOD` has been set.

LOD_Size_X float: Offset for the culling volume's size, on the -axis. Requires that `LOD` has been set.

LOD_Size_Y float: Offset for the culling volume's size, on the -axis. Requires that `LOD` has been set.

LOD_Size_Z float: Offset for the culling volume's size, on the -axis. Requires that `LOD` has been set.

24.1.3 Interactables

Interactability *enum* (None, Binary_State, Dropper, Note, Water, Fuel, Rubble, NPC, Quest): All Interactability_ properties will require that this property has been set. The enumerator selected for this property will affect which properties can be used, how these properties will function when used, and how this object will behave in-game. Defaults to the NPC enumerator when using Type NPC, otherwise this property will default to None.

- Binary_State objects will change between their two states when interacted with – such as an open or closed door.
- Dropper objects can spawn items when interacted with.
- Note objects can display lines of text when interacted with.
- Water objects can be siphoned for water, and Fuel objects can be siphoned for fuel.
- Rubble objects are destructible. It is preferable to use Rubble Destroy instead of Interactability Rubble.
- NPC objects can provide access to dialogue, quests, and vendors.
- Quest objects can be interacted with, but unlike other options they have no additional functionality.

Note: Although Interactability properties can be used to create a destructible object, it is preferable to use Rubble properties as they are more specific. This allows for creating destructible objects that are also interactable.

Interactability_Blade_ID *byte*: When using Interactability Rubble, weapons are unable to damage this object unless they have a matching BladeID_# value. Defaults to 0.

Interactability_Delay *float*: In seconds, the cooldown before the object can be interacted with again.

Interactability_Drops *byte*: Total number of items dropped from an object using Interactability Dropper. This property is used in conjunction with Interactability_Drop_#. Defaults to 0. It is preferable to use the Interactability_Reward_ID property instead.

Interactability_Drop_# *uint16*: ID of an item that should be dropped. This property is used in conjunction with Interactability_Drops.

Interactability_Editor *enum* (None, Toggle): Determines how this interactable object should appear in the level editor. If this is set to Toggle, then the object's alternative state will be shown. Defaults to None.

Interactability_Effect *GUID* or *uint16*: GUID or legacy ID of an *EffectAsset* to play when interacted with. When using Interactability Rubble, this is effect is played when a section of the object is destroyed.

Interactability_Finale *GUID* or *uint16*: GUID or legacy ID of an *EffectAsset* to play when all sections of the object using Interactability Rubble are destroyed. If this property is used, then all of the dead object's sections will also be hidden when fully destroyed.

Interactability_Health *uint16*: Total amount of health each section of the object has, when using Interactability Rubble. Defaults to 0.

Interactability_Hint *enum* (Door, Switch, Fire, Generator, Use, Custom): Localization key to use for the interact prompt. Setting this to Custom allows for displaying custom text instead, when used in conjunction with Interact.

Interactability_Invulnerable *flag*: This resource cannot be damaged by lower-power *Weapon Assets* that do not have the Invulnerable flag, when using Interactability Rubble.

Interactability_Nav *enum* (None, On, Off): How navigation should change when the object's state is changed. Defaults to None.

Interactability_Power *enum* (None, Toggle, Stay): Whether or not this object must be powered to be usable. When set to None, this object cannot be powered. When set to Toggle, the object must be powered to be interacted with.

When set to `Stay`, the object must be powered to remain on. For example, a door might use `Toggle` if it should remain open after it loses power, while a streetlight might use `Stay` so that the light turns off when it loses power. Defaults to `None`.

Interactability_Proof_Explosion *flag*: Immune to area-of-effect explosive damage, when using `Interactability Rubble`.

Interactability_Remote *flag*: Disables the ability for players to interact with this via a button prompt.

Interactability_Reset *float*: Delay before an interacted object resets, or a destroyed object respawns, in seconds.

Interactability_Resource *uint16*: When using `Interactability Fuel` or `Interactability Water`, this value is how many units of fuel or water is stored in the object. Defaults to 0.

Interactability_Reward_ID *uint16*: ID of an item *spawn table* to use for rewards, when using `Interactability Rubble`. Defaults to 0.

Interactability_Rewards_Min *byte*: Minimum amount of item drops to reward, when using `Interactability Rubble`. Defaults to 1.

Interactability_Rewards_Max *byte*: Maximum amount of item drops to reward, when using `Interactability Rubble`. Defaults to 1.

Interactability_Reward_Probability *float*: Probability of receiving a reward, as a decimal-to-percent chance, when using `Interactability Rubble`. Defaults to 1.

Interactability_Reward_XP *uint32*: Amount of experience to reward when the object using `Interactability Rubble` is destroyed.

Interactability_Text_Lines *uint16*: Total number of lines to display when an object using `Interactability Note` is interacted with. This property is used in conjunction with `Interactability_Text_Line_#`. Defaults to 0.

24.1.4 Rubble

Rubble *enum* (`None`, `Destroy`): The destruction mode that should be used, although the only functional option for this is `Destroy`. All `Rubble_` properties require that this property has been set.

Rubble_Blade_ID *byte*: Weapons are unable to damage this object unless they have a matching `BladeID_#` value. Defaults to 0.

Rubble_Editor *enum* (`Alive`, `Dead`): Determines how this destructible object should appear in the level editor. If this is set to `Dead`, the fully destroyed state of the object will be shown. Defaults to `Alive`.

Rubble_Effect *GUID* or *uint16*: GUID or legacy ID of an *EffectAsset* to play when a section of the destructible object is destroyed.

Rubble_Finale *GUID* or *uint16*: GUID or legacy ID of an *EffectAsset* to play when all sections of the destructible object are destroyed. If this property is used, then all of the dead object's sections will also be hidden when fully destroyed.

Rubble_Health *uint16*: Total amount of health each section of the object has. Defaults to 0.

Rubble_Invulnerable *flag*: This resource cannot be damaged by lower-power *Weapon Assets* that do not have the `Invulnerable` flag.

Rubble_Proof_Explosion *flag*: Immune to area-of-effect explosive damage.

Rubble_Reset *float*: Delay before a destroyed object respawns, in seconds.

Rubble_Reward_ID *uint16*: ID of an item *spawn table* to use for rewards. Defaults to 0.

Rubble_Rewards_Min *byte*: Minimum amount of item drops to reward. Defaults to 1.

Rubble_Rewards_Max *byte*: Maximum amount of item drops to reward. Defaults to 1.

Rubble_Reward_Probability *float*: Probability of receiving a reward, as a decimal-to-percent chance. Defaults to 1.

Rubble_Reward_XP *uint32*: Amount of experience to reward when the destructible object is destroyed.

24.1.5 Conditions and Rewards

Conditions can be used to control the visibility of an object. For example, if an object should only be visible after a certain quest has been completed. These properties do not have a unique prefix, and instead use the standard `Conditions` and `Condition_#` property names.

Conditions and *rewards* can also be tied to the interactability of an object. An object could become interactable during a quest, and then trigger rewards (such as completing the quest) once it has been interacted with. These properties are prefixed with `Interactability_`. For example, `Interactability_Conditions` and `Interactability_Reward_#`.

24.2 Localization

Name *string*: Object name in user interfaces.

Interact *string*: When an interactable object is using `Interactability_Hint Custom`, this property is used to set the text that should be displayed as the interact prompt for the object.

Interactability_Text_Line_# *Rich Text*: A line of text that should be displayed when an object using `Interactability Note` is interacted with. This property is used in conjunction with `Interactability_Text_Lines`.

CHAPTER

TWENTYFIVE

OUTFIT ASSETS

A selection of clothing items that should be worn together when generating preview images for outfits. Outfit preview images can be generated with the [F1] menu available from the Workshop section of the main menu.

25.1 Metadata

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *string*: `SDG.Unturned.OutfitAsset`

25.2 Outfit Properties

Items *array* of *Asset Pointers*: Asset pointers to clothing item(s). These clothing items will be worn together in any preview images generated of this outfit. For example:

```
"Asset"
{
	"Items"
	[
		// Top
		"9fd6032f9a24404eaf28961fc7f2d289"

		// Bottom
		"d4ec52a157f746edbc3a4df8ae79ddef"

		// Mask
		"1adc30f0dbf246eba1c0c6a183206aad"

		// Hat
		"daf02a225ebc4b76ae51ca485706e470"
	]
}
```

CHAPTER

TWENTYSIX

PHYSICS MATERIAL ASSETS

Work-in-progress feature to allow custom physics effects rather than hardcoding them.

The `PhysicsMaterialAsset` type associates gameplay properties and effects with custom Unity "physic" materials. For example if none of the built-in physics materials has appropriate effects for the surface of Mars, a custom Unity physics material named "MarsDirt" could be created. Then a physics material asset with `UnityName` set to "MarsDirt" and `Fallback` of 33650ff924b34f8d9c5a0fd97418cd3e (built-in gravel) could add custom effects for the Martian surfaces.

The `PhysicsMaterialExtensionAsset` type can be used to insert custom properties into built-in physics materials. For example if a custom laser gun should leave burn marks on the hit surface rather than bullet holes, an extension asset can set the *Base* property to a built-in physics material to add custom effects.

26.1 Properties

Type *string*: `SDG.Unturned.PhysicsMaterialAsset` or `SDG.Unturned.PhysicsMaterialExtensionAsset`

UnityName *string* or `UnityNames` *string array*: Names of Unity "physic" materials to associate with this asset. Not set by extension assets. Multiple names can be specified as an array because the old built-in physics materials had several variants for special cases that should now be handled by these assets.

Fallback *Asset Pointer*: Points to a different physics material asset. Fallbacks are used when a property is not set. For example the snow physics material does not have a bullet casing bounce audio clip, so the gravel fallback is used instead.

Base *Asset Pointer*: Points to a physics material asset to extend. Properties from the extension asset will be appended to the base asset.

AudioDefs *dictionary*: pairs of key/name and *Master Bundle Pointer* to OneShotAudioDefinition. For example the `ParticleSystemCollisionAudio` component `MaterialPropertyName` is referring to these keys. Official properties include:

- BulletCasingBounce: used by vanilla non-shotgun particle collision audio.
- BulletImpact: fired bullet hitting surface.
- BipedLand: player landing on a surface after falling. Could be used for other two-legged characters (zombies) in the future.
- FootstepWalk: individual non-sprinting footstep.
- FootstepRun: individual sprinting footstep.
- LegacyImpact: will probably be phased-out. Still used by vehicle bumper collision and as a fallback for melee impact.
- MeleeImpact: melee attack hitting surface.

- ShotgunShellBounce: used by vanilla shotgun particle collision audio.

IsArable *bool*: If true, crops can be planted on this material.

HasOil *bool*: If true, oil drills can be placed on this material. Note at the time of writing (2022-02-10) oil drills can only be placed on terrain materials.

Character_Friction_Mode *enum* (`ImmediatelyResponsive`, `Custom`): If custom the acceleration, deceleration, and max speed properties are used. Replacement for the hardcoded ice and slippery metal plate.

Character_Acceleration_Multiplier *float*: Default acceleration is equal to the target move speed.

Character_Deceleration_Multiplier *float*: Default deceleration is 2m/s^2.

`Character_Max_Speed_Multiplier` *float*: Allows speed to reach up to this multiplied by the target move speed.

CHAPTER

TWENTYSEVEN

RESOURCE ASSETS

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Resource`)

ID *uint16*: Must be a unique identifier.

27.1 Resource Properties

Auto_Skybox *flag*: Generate and assign a material and texture to the resource's Skybox prefab. The mesh should have custom normals to match the lighting. For example, vanilla pine trees have upward normals, whereas spherical trees have outward normals.

BladeID *byte*: Weapons are unable to damage this resource unless they have a matching `BladeID_#` value. Defaults to 0.

Bypass_ID_Limit *flag*: Allows for using an `ID` value within the range reserved for official content.

Chart *enum* (*EObjectChart*): When a resource obstructs the terrain, it can appear on a map's chart view. The pixel sampled for the resource's color on the chart view can be set or overridden with this property. Defaults to `None`, and will sample (14, 0) from the Layer_Strip.

Christmas_Redirect *GUID*: GUID of the resource that should appear during the Festive holiday.

Exclude_From_Level_Batching *bool*: Exclude this resource from *level batching*. This property may be helpful when using elaborate setups with Unity Event components. Defaults to true when the `SpeedTree` flag is set.

Explosion *GUID* or *uint16*: GUID or legacy ID of *EffectAsset* to play when destroyed.

Forage *flag*: Instead of being destroyable, this resource can be foraged from by interacting with it.

Forage_Reward_Experience *uint32*: Amount of experience to reward when the resource is foraged from. Defaults to 1.

Halloween_Redirect *GUID*: GUID of the resource that should appear during the Halloween holiday.

Health *uint16*: Total amount of health the resource has. Defaults to 0.

Holiday_Restriction *enum* (*ENPCHoliday*): If a valid value is specified, then this resource will only be visible during the corresponding holiday. The specified holiday will be appended to the resource's user-friendly name. Defaults to `None`.

Log *uint16*: ID of an item that should be dropped when the resource is destroyed. Before multipliers, this item is dropped in bunches of 3 to 7. Defaults to 0. Deprecated in favor of `Reward_ID`.

No_Debris *flag*: This resource does not have debris that should appear when it has been destroyed.

Reset *float*: Delay before respawning, in seconds.

Reward_ID *uint16*: ID of an item *spawn table* to use for rewards. Defaults to 0.

Reward_Min *byte*: Minimum amount of item drops to reward. Defaults to 6.

Reward_Max *byte*: Maximum amount of item drops to reward. Defaults to 9.

Reward_XP *uint32*: Amount of experience to reward when the resource is destroyed.

Scale *float*: The value of this property is always parsed as its absolute (positive) value. An object's scale is a random number between 1.1 and the result of `1.1 + (Scale * 2)`.

SpeedTree *flag*: This resource should be considered a SpeedTree when using higher graphical settings.

SpeedTree_Default_LOD_Weights *flag*: Use the default LOD weights intended for a SpeedTree.

Stick *uint16*: ID of an item that should be dropped when the resource is destroyed. Before multipliers, this item is dropped in bunches of 2 to 5. Defaults to 0. Deprecated in favor of `Reward_ID`.

Vertical_Offset *float*: A vertical offset above or below wherever this resource is placed, in meters. Defaults to -0.75.

Vulnerable_To_All_Melee_Weapons *bool*: When true, this resource can be damaged by melee weapons that do not have a corresponding `BladeID_#` value. Defaults to false.

Vulnerable_To_Fists *bool*: When true, this resource can be damaged by a player's fists. Defaults to false.

27.2 Localization

Name *string*: Resource name in user interfaces.

Interact *string*: Override the text shown for interactable resources using the `Forage` flag.

CHAPTER

TWENTYEIGHT

SPAWN ASSETS

The spawn asset type represents the weighted chances of an individual item, vehicle, or animal spawning at an any given spawn point. Create custom spawn tables that can be used on custom, curated, and official maps.

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Spawn`)

ID *uint16*: Must be a unique identifier. IDs 1–1000 are reserved for official content.

28.1 Tables

Tables are the assets spawned from the spawner, referenced by ID. These could be additional spawn tables; or individual assets like items, vehicles, and animals.

Tables: list of dictionaries. Each dictionary entry can contain these properties:

Guid *GUID*: GUID of an asset to spawn or a spawn table to recursively spawn a child from. Used if `LegacySpawnId` and `LegacyAssetId` are both unset or zero.

LegacySpawnId *uint16*: ID of a spawn table to recursively spawn a child from. Use of `Guid` is encouraged instead to prevent ID conflicts between mods.

LegacyAssetId *uint16*: ID of asset to spawn. Use of `Guid` is encouraged instead to prevent ID conflicts between mods.

Weight *int32*: Weight of this entry in the table.

For example, this configuration has a 90% chance of spawning a Military Magazine and a 10% chance of spawning an Eaglefire:

```
Tables
[
    {
        // Military Magazine
        Guid dbfb1d0d11ca438e9dffb95f76e61274
        Weight 180
    }
    {
        // Eaglefire
        Guid b03d581a5c1a490f995f8deba57b0f17
        Weight 20
    }
]
```

Note: This older format is used by most spawn assets. The newer format is recommended because it is more user-friendly, but the older format will continue to be supported.

Tables *int32*: Total number of children.

Table_#_Spawn_ID *uint16*: ID of a spawn table to recursively spawn a child from.

Table_#_Asset_ID *uint16*: ID of asset to spawn.

Table_#_Weight *int32*: Weight of this child in the table.

Table_#_GUID *GUID*: GUID of an asset to spawn or a spawn table to recursively spawn a child from. Used if `Spawn_ID` and `Asset_ID` are both unset or zero.

28.2 Roots

Roots are the spawners that your spawn table will attach itself to. This is useful when adding new spawn tables to preexisting spawn tables, such as those used by official maps. Tables are the things that will get spawned by your spawner. A spawner at the bottom of the chain will be entirely assetIDs, whereas one near the top will likely be entirely spawnIDs.

Roots: list of dictionaries. Each dictionary entry can contain these properties:

Guid *GUID*: GUID of parent spawn table. Used if `LegacySpawnId` is unset or zero.

LegacySpawnId *uint16*: ID of parent spawn table. Use of `Guid` is encouraged instead to prevent ID conflicts between mods.

IsOverride *bool*: If true, zeroes the weight of default spawns in the parent spawn table. Useful for mods intended to replace official content, such as with total conversions.

Weight *int32*: Weight of this entry in the parent spawn table.

Note: This older format is used by most spawn assets. The newer format is recommended because it is more user-friendly, but the older format will continue to be supported.

Roots *int32*: Total number of parents.

Root_#_Spawn_ID *uint16*: ID of parent spawn table.

Root_#_Override *flag*: Zeroes the weight of default spawns in the parent spawn table. Useful for mods intended to replace official content, such as with total conversions.

Root_#_Weight *int32*: Weight of this entry in the parent spawn table.

Root_#_GUID *GUID*: GUID of parent spawn table. Used if `Spawn_ID` is unset or zero.

CHAPTER

TWENTYNINE

STEREO SONG ASSETS

Defines a music track that can be played on the in-game stereo item. (Or any custom music player item for that matter.) For an example refer to `Unturned_Theme.asset` in the Songs folder.

29.1 Asset Properties Reference

Type *string*: `SDG.Unturned.StereoSongAsset`

Title string: display text to show in the music player menu. If a localization .dat file is present the `Name` key will be used, or a translation reference can be used. Examples:

```
"Title" "My song"
```

OR

Name in {Language}.dat file

OR

```
"Title"
{
	"Namespace" "SDG"
	"Token" "Stereo_Songs.Unturned_Theme.Title"
}
```

Song *Master Bundle Pointer*: audio clip to play. Can either be a newer master bundle pointer or an older content pointer. Examples:

```
"Song"
{
	"MasterBundle" "core.masterbundle"
	"AssetPath" "Effects/Ambience/Cave_0/Cave_0.ogg"
}
```

OR

```
"Song"
{
	"Name" "core.content"
	"Path" "assets/resources/bundles/songs/unturned_theme.mp3"
}
```

Link_URL *string*: Optional URL to open in web browser when external link button is clicked.

Is_Loop *bool*: Whether audio source should loop. Recommend **NOT** using .mp3 format for looping music.

CHAPTER THIRTY

UPGRADING FROM UNITY 2017 LTS TO 2018 LTS

30.1 Asset Bundles

Older .unity3d/.content/.masterbundle files should work properly without needing any update unless they use custom shaders. The game automatically tries to consolidate their shaders with the latest versions during loading. Once re-exported, Asset_Bundle_Version can be set to 3 in MasterBundle.dat or individual .dat files to disable this shader consolidation step.

Some of the slower asset checks like finding missing meshes have been made optional. Running the game with the "-ValidateAssets" command-line option enables them, and is recommended while working on new content.

30.2 Unity Packages

All example content has been updated for 2018 LTS, and now has a consistent export process to ensure the contents are kept valid. What were once individual packages (e.g. All_Shaders.unitypackage) have been merged into a single ExampleAssets.unitypackage in the Extras/Sources/Examples directory.

30.3 Logging / Server Console

Usage of Unity's built-in Debug.Log has been replaced with logging to the Client.log or Server_XYZ.log files in the Logs folder. This works around conflict with standard output on the Linux server, so -logfile redirect workarounds should no longer be necessary. -ThreadedConsole implementation has been made the default, but can be disabled by -LegacyConsole.

30.4 Workshop

Uploads from 2018 LTS are incompatible with past versions of the game, and a warning message is shown when loading newer content in the 2017 LTS version.

30.5 2017 LTS Availability

For archival purposes the 2017 LTS version of the game will remain in a “unity-2017” beta branch.

30.6 Platforms

Linux 32-bit and MacOS 32-bit have been removed in favor of the 64-bit versions. Servers that were still using the outdated Linux 32-bit depot should update to the 64-bit Linux dedicated server.

Headless server files have been removed from the player Linux depot, and are instead only in the dedicated server Linux depot. Windows headless mode is now supported in 2018 LTS, and is enabled for the Windows dedicated server depot.

CHAPTER

THIRTYONE

UPGRADING FROM UNITY 2018 LTS TO 2019 LTS

31.1 Blender Animations

Unity no longer supports importing multiple animations from a single .blend file. Exporting to an exchange format like .fbx is recommended instead, which is how the base game assets have always been handled. Note that this is recommended for meshes/models as well. Read more details on the Unity issue tracker here.

31.2 2018 LTS Availability

For archival purposes the 2018 LTS version of the game will remain in a “unity-2018” beta branch.

CHAPTER

THIRTYTWO

VEHICLE ASSETS

GUID *32-digit hexadecimal*: Refer to *GUID* documentation.

Type *enum* (`Vehicle`)

Rarity *enum* (`Common`, `Uncommon`, `Rare`, `Epic`, `Legendary`, `Mythical`): Rarity of the vehicle, as text shown in menus and colors used for highlights. Defaults to `Common` rarity.

Engine *enum* (`Blimp`, `Boat`, `Car`, `Helicopter`, `Plane`, `Train`): Defaults to `Car`. Some properties will only be usable depending on the `Engine` enumerator used, or may behave differently.

ID *uint16*: Must be a unique identifier.

32.1 Vehicle Properties

Bicycle *flag*: Player character should use a bicycling animation.

Bicycle_Anim_Speed *float*: Multiplier on the speed of the bicycling animation.

Bypass_Hash_Verification *flag*: Disable hash verification check, and allow for mismatched files.

Bypass_ID_Limit *flag*: Allows for using an `ID` value within the range reserved for official content.

Cam_Driver_Offset *float*: The vertical offset for the driver's first-person camera, in meters. This is additive with the value of `Cam_Passenger_Offset`.

Cam_Follow_Distance *float*: The distance behind the player the third-person camera should be placed at, in meters. Defaults to 5.5.

Cam_Passenger_Offset *float*: The vertical offset for any passenger's (including the driver's) first-person camera, in meters.

Can_Be_Locked *bool*: Whether or not the vehicle can be locked a player. Defaults to true.

Crawler *flag*: Disables the `Wheel_#` GameObjects from turning when steering by setting the default value of `Num_Steering_Tires` to 0. This property has no effect if `Num_Steering_Tires` has been manually set.

Drops_Table_ID *uint16*: ID of the item spawn table to use when the vehicle is destroyed. Defaults to 962.

Drops_Min *uint8*: Minimum amount of item drops to spawn when the vehicle is destroyed. Defaults to 3.

Drops_Max *uint8*: Maximum amount of item drops to spawn when the vehicle is destroyed. Defaults to 7.

Exit *float*: Distance away from the vehicle to teleport when exiting. Defaults to 2.

Has_Clip_Prefab *bool*: Whether or not the vehicle has a Clip.prefab. If the vehicle should use the same prefab on the server as on the client, set to false. For example, most official content uses `Has_Clip_Prefab false`. Defaults to true.

Has_Horn *bool*: Whether or not the vehicle should have a horn. Defaults to true when the vehicle either has a `Horn` AudioClip, or the `HornAudioClip` property has been set to a valid path. Otherwise, defaults to false.

HornAudioClip *Master Bundle Pointer*: AudioClip to play when using the horn.

IgnitionAudioClip *Master Bundle Pointer*: AudioClip to play when after entering the driver's seat.

LockMouse *flag*: First-person camera movement is locked while driving. This is useful for `Engine Plane` and `Engine Helicopter`, as a player's mouse movement while in first-person can be used to steer the vehicle.

Num_Steering_Tires *int32*: Total number of tires that should turn when steering. Defaults to 2 when using `Engine Car`, to 1 when using any other `Engine` enumerator, or to 0 if the `Crawler` property has been set.

Pitch_Drive *float*: Multiplier on the pitch of the engine audio while driving. Defaults to 0.03 when using `Engine Helicopter`, or to 0.1 when using `Engine Blimp`. For other `Engine` enumerators, it defaults to 0.025 if the audio clip is named "Engine_Large", or to 0.075 if the audio clip is named "Engine_Small".

Pitch_Idle *float*: Multiplier on the pitch of the engine audio while idle. Defaults to 0.625 if the audio clip is named "Engine_Large, or to 0.75 if the audio clip is named "Engine_Small".

Reclined *flag*: Player character should use a reclined idle animation.

Should_Spawn_Seat_Capsules *bool*: If true, capsule colliders will be attached to the `Seat` GameObject in order to prevent players from clipping into the ground. This is useful for vehicles that do not have a roof. Defaults to false.

Steering_Tire_# *int32*: Set a `Wheel_#` GameObject as a steering tire, which will visibly turn when steering. By default, a number of steering tires equal to the value of `Num_Steering_Tires` will be automatically set. These will start at `Steering_Tire_0 0` (corresponding to `Wheel_0`), and increment upwards.

Supports_Mobile_Buildables *flag*: Barricades and other buildables cannot be placed on this vehicle.

Tire_ID *uint16*: ID of the item that should given when a tire is manually removed with a *ToolAsset* that has `Mode Remove`, and can also be manually attached to the vehicle if the specified item ID is for a *ToolAsset* with `Mode Add`. Defaults to 1451.

Trunk_Storage_X *byte*: Number of columns (horizontal storage space). Defaults to 0.

Trunk_Storage_Y *byte*: Number of rows (vertical storage space). Defaults to 0.

Valid_Speed_Down *float*: Configuring this will override the sanity check for reversing speed, in m/s (meters per second). Defaults to 25 when using `Engine Car`, to 25 when using `Engine Boat`, or to 100 otherwise.

Valid_Speed_Horizontal *float*: Configuring this will override the sanity check for horizontal speed. This value is multiplied by PlayerInput.RATE (0.08), and then squared. Defaults to `(Speed_Max * 0.125)^2` when using `Engine Helicopter` or `Engine Blimp`, or to `(Speed_Max * 0.1)^2` otherwise. This property is useful for vehicles with speed that the server cannot predict, such as force-applying Unity components.

Valid_Speed_Up *float*: Configuring this will override the sanity check for forward speed, in m/s (meters per second). Defaults to 12.5 when using `Engine Car`, to 3.25 when using `Engine Boat`, or to 100 otherwise.

Zip *flag*: Player character should use a handlebar idle animation.

32.1.1 Handling

Air_Steer_Min *float*: The angle to turn when moving slowly, when using `Engine Plane`. Defaults to the value of `Steer_Min`.

Air_Steer_Max *float*: The angle to turn when moving quickly, when using `Engine Plane`. Defaults to the value of `Steer_Max`.

Air_Turn_Responsiveness *float*: Sensitivity on steering while airborne, when using `Engine Plane`. Defaults to 2.

Brake *float*: The amount of braking force to apply.

Center_Of_Mass_X *float*: Overrides the vehicle's center of mass on the -axis, when using `Override_Center_Of_Mass true`.

Center_Of_Mass_Y *float*: Overrides the vehicle's center of mass on the -axis, when using `Override_Center_Of_Mass true`.

Center_Of_Mass_Z *float*: Overrides the vehicle's center of mass on the -axis, when using `Override_Center_Of_Mass true`.

Lift *float*: The amount of upwards lift force to apply, when using `Engine Plane`.

Override_Center_Of_Mass *bool*: If true, override the vehicle's center of mass with the values from the `Center_Of_Mass_#` Vector3 properties. This allows for modifying a vehicle's center of gravity without needing to move the `Cog` GameObject in Unity.

Physics_Profile *GUID*: GUID of a *VehiclePhysicsProfileAsset* to use. Using a vehicle physics profile is optional. Defaults to `47258d0dcad14cb8be26e24c1ef3449e` when using `Engine Boat`, to `6b91a94f01b6472eaca31d9420ec2367` when using `Engine Car`, to `bb9f9f0204c4462ca7d976b87d1336d4` when using `Engine Helicopter`, or to `93a47d6d40454335b4784e803628ac54` when using `Engine Plane`.

Sleds *flag*: Tires should easily roll. For example, most planes will use this property.

Speed_Min *float*: The vehicle's maximum reversing speed, in m/s (meters per second). In-game, a vehicle's speed is displayed as either kph (kilometers per hour) or mph (miles per hour). For example, a vehicle that uses `Speed_Min -7` will have a maximum reversing speed of 25.2 kph (15.66 mph).

Speed_Max *float*: The vehicle's maximum forward speed, in m/s (meters per second). For all `Engine` enumerators except for the `Train` enumerator, this value is multiplied by 1.25 because the vehicle adjusts wheel torque trying to match a specific speed. For example, a vehicle that uses `Speed_Max 12.5` and is using `Engine Car` will have a maximum forward speed of 56.25 kph (34.95 mph).

Steer_Min *float*: The angle to turn when moving slowly.

Steer_Max *float*: The angle to turn when moving quickly. This value is multiplied by 0.75.

Traction *flag*: Tires should have traction in snowy positions.

Wheel_Collider_Mass_Override *float*: Override the mass of the vehicle's Wheel Collider components. This allows for quickly modifying the mass of the wheel colliders without needing to rebundle the asset in Unity. If a vehicle has realistic mass, then it may be helpful to set this value to something exceptionally high (e.g., 500). Defaults to `null`.

32.1.2 Health

Bumper_Invulnerable *flag*: The vehicle cannot be damaged by collisions (such as with other vehicles, objects, placeables, or entities).

Bumper_Multiplier *float*: Multiplier on the value for detecting collisions. When less than 1, the vehicle must be moving at a higher speed to enter a collision. When greater than 1, the vehicle can enter a collision while moving at a lower speed. Defaults to 1.

Can_Repair_While_Seated *bool*: If true, a player can repair the vehicle while also sitting in it. Defaults to false.

Child_Explosion_Armor_Multiplier *float*: Multiplier on the damage taken by barricades and other buildables placed on the vehicle, by explosions. Defaults to 0.2.

Environment_Invulnerable *flag*: The vehicle cannot be damaged by animals, zombies, or boulders thrown by mega zombies.

Explosions_Invulnerable *flag*: The vehicle cannot be damaged by explosions.

Health *uint16*: Total health value. Defaults to 0.

Health_Min *uint16*: Maximum possible health to spawn with. Defaults to 0.

Health_Max *uint16*: Minimum possible health to spawn with. Defaults to 0.

Invulnerable *flag*: The vehicle cannot be damaged by lower-power *Weapon Assets* that do not have the `Invulnerable` flag.

Passenger_Explosion_Armor *float*: Multiplier on the damage taken by players sitting in the vehicle, by explosions. Defaults to 1.

Tires_Invulnerable *flag*: Tires cannot be damaged.

32.1.3 Fuel

Fuel *uint16*: Total fuel capacity. Defaults to 0.

Fuel_Burn_Rate *float*: The rate fuel burns at. Defaults to 2.05 when using `Engine Car`, or to 4.2 otherwise.

Fuel_Min *uint16*: Minimum possible fuel to spawn with. Defaults to 0.

Fuel_Max *uint16*: Minimum possible fuel to spawn with. Defaults to 0.

32.1.4 Battery

Battery_Burn_Rate *float*: The rate battery charge is consumed at. Defaults to 20.

Battery_Charge_Rate *float*: The rate battery charge is recharged at. Defaults to 20.

Battery_Powered *flag*: The vehicle does not use fuel. For example, this flag is useful for creating electric vehicles.

Battery_Spawn_Charge_Multiplier *float*: Multiplier on the battery charge a newly-spawned vehicle with a vehicle battery will have. Setting this to a number less than 1 will result in the vehicle spawning with less battery charge than normal. Defaults to 1.

BatteryMode_Driving *enum* (*EBatteryMode*): How the vehicle battery should behave when a player is driving it. Defaults to `Charge`.

BatteryMode_Empty *enum* (*EBatteryMode*): How the vehicle battery should behave when the vehicle is empty. Defaults to `None`.

BatteryMode_Headlights *enum* (*EBatteryMode*): How the vehicle battery should behave when the headlights are on. Defaults to `Burn`.

BatteryMode_Sirens *enum* (*EBatteryMode*): How the vehicle battery should behave when the siren is on. Defaults to `Burn`.

Can_Steal_Battery *bool*: Whether or not the vehicle battery can be removed from the vehicle by a player. Defaults to true.

Cannot_Spawn_With_Battery *flag*: The vehicle does not spawn with a vehicle battery.

Default_Battery *guid*: Battery item given to the player when a specific battery hasn't been manually installed yet. Defaults to the vanilla car battery (098b13be34a7411db7736b7f866ada69).

32.1.5 Stamina

Stamina_Boost *float*: When a value is specified, this property allows for using stamina to boost. The value specified is the multiplier on the speed a vehicle can go without using a stamina boost. For example, `Stamina_Boost 0.5` would only let vehicle move at 50% its maximum speed normally, but using stamina to boost would it reach its maximum speed. This property is often used with `Stamina_Powered`, but this is not required.

Stamina_Powered *flag*: The vehicle does not use fuel or a vehicle battery.

32.1.6 Explosion

Explosion *GUID* or *uint16*: GUID or legacy ID of *EffectAsset* to play when destroyed.

Explosion_Min_Force_X *float*: Minimum amount of force applied on the -axis when the vehicle explodes. Defaults to 0.

Explosion_Max_Force_X *float*: Maximum amount of force applied on the -axis when the vehicle explodes. Defaults to 0.

Explosion_Min_Force_Y *float*: Minimum amount of force applied on the -axis when the vehicle explodes. This property must be set in order to use other `Explosion_Min_Force_#` properties. Defaults to 1024.

Explosion_Max_Force_Y *float*: Maximum amount of force applied on the -axis when the vehicle explodes. This property must be set in order to use other `Explosion_Max_Force_#` properties. Defaults to 1024.

Explosion_Min_Force_Z *float*: Minimum amount of force applied on the -axis when the vehicle explodes. Defaults to 0.

Explosion_Max_Force_Z *float*: Maximum amount of force applied on the -axis when the vehicle explodes. Defaults to 0.

ShouldExplosionCauseDamage *bool*: If true, the explosion caused by the vehicle being destroyed will deal damage. Defaults to true if `Explosion` is specified.

ShouldExplosionBurnMaterials *bool*: If true, the materials of the vehicle's `Model_#` GameObjects will be tinted black when the vehicle is destroyed. Defaults to true if `Explosion` is specified.

32.2 Turret

Turrets *uint8*: Number of turrets on the vehicle. All of the other turret properties require that this property is set. Defaults to 0.

Turret_#_Seat_Index *uint8*: Which `Seat_#` GameObject the turret is usable from. Defaults to 0 (corresponding to `Seat_0`).

Turret_#_Item_ID *uint16*: ID of the item usable from the turret seat. This is often used with a *GunAsset* that has the `Turret` property, but any item can be used.

Turret_#_Yaw_Min *float*: Minimum allowed rotation of the turret through the azimuth, in degrees. If this is set to -360, it can rotate leftward forever.

Turret_#_Yaw_Max *float*: Maximum allowed rotation of the turret through the azimuth, in degrees. If this is set to 360, it can rotate rightward forever.

Turret_#_Pitch_Min *float*: Minimum allowed rotation of the turret through the elevation, in degrees.

Turret_#_Pitch_Max *float*: Maximum allowed rotation of the turret through the elevation, in degrees.

Turret_#_Ignore_Aim_Camera *flag*: Disable having the camera positioned at the `Aim` GameObject.

32.2.1 Train

These properties should be used with `Engine Train`.

Train_Car_Length *float*: The distance between each train car on the train, in meters.

Train_Track_Offset *float*: The offset the train car is above the track, in meters.

Train_Wheel_Offset *float*: The offset between the wheels, in meters.

32.2.2 Economy

Shared_Skin_Lookup_ID *uint16*: Share skins with another vehicle. Defaults to the value of `ID`.

Shared_Skin_Name *string*: When generating images, the image name will contain the value of this string instead of the vehicle's file name. Often used with `Shared_Skin_Lookup_ID`.

Size2_Z *float*: Orthogonal camera size for economy icons.

32.3 Localization

Name *string*: Vehicle name in user interfaces.

CHAPTER

THIRTYTHREE

VEHICLE PHYSICS PROFILE ASSETS

This asset exists to tune vehicle physics in bulk without rebuilding asset bundles, and exposes more control than the individual vehicle assets.

One of the goals introducing profiles is to improve the handling of vanilla wheeled vehicles. Feel free to experiment with the default profile, and propose changes to it.

33.1 How to test?

In 3.19.18.0 the **reload** command was introduced which can be used to reload specific assets or directories of assets while playing. Simply reload the physics profile and then respawn a vehicle. For example to reload the default profile:

```
/reload 6b91a94f01b6472eaca31d9420ec2367
```

33.2 How are vehicles assigned a profile?

Individual vehicle assets can specify a profile by setting **Physics_Profile** to the GUID.

If a profile is not set, and the vehicle prefab's root rigidbody has a mass of 1.0, and the wheel colliders also have masses of 1.0, the default profile at Bundles/Assets/VehiclePhysicsProfiles/DefaultProfile.asset is used.

33.3 Asset Properties Reference

Type *string*: `SDG.Unturned.VehiclePhysicsProfileAsset`

Root_Mass: If set, overrides vehicle rigidbody's mass.

Root_Mass_Multiplier: If set, multiplies vehicle rigidbody's mass.

Root_Drag_Multiplier: If set, multiplies vehicle rigidbody's positional drag force.

Root_Angular_Drag_Multiplier: If set, multiplies vehicle rigidbody's angular drag force.

Carjack_Force_Multiplier: If set, multiplies carjack item's force applied.

Wheel_Mass: If set, overrides wheel collider mass. Ignored if vehicle asset has Wheel_Collider_Mass_Override set.

Wheel_Mass_Multiplier: If set, multiplies wheel collider mass.

Wheel_Damping_Rate: If set, override wheel collider damping rate. Lower values accelerate faster.

Wheel_Suspension_Force: If set, overrides wheel collider suspension force.

Wheel_Suspension_Damper: If set, overrides wheel collider suspension damper.

Wheel_Stiffness_Traction_Multiplier: Wheel collider stiffness multiplier when driving in snow. Default: 0.25

Wheel_Friction_Sideways and **Wheel_Friction_Forward**:

- **Extremum_Slip**: If set, overrides friction curve extremum slip.
- **Extremum_Value**: If set, overrides friction curve extremum value.
- **Asymptote_Slip**: If set, overrides friction curve asymptote slip.
- **Asymptote_Value**: If set, overrides friction curve asymptote value.
- **Stiffness**: If set, overrides friction curve stiffness. Multiplies the extremum and asymptote values. Sideways default is 1.0 and forward default is 2.0.

Motor_Torque_Multiplier: Multiplies wheel collider motor torque which is usually driven by vehicle speed.

Motor_Torque_Clamp_Multiplier: Multiplies wheel collider motor torque when exceeding maximum speed. Default: 0.5

Brake_Torque_Multiplier: Multiplies wheel collider brake torque.

Brake_Torque_Traction_Multiplier: Multiplies wheel collider brake torque when driving in snow. Default: 0.5

Wheel_Drive_Model: `Front`, `Rear` or `All`. By default all cars are rear-wheel drive, but in the real world cars are front-wheel drive. This discrepancy is due to a poor understanding of cars when they were added to the game in 2014.

Wheel_Brake_Model: `Front`, `Rear` or `All`. Modern cars have all-wheel breaking, which is the default.

CHAPTER

THIRTYFOUR

WEATHER ASSETS

Overrides the built-in snow and rain weather with custom events. This is feature is a work-in-progress.

Random weather can be scheduled to occur naturally on a map with the `Weather_Types` property of the *Level Asset*.

34.1 How to test?

When a GUID is passed to the weather command it will start a custom weather event, and 0 can be used to end it.

```
/weather 819982d7a2b6453488a8c4c5d9efe67f
```

34.2 Properties Reference

Type *string*: `SDG.Unturned.WeatherAsset`

Volume_Mask *u32 Mask*: Only enabled while inside an ambience volume with non-zero bitwise AND result. Defaults to 0xFFFFFFFF.

Fade_In_Duration *float*: Seconds between weather event starting and reaching full intensity.

Fade_Out_Duration *float*: Seconds between weather event ending and reaching zero intensity.

Ambient_Audio_Clip *Master Bundle Pointer*: Audio clip to play globally. Volume matches intensity.

Override_Fog *bool*: Should fog configured in the lighting be overridden?

Override_Atmospheric_Fog *bool*: Should fog affect the skybox?

Shadow_Strength_Multiplier *float*: Directional light shadow strength multiplier.

Fog_Blend_Exponent *float*: Power applied to fog blending alpha.

Cloud_Blend_Exponent *float*: Power applied to cloud blending alpha.

Wind_Main *float*: Wind zone windMain value. Will be replaced by more configurable game-specific wind eventually.

Dawn, **Midday**, **Dusk** and **Midnight**: Refer to the "*Time of Day*" section.

Effects *array*: Refer to the "*Effects*" section.

Stamina_Per_Second *float*: Stamina +/- buff.

Health_Per_Second *float*: Health +/- buff.

Food_Per_Second *float*: Food +/- buff.

Water_Per_Second *float*: Water +/- buff.

Virus_Per_Second *float*: Virus +/- buff.

Has_Lightning *bool*: If true, lightning will be enabled for this weather type. In the future this should get cleaned up, but for now it is hardcoded for assigning a net id.

Min_Lightning_Interval *float*: Minimum seconds between lightning strikes.

Max_Lightning_Interval *float*: Maximum seconds between lightning strikes.

34.3 Time of Day Properties

Each of the four main times of day can override certain properties.

Fog_Color *struct*: Distance-based fog. Optionally overrides the skybox color. Refer to the "*Color*" section.

Fog_Density *float*: Functions similarly to fog intensity in ambiance volume. Value must be within [0, 1].

Cloud_Color *struct*: Inner body of cloud. Refer to the "*Color*" section.

Cloud_Rim_Color *struct*: Outer edge of cloud. More visible than inner color. Refer to the "*Color*" section.

Brightness_Multiplier *float*: All ambient lighting colors are multiplied by this.

34.4 Effect Properties

Multiple effects can be instantiated while the weather is active.

Prefab *Master Bundle Pointer*: Game object with a particle system. PlayOnAwake should be disabled. For effects tied to the view it may be helpful to change the culling mode to "Always Simulate".

Emission_Exponent *float*: Power applied to weather intensity multiplied by default constant rate over time.

Pitch *float*: X-axis rotation when `Rotate_Yaw_With_Wind` is enabled.

Translate_With_View *bool*: Should position in world-space match the camera? The built-in snow and rain move with the view. Position is zeroed when false. May be useful for transition effects like dust blowing into the map signaling the start of a sandstorm.

Rotate_Yaw_With_Wind *bool*: Should y-axis rotation match the wind direction? The built-in snow and rain rotate with wind.

34.5 Color Properties

Each color can use a custom override, or a color from the level editor lighting panel. Using a level color is primarily for rain and snow backwards compatibility.

Level_Enum *enum*: If set, then the RGB specified are multiplied by this color.

R, **G**, **B** *uint8*: Color channel values.

34.6 NPC Conditions

Global weather state and current weather intensity blend can be tested through NPC conditions. Refer to *Conditions* documentation for documentation.

CHAPTER

THIRTYFIVE

ZOMBIE DIFFICULTY ASSETS

Override the difficulty settings for zombies in a navmesh. For reference, official ZombieDifficulty.asset files can be found at `...\Steam\steamapps\common\Unturned\Bundles\Assets\Zombie_Difficulty`.

35.1 Properties Reference

Type *string*: `SDG.Unturned.ZombieDifficultyAsset`

Overrides_Spawn_Chance *bool*: Whether or not the spawn chance for zombies in the navmesh should be overridden by the values set in this asset. For example, it may be useful to set this to `false` if you *only* wanted to tweak properties not related to spawn chances, such as the damage thresholds for stuns. Defaults to true.

Mega_Stun_Threshold *int*: Override the damage threshold for a hit to cause a stun.

Normal_Stun_Threshold *int*: Override the damage threshold for a hit to cause a stun.

Allow_Horde_Beacon *bool*: Whether or not Horde Beacons can be placed in the navmesh. Defaults to true.

35.2 Spawn Chance Properties

Crawler_Chance *float*: Decimal-to-percent chance for the zombie to be a Crawler. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Sprinter_Chance *float*: Decimal-to-percent chance for the zombie to be a Sprinter. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Flanker_Chance *float*: Decimal-to-percent chance for the zombie to be a Flanker. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Burner_Chance *float*: Decimal-to-percent chance for the zombie to be a Burner. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Acid_Chance *float*: Decimal-to-percent chance for the zombie to be a Acid Spitter. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Boss_Electric_Chance *float*: Decimal-to-percent chance for the zombie to be a Lightningstrike Zombie Boss. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Boss_Wind_Chance *float*: Decimal-to-percent chance for the zombie to be a Groundpounder Zombie Boss. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Boss_Fire_Chance *float*: Decimal-to-percent chance for the zombie to be a Flamethrower Zombie Boss. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Spirit_Chance *float*: Decimal-to-percent chance for the zombie to be a Spirit. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

DL_Red_Volatile_Chance *float*: Decimal-to-percent chance for the zombie to be a Volatile. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

DL_Blue_Volatile_Chance *float*: Decimal-to-percent chance for the zombie to be a Blue Volatile. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Boss_Elver_Stomper_Chance *float*: Decimal-to-percent chance for the zombie to be a Stomper Zombie Boss. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

Boss_Kuwait_Chance *float*: Decimal-to-percent chance for the zombie to be an Evil Eye Zombie Boss. Defaults to 0. Requires `Overrides_Spawn_Chance` to be true.

CHAPTER

THIRTYSIX

CHARTS

The Charts.unity3d file determines the colors usable by a map's chart view. This file contains two 32×1px textures: "Height_Strip" and "Layer_Strip".

36.1 Height_Strip

Height_Strip is used for topographical colors. The leftmost pixel (0, 0) is used for water. Other pixels are sampled based on the height of the terrain, going from the lowest potential point (1, 0) to the highest potential point (31, 0).

Note: Objects can also be configured to sample from the "Water" pixel (0, 0) or the "Ground" pixel (20, 0) of the Height_Strip.

36.2 Layer_Strip

Layer_Strip is used when something obstructs the terrain, such as an object or tree. The pixel sampled depends on the type of obstruction. Objects can be configured to use a different part of the Layer_Strip than their default, with some pixels only being used by such objects.

- (0, 0): Concrete roads wider than 8 meters use this pixel. Usable by objects.
- (1, 0): Concrete roads where width is less than or equal to 8 meters use this pixel. Usable by objects.
- (2, 0): Usable by objects.
- (3, 0): Non-concrete roads. Usable by objects.
- (4, 0): Usable by objects.
- (14, 0): Resources.
- (15, 0): Large-type objects. Usable by objects.
- (16, 0): Medium-type objects. Usable by objects.

CHAPTER

THIRTYSEVEN

CURATED MAPS

Community-created maps that are officially linked from the game are considered **Curated Maps**. They help players find high-quality new experiences, and keep the game updated with fresh content. Modders can then earn money from their work on their hobby, and get their creations highlighted.

37.1 Eligibility for Curation

Even if a map is of substantial quality, it may not be eligible for curation. When considering eligibility, it is important that the map fits the vanilla style of the game.

Although not every map should be curated, it is important to recognize the amount of effort that modders have put into their projects. Depending on the direction of the project, non-vanilla assets should either be adjusted to fit the vanilla style, or alternatives to curation should be considered instead.

Vanilla-style maps are eligible for curation, whereas non-vanilla maps are eligible to be featured. To help support the creators, both are eligible for revenue sharing cosmetic content in the game's item store e.g. the Candyland map.

It is encouraged to still work on a project even if it is not eligible for curation. Creating a variety of Workshop content creates a valuable portfolio for future curation attempts, and may still benefit from being featured.

37.2 Menu Visibility

One key component of sharing a curated map, trending popular workshop item, or specifically featured map is spotlighting it on the main menu. In any of these cases an article is placed above the recent news and announcements showcasing the preview image, and an expandable description.

In-game the description supports the following BBCode tags:

- `[b]`
- `[i]`
- `[list]`
- `[*]`
- `[h1]`
- `[img]`
- `[url]`

Ideally descriptions are kept succinct, so separate discussion topics might be a better place for long sections like ID lists.

By default popular items are surfaced, but workshop items can be specifically featured to help bring attention to projects with a lot of effort put into them, and curated maps. This can be done for new releases or major updates. Updates can link to a separate URL for the release notes.

Additionally, curated maps are linked from the singleplayer curated maps list, and will inherit the main menu "new" or "updated" label.

37.3 Quality Assurance

The primary focus of this article at the time of posting is to begin laying out standards to help ensure top quality between maps. This will be an iterative process, and is open to collaboration and feedback from the community.

Map Variety: New maps should feel noticeably different in order to be a candidate for curation. This is a subjective rule, and is covered by factors like interesting progression, new items, new environments, new architecture styles, new buildings, etc.

Asset Validation: Running the game with the *-ValidateAssets* command-line flag should not produce any warnings or errors.

English Text: Having an English-speaking member of the mod team is recommended, and MoltonMontro has offered to help with English-related questions. Most importantly in this regard is proper punctuation and grammar: while native English speakers can easily read incorrect punctuation, it is very helpful for non-native readers. Ironically this paragraph probably has some punctuation errors.

Project Organization: To prevent unintended assets from being exported into asset bundles, convention is to separate the project files into Sources and MasterBundle directories. Hawaii is split between a directory called "HawaiiMaster-Bundle" in the project root, and a Sources directory which contains all of the .blend, .mb, .xcf, .psd, .ai, etc files. When exporting the asset bundle this ensures only game files like .fbx and .prefab are included.

Asset Duplication: If multiple objects share identical prefabs, or use vanilla content, asset bundle size can be improved by sharing the same prefabs. For an example of this refer to the vanilla notes using Bundle_Override_Path.

Water Reflections: When using water volumes, only one should have planar reflections enabled. These reflections require rendering the world a second time for each enabled volume, and are some of the most performance expensive effects in the game.

Overlapping Navmeshes: No bounds should overlap. Otherwise zombies will appear and disappear unexpectedly in multiplayer.

Visibility Overlay: The visibility menu in the editor sums the mesh complexity in each region. Ideally there should only be a few red zones.

Item Icons: Each item should have a proper icon in the inventory. One way to quickly preview the icon is to attach an orthographic camera in Unity, and tune the orthographic size before setting the size in the .dat file.

37.4 Age Appropriacy

All official and curated game content should be what is typically considered family-friendly. As such, there are various restrictions on the type of content that maps seeking curation are allowed implement.

37.4.1 Profanity

Curated maps should not contain profanity. Objects, story notes, NPC dialogue, quest descriptions, and other such text should be devoid of any profanity. Profanity is anything considered heavy or mild cursing, slurs, or strongly implied.

Occasionally, there may be situations where an alternative to traditional profanity is useful. Nonsense words such as *gosh*, *darn*, *dang*, *drats*, and *heck* are fine to be used. Soft implications—such as a cut-off at the end of dialogue or a lore note—can also be utilized, so long as they are carefully worded to be open to interpretation.

In some situations, it may be better to replace large swaths of text on content such as lore notes with *[REDACTED]* or *[UNINTELLIGIBLE]*. Such redactions should be open to interpretation as to what may have been intended, and make sense within the context of the source material. Redactions should not be treated as a method of censoring individual words.

37.4.2 Drugs, alcohol, and other substances

Explicit depictions of drugs, alcohol, and other substances is not allowed. Although, maps may implement more family-friendly content that is similar in idea, such as berry mixes instead of alcohol. Maps may also choose to implement content that merely has looser, more distant association. Such as vineyards, bottles, kegs, or distilleries.

37.4.3 Inappropriate content

The depiction of sexual content, or explicit references regarding sexual content, are prohibited in all forms.

37.5 Stockpile Preparation

Each curated map release is usually accompanied by a few cosmetics and skins in the game's item store. Royalties from the sales are shared with the mod team. Even if a project is not eligible for curation, it may be appropriate to have some support-the-creator items in the Stockpile. For example, the Stockpile bundles created for the Candyland map.

File Sharing: Ideally the items have been setup for use as clothes in-game, and then exported into a .unitypackage. This package will then be imported into the vanilla project.

Curated Workshop Item: Payment splits are handled by a hidden curated workshop item. Setting this up usually takes a few weeks for new contributors' bank and tax information to be processed.

Bundles: Two or three collections of sets with four to six items each. Bundles can either be a collection of loosely-related items, or a complete outfit. Outfit bundles should avoid having multiple items that take up the same cosmetic slot.

Mystery Boxes: Fifteen to twenty items of rare, epic, or legendary rarity. The box can be themed, but all of the items should be usable individually – avoiding things like a set of matching shirts and pants that cannot be easily mixed with other cosmetic pieces.

Playtime Drops: Ten to twenty items of uncommon rarity. Unlike mystery box contents, it is far more appropriate for playtime drops to have matching sets and simple recolors.

CHAPTER

THIRTYEIGHT

EDITOR ASSET REDIRECTORS

If many objects need to be replaced on a map the old object guid can be redirected to a new guid rather than manually replacing them. This works similarly to the automatic holiday object replacements, but applies while loading a map in the editor, and changes are kept when the map is saved.

The game looks for a file named "EditorAssetRedirectors.txt" in the Unturned folder. Empty lines and lines starting with "//" or "#" (comments) are ignored. The .txt file extension was chosen because it is the notepad default. Each line contains two guids separated by an arrow "->".

For example:

```
// Replace Boulder_00 with Boulder_01
6125b4de591b44359237f6d7191dd919 -> ee402fc9debe4f03bffb31a49eb04fb7

// Replace Grass_1 with Grass_France_1
9a9655656f704b3caf717cea5a3b3cc2 -> d6dc5cc36f43429da668525e6ad174da
```

CHAPTER THIRTYNINE

FAVORITE SEARCHES

The objects editor supports **Favorite Searches** which allows lists of objects to be quickly looked up.

Entering “fv:xyz” in the search bar loads xyz.txt from the game folder, and will match any of the lines in the file. Empty lines and lines starting with “//” (comments) are ignored. The .txt file extension was chosen because it is the notepad default.

For example this matches anything with “fire” in the name or Road Line Cap #1:

```
// Fire related props
Fire

// Specific road prop
Cap #1 Road Line
```

Recursive usage of filters is supported, so multiple favorite searches can be nested, or other filter types e.g. “Tunnel mb:core” includes tunnels from the vanilla objects.

CHAPTER

FORTY

LEVEL BATCHING

This article is intended for map developers and explains how to maximize draw call batching.

For background information on the purpose of batching:

- Optimizing draw calls (Unity docs)
- Texture atlas (Wikipedia)
- Static batching (Unity docs)

40.1 Enabling batching in your level

By default batching is disabled because some parts of the level may be incompatible (causing graphical bugs), the texture atlas might be too big, it might worsen performance, etc. Publishing your map with batching enabled is only recommended after double-checking each location in singleplayer. (batching is disabled in the level editor) You can test it by adding this property to your level's `Config.json` file:

```
"Batching_Version": 2
```

The purpose of the version number is to allow future improvements without potentially breaking existing maps. For example, if atlas generation is supported for more shaders in an update it might behave unexpectedly, so those shaders would be excluded on older versions.

40.2 Purpose of atlas generation

Using fewer unique materials is almost always better for performance. Combining materials which only differ in their texture allows them to benefit from static and dynamic batching. If you want you can manually create a texture atlas for your own meshes, but resizing requires updating all your mesh UVs, and is generally a hassle. Considering that most workshop maps use objects from a variety of different mod packs, atlas generation helps them all work together.

40.3 Materials eligible for atlas inclusion

Standard (Decalable) or Standard (Specular setup) (Decalable):

- Mode is Opaque
- Texture is unset, or is 128x128 or smaller with Clamp wrap mode
- All other material features are default

Custom/Card: supported for the automatically generated tree skybox models.

Custom/Foliage: default trees/bushes.

40.4 Excluding specific objects and resources from batching

If you know your asset is incompatible you can add this line to the .dat file:

```
Exclude_From_Level_Batching true
```

NPCs, decals, and speedtrees (when enabled) are excluded by default. This option may be useful for elaborate setups using Unity Event components. For example if an event moves the renderer transform or sets material parameters.

40.5 Finding renderers that could benefit from atlas inclusion

By default the game considers every renderer in objects and resources. You can enable logging for why each renderer is excluded with the `-LogLevelBatchingTextureAtlasExclusions` launch option. Inclusion in the atlas is beneficial to merge as many meshes as possible into as few static batches as possible, but ineligible renderers will use static batching regardless.

None of the messages logged are "errors" per se. It only explains why the game cannot (yet) atlas them. The most useful message for finding assets to modify is "Wrap Mode is not Clamp" because if the mesh does not require UVs outside the 0-1 square it can use `Clamp` wrap mode.

40.6 Validating UVs

When textures are merged into an atlas any meshes referencing them need their UV coordinates updated. If any UVs are outside the 0-1 square they will now be overlapping a completely different texture and appear incorrectly. You can use the `-ValidateLevelBatchingUVs` launch option to log any batched meshes with out-of-bounds UVs. For example this error with the vanilla chess board:

```
Mesh "Model_0" in renderer "Chess_0/Model_0" has UVs outside [0, 1] range (should be␣
↪excluded from level batching)
```

In the case of the chess board it was a mistake in the unwrapping which was then fixed, but in most cases this would suggest the mesh relies on `Wrap Mode` being `Repeat`.

40.7 Previewing renderers using atlas

You can visualize which renderers have been included in the texture atlas by loading singleplayer with the `-PreviewLevelBatchingTextureAtlas` launch option:

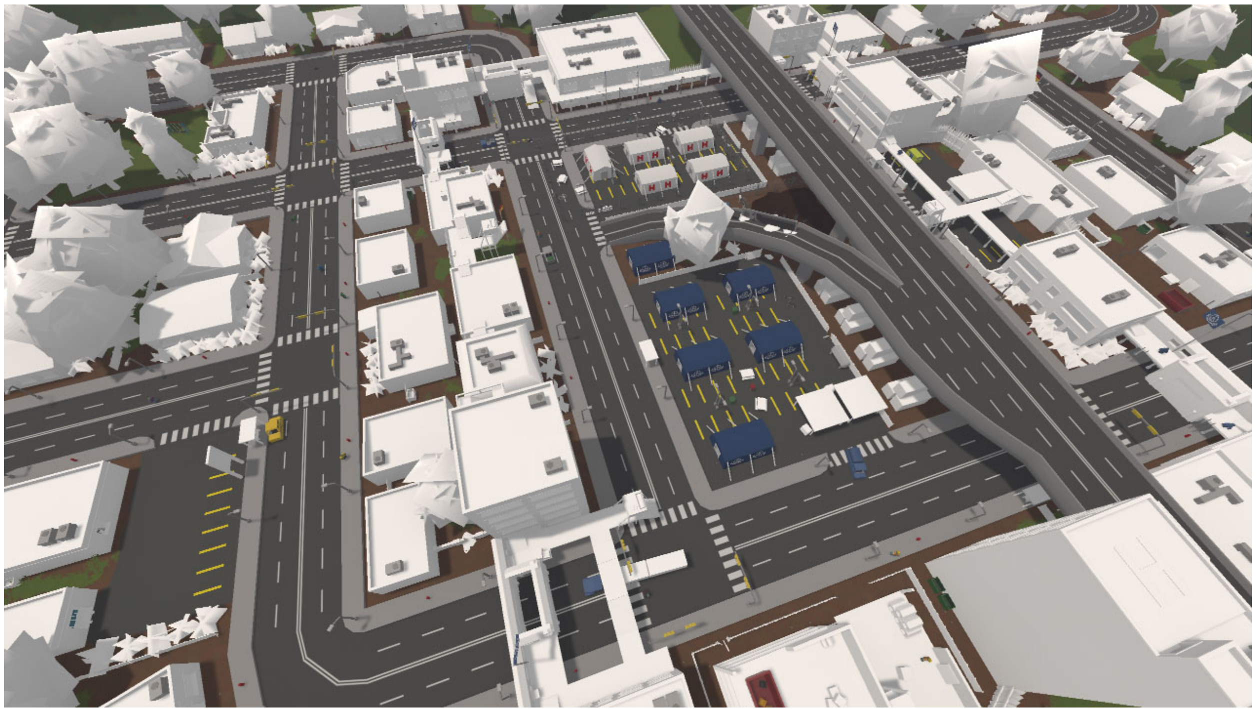


Fig. 1: Berlin with texture atlas preview enabled.

All renderers in white were merged into a single material per shader. It is not necessarily bad that some materials were not merged. For example, the HVAC units on the rooftops in the screenshot all share a material already, so they are able to use static batching together. The same goes for the roads and overpass.

CHAPTER

FORTYONE

LEVEL CONFIG

Each level is associated with an optional Config.json file. These files were originally added to make it easy to provide extra information about the level on the main menus, but grew over time to include gameplay parameters as well. In the future some of these might be moved to a more appropriate file like a level asset.

41.1 Main Menus

Creators *string[]*: Names in credits.

Collaborators *string[]*: Names in credits.

Thanks *string[]*: Names in credits.

Associated_Stockpile_Items *int[]*: Economy itemdefids to feature on map screens. One is chosen at random each time the map is shown. Used by curated maps to link their items which have payment splits.

Feedback *string*: URL to discussions. If not explicitly set, defaults to the workshop item's discussions page.

Visible_In_Matchmaking *bool*: Should this map be listed in the matchmaking menu? Used to filter out test and demo maps.

Version *string*: #.#.#.# format version number. Vanilla version numbers use 3.Year.Update.Patch, but that is optional. Incrementing the version number for every upload is good practice because:

1. When client and server files do not match it is more helpful to show a version number error message rather than a generic file mismatch error.
2. Searching by map in the server browser can filter servers running the same version of the map.

Tips *int*: Number of Tip_# keys defined in level's localization files, if any. Overrides vanilla tip messages on the loading screen.

41.2 Arena Mode

Use_Arena_Compactor *bool*: Should circles be randomized periodically?

Arena_Loadouts: Array of items to grant when spawning into arena. Each entry has a Table_ID spawn table to generate from, and an Amount number of times to grant from spawn table. For example:

```
"Arena_Loadouts":
[
        {
                "Table_ID": 28007,
```

(continues on next page)

(continued from previous page)

```
                "Amount": 1
        },
        {
                "Table_ID": 28008,
                "Amount": 1
        }
]
```

41.3 General

Asset: Object with GUID of *Level Asset* to instantiate on this map. For example:

```
"Asset": { "GUID": "12dc9fdbe9974022afd21158ad54b76a" }
```

Trains: Array of train vehicles to spawn. Only one of each train asset can exist at a given time because the vehicle ID is used to match saved trains to tracks. Road index can be seen by selecting a road in the level editor. Placement is normalized between the start and end of the track length. For example:

```
"Trains":
[
        {
                "VehicleID": 187,
                "RoadIndex": 0,
                "Min_Spawn_Placement": 0.1,
                "Max_Spawn_Placement": 0.9
        }
]
```

Mode_Config_Overrides: Pairs of server config properties and values to override them. For example:

```
"Mode_Config_Overrides":
{
        "Zombies.Min_Drops": 5,
        "Zombies.Max_Drops": 10,
        "Vehicles.Armor_Multiplier": 0.1,
        "Gameplay.Allow_Shoulder_Camera": false
}
```

Allow_Underwater_Features *bool*: Should legacy details and navigation bounds be restricted underwater?

Terrain_Snow_Sparkle *bool*: Should IS_SNOWING shader keyword be enabled?

Use_Legacy_Clip_Borders *bool*: Should invisible walls matching map size be created? Defaults to true.

Use_Legacy_Ground *bool*: Should default terrain be created? Alternative is to use devkit landscape tiles. Defaults to true.

Use_Legacy_Water *bool*: Should global water plane be enabled? Alternative is to use water volumes in devkit. Defaults to true.

Use_Vanilla_Bubbles *bool*: Should vanilla water bubble effects be enabled? Defaults to true.

Use_Legacy_Snow_Height *bool*: Should travelling vertically past snow height threshold enable snow effects? Defaults to true.

Use_Legacy_Oxygen_Height *bool*: Should travelling vertically past a certain point deplete oxygen? Defaults to true.

Use_Rain_Volumes *bool*: Should rain flag in ambiance volume be used?

Use_Snow_Volumes *bool*: Should snow flag in ambiance volume be used?

Use_Underground_Whitelist *bool*: Should underground players not inside a whitelist volume be teleported to the terrain surface? Useful to curb out-of-bounds exploits.

Is_Aurora_Borealis_Visible *bool*: Should aurora borealis effects be enabled?

Snow_Affects_Temperature *bool*: Should snow inflict cold damage?

Weather_Override *ELevelWeatherOverride*: Can be set to rain or snow to lock weather type.

Has_Atmosphere *bool*: If false, disable stars in skybox.

Has_Global_Electricity *bool*: Should all powerable items and objects have power by default?

Gravity *float*: Acceleration of gravity. Defaults to -9.81.

Blimp_Altitude *float*: Height override for blimp buoyancy. Defaults to 150.

Max_Walkable_Slope *float*: Steepest ground angle players can walk without sliding. Defaults to 59.

Prevent_Building_Near_Spawnpoint_Radius *float*: Closest distance players can build to spawn points. Useful to override for close-quarters maps. Defaults to 16.

Spawn_Loadouts: Array of items to grant when spawning in any mode. Refer to `Arena_Loadouts`.

Allow_Holiday_Redirects *bool*: Whether certain assets like objects, trees and landscapes should load alternative versions during holiday events.

41.4 HUD

Disable various elements of the heads-up display.

PlayerUI_HealthVisible *bool*

PlayerUI_FoodVisible *bool*

PlayerUI_WaterVisible *bool*

PlayerUI_VirusVisible *bool*

PlayerUI_StaminaVisible *bool*

PlayerUI_OxygenVisible *bool*

PlayerUI_GunVisible *bool*

Allow_Crafting *bool*

Allow_Skills *bool*

Allow_Information *bool*

41.5 Deprecated

Can_Use_Bundles *bool*: Used in the past for timed curated maps to disable using their assets in the level editor which could break after moving the map from the vanilla content to the workshop.

Category *ESingleplayerMapCategory*: Mostly automated now. Can be set to Misc to explicitly show in the miscellaneous map category.

Has_Discord_Rich_Presence *bool*: Only valid for official maps. If discord integration is enabled and this flag is true discord will check for a map icon configured in their partner page.

Item *int*: Kept for backwards compatibility. Ignored if `Associated_Stockpile_Items` are set.

Load_From_Resources *bool*: Used in the past for curated maps with assets in the vanilla Resources/Bundles/* directory. Master Bundles completely replaced this.

Should_Verify_Objects_Hash *bool*: With the newer asset integrity checks this is obsolete because each object/tree used in the level is checked with the server, and ignored if the server is missing the asset. Trees.dat and Objects.dat can always be included because missing assets do not factor into those hashes anymore.

Use_Legacy_Fog_Height *bool*: Should default terrain height be used for fog falloff? If false, devkit landscape tile limits are used instead. Defaults to true.

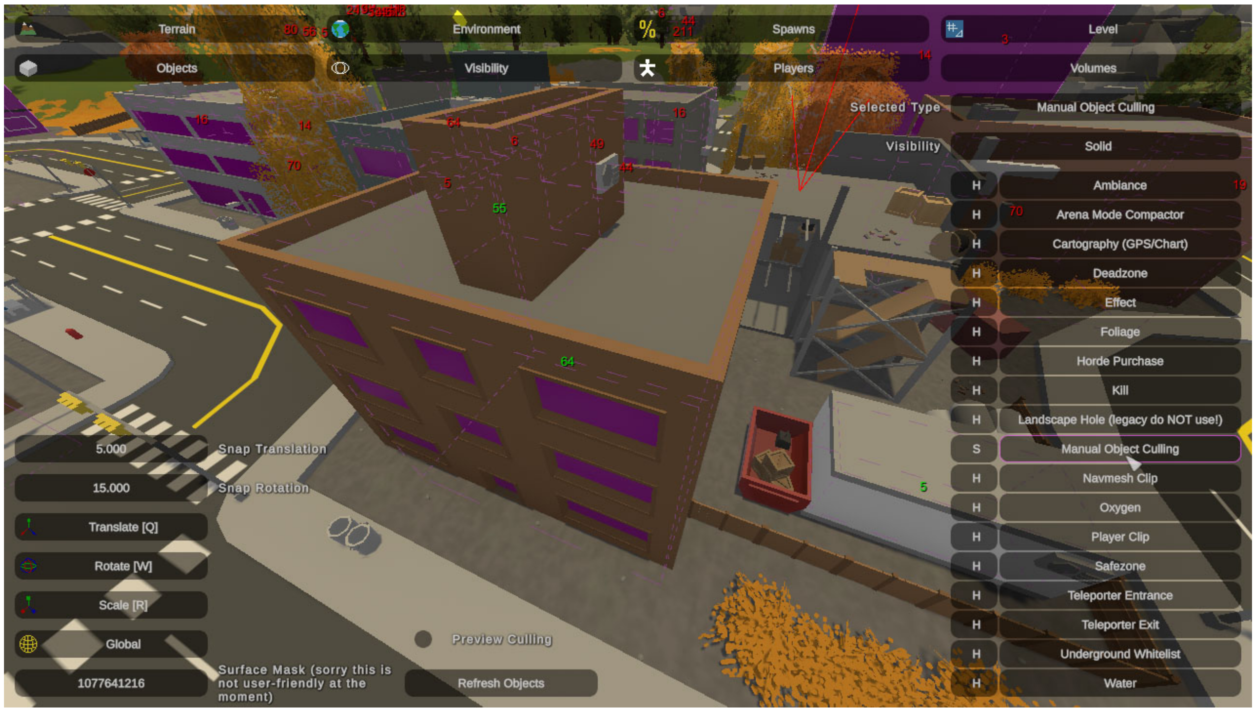
Use_Legacy_Objects *bool*: Should objects be loaded from Objects.dat file? Devkit objects were moved into this file, so this option no longer has any effect.

CHAPTER
FORTYTWO

MANUAL OBJECT CULLING

This article is intended for map developers and explains **Manual Object Culling** volumes.

Drawing fewer objects is usually better for performance. Culling volumes allow you to override the distance at which the contained objects are drawn where otherwise they might be drawn much further away than necessary. For example, by default the vanilla chair models are visible from quite far away in case they are placed outdoors, whereas inside an office building they only need to be seen while near the building.



Selected Type: Manual Object Culling
Visibility: Solid
Snap Translation: 5.000
Snap Rotation: 15.000
Surface Mask (sorry this is not user-friendly at the moment): 1077641216

Fig. 1: A building in Moscow with culling volumes.

This comes with an obvious downside: when zooming in on most buildings it is readily apparent that the furniture is missing. I experimented with some workarounds like enabling objects near the center of your view while zoomed, but it did not feel any better. In my opinion the performance trade-off is worth it. “Large” objects like shipping containers that have higher gameplay importance as cover are excluded by default, and most buildings have their culling volumes inset from the edges slightly so that objects in the windows are excluded.

42.1 Editing volumes

You can enable the **Preview Culling** checkbox to hide all objects inside culling volumes. This is useful to find any objects that are not included when they should be. For example, while working on this update I realized the volumes in the vanilla cargo ships did not extend low enough to catch some of the furniture in the crew quarters.

Objects inside volumes are found when loading the level. While working in the editor you can click **Refresh Objects** to re-find all objects/volumes on the map.

Updating the culling volumes costs some performance, so a large number of small volumes may actually make performance worse. It is worth comparing though.

42.2 Excluding specific objects

If you know your asset should never be managed by culling volumes you can add this line to the .dat file:

```
Exclude_From_Culling_Volumes true
```

For example, the aerospace facility on Germany is excluded so that the manually placed culling volumes can hide large objects like shipping containers without accidentally hiding the giant structure. Note: volumes owned by objects automatically exclude their owner object.

42.3 Volumes owned by objects

Most vanilla buildings come with default culling volumes which are not selectable because they are not saved in the level. These are the precursor to the manually placeable culling volumes, and have been invisibly hiding objects since 2014! Part of the goal with the culling volumes was to finally make these viewable in the editor because they have caused a lot of confusion over the years, especially for modded objects created without knowing they were there.

This is an area for future improvement, so I would not necessary recommend for/against adding them to your own objects. The following options are very old and poorly named, but can be specified in your object's .dat file to automatically create a culling volume:

LOD *enum*: Can be set to `Mesh` or `Area`. `Mesh` uses the bounds of renderers, and `Area` uses the size of Occlusion Area components. Note the Occlusion Area component's do not have any special functionality. At the time it was a workaround to allow placement in Unity with an otherwise unused component.

LOD_Bias *float*: Multiplier for the default 64m culling distance. For example, 2 would be visible up to 128m.

LOD_Center_X, LOD_Center_Y, LOD_Center_Z *float*: Offsets volume position relative to object transform along each axis.

LOD_Size_X, LOD_Size_Y, LOD_Size_Z *float*: Offsets calculated volume size in Mesh mode. Many vanilla buildings with flat rooftops use a negative Z offset to exclude HVAC units placed on the roof.

42.4 Testing culling volume performance benefit

If you would like to check whether culling volumes are providing any benefit you can run the game with the `-DisableCullingVolumes` launch option. Dense areas like Seattle tend to have the most noticeable difference.

CHAPTER

FORTYTHREE

COMMAND IO

By default Unturned executes commands from console input, and logs information to console output. This can be overridden however, for example to interact with an external process or remote console.

To replace the vanilla implementation:

1. Create a class that implements the ICommandInputOutput interface.
2. Get the CommandWindow singleton from Dedicator.commandWindow.
3. Pass instance to CommandWindow.setIOHandler.
4. (Optional) Specify `-NoDefaultConsole` on the command-line to disable vanilla console.

CHAPTER

FORTYFOUR

DEBUGGING EXCEPTIONS

In release builds it can be difficult to narrow down exactly what's causing an exception. Thanks to DiFFoZ post here for sharing a technique to find the line number, summarized here for posterity:

1. Unity logs an IL offset in square brackets to the `Player.log` file. This is not included in the stack trace sent to the game for the `Client.log` file unfortunately. For example 0x003db in this log:

   ```
   at SDG.Unturned.ResourceSpawnpoint..ctor (System.Byte newType, System.
   ↪UInt16 newID, System.Guid newGuid, UnityEngine.Vector3 newPoint, System.
   ↪Boolean newGenerated, SDG.Unturned.NetId netId) [0x003db] in
   ↪<08e91a6d9e1d4bd5bf2e982fa4148205>:0
                                                                               ␣
   ↪                                                                           ␣
   ↪                                        ^^^^^^^^^
   ```

2. Open the related assembly (in this case `Assembly-CSharp.dll`) in a c# decompiler like DnSpy or ILSpy.
3. In ILSpy use `Search` (Ctrl+Shift+F) to find the method from the stack trace.
4. Change display mode to `IL with C#`.
5. Find related line, `IL_03db` in the 0x003db example case.

CHAPTER

FORTYFIVE

DEDICATED WORKSHOP UPDATE MONITOR

When hosting a server with content from the Steam Workshop it may be desirable to restart the server when updates are released by creators. In particular, maps have their own version numbers which the server list uses to test compatibility before players connect.

By default the game monitors for changes to the hosted map. When a change is detected it notifies players via chat and shuts down the server after a timer. This can be overridden however, for example to notify players with a custom modal prompt and perform a custom restart process.

To replace the vanilla implementation:

1. Create a class that implements the IDedicatedWorkshopUpdateMonitor interface, or create a subclass of DedicatedWorkshopUpdateMonitor and override as-needed.
2. Bind the DedicatedWorkshopUpdateMonitorFactory.onCreateForLevel event and return the custom instance.

CHAPTER

FORTYSIX

FAKE IP

Using a Steam **Fake IP** allows players to join your server by IP address without *port forwarding*.

Unlike *Server Codes*, a server using Fake IP can be visible on the Internet server list (still without port forwarding) as long as you set a *Login Token*.

To enable Fake IP, change `Use_FakeIP` from `false` to `true` in your server's `Config.json` file.

After startup, you can run the `CopyFakeIP` command from the server console to copy the IP and port to your clipboard. Pasting this into the Host field of the Connect menu automatically moves the port to the Port field.

Connecting by Fake IP utilizes Steam Datagram Relay (SDR). To quote that page: "Relaying the traffic protects your servers and players from DoS attack, because IP addresses are never revealed. All traffic you receive is authenticated, encrypted, and rate-limited. Furthermore, for a surprisingly high number of players, we can also find a faster route through our network, which actually improves player ping times."

The downside of Fake IP as opposed to port forwarding is that the address and port will change after every restart, so, for example, a domain name cannot be used without custom scripts.

CHAPTER

FORTYSEVEN

GAME SERVER LOGIN TOKENS

Unturned dedicated servers can be logged-in to Steam using a **Game Server Login Token** or **GSLT**. This provides a few benefits:

1. If using Server Codes to connect, your code will remain linked to your GSLT between sessions. Otherwise, each time you start the server you will need to send your friends the new code.
2. Servers without a GSLT are considered "anonymous" and are hidden from the Internet server list.
3. Steam tracks your Favorites and History lists by address and port, but with a GSLT it can automatically transfer them if the server details change. This should happen within approximately 24 hours. (Verified in issue #3980 thanks to CyberAndrii and joeymisfit, and on AlliedModders.)

47.1 Creating GSLTs

You can manually create GSLTs while logged in with your Steam account here: https://steamcommunity.com/dev/managegameservers

Use Unturned's app ID `304930`, and a memo to remind you which server the token is for.

47.2 Unturned Configuration

The GSLT can be set in one of two places depending on your preference:

- With the `Login_Token` property in each server's `Config.json` file under the `Browser` section.

OR

- Using the `GSLT` command during startup. This can be specified in the `Commands.dat` file or on the command-line.

47.3 Automating GSLTs

Valve provides an `IGameServersService` web API for managing GSLTs. Consult their documentation here: https://partner.steamgames.com/doc/webapi/IGameServersService

CHAPTER

FORTYEIGHT

GLAZIER

Unity (the game engine Unturned runs on) has three different incompatible UI systems:

1. IMGUI
2. uGUI
3. UIToolkit

Unturned has a feature nicknamed **Glazier** which abstracts the underlying UI system, allowing IMGUI, uGUI, or UIToolkit to be used.

uGUI is Unity's current recommended UI system, but unfortunately some players run into visual artifacts and flickering UI with it. In those cases enabling IMGUI is recommended.

48.1 Context

At its 2014 release into Early Access, Unturned used IMGUI, as it was Unity's only built-in UI system. For performance reasons, automatic layout was turned off in favor of manually specifying the position and size of UI elements.

uGUI support was introduced in late 2020 for players running into issues with IMGUI. Unfortunately, IMGUI support needed to be kept for players facing different problems. Knowing that Unity was working on UIToolkit as a potential replacement, automatic layout was held off until an abstraction ideally supporting all three could be implemented.

Upon updating to Unity 2021 LTS, runtime UIToolkit became available, and it could be integrated as a mostly functional Glazier option. With UIToolkit supported, automatic layout started being added to the game in the form of stats in item descriptions, multi-line text chat, and an updated main menu news feed.

48.2 IMGUI

You can opt to use Unity's legacy UI system, IMGUI, by enabling a command-line argument:

1. Right-click Unturned in your Steam library
2. Click "Properties…"
3. Click "Select Launch Options…"
4. Add "-Glazier=IMGUI" without quotes

Pros:

- Faster on some systems because it has less overhead (no layout, no gameobjects).

Cons:

- Certain features like multi-line chat, extended item descriptions, and the main menu news feed are not supported in IMGUI mode.
- Visual bugs (e.g. incorrect gamma) and input issues on both Mac and Linux.
- Slower on some systems due to increased garbage collection.
- Does not support layered interactive UI. Some menus like crafting and inventory selection use workarounds for this, and thus behave differently from their uGUI counterparts.
- Plugin UIs are sorted underneath the game UI i.e. plugin UI cannot overlay.
- Rich text does not fade out in chat.

48.3 uGUI

This is Unturned's current default UI system, so opting in is not necessary.

Pros:

- Faster on some systems because the UI is not rebuilt every frame.
- More user-friendly e.g. can drag items outside the inventory to drop them.
- Looks better e.g. nicer scaling on high DPI monitors, foreground color universally supported.
- Rich text fades out properly in chat.

Cons:

- Visual artifacts and flickering on some systems.
- Slower on some systems because it has more overhead (layout, gameobjects).

48.4 UIToolkit

Integration into Unturned is experimental. It's not ready to be the default yet. You can check it out with a command-line argument:

1. Right-click Unturned in your Steam library
2. Click "Properties…"
3. Click "Select Launch Options…"
4. Add "-Glazier=UIToolkit" without quotes

Cons:

- Scroll views have incorrect content size (for now). With IMGUI and uGUI it was possible to explicitly specify the content size, whereas UIToolkit tries to automatically calculate it from the content bounds. Unfortunately, many of Unturned's scroll views have content clipping outside the intended content area, and so they don't appear correctly. For example, the location labels on the map can intersect the content border.
- Text shadows and outlines are not as nice as with uGUI.

CHAPTER

FORTYNINE

OPENMOD

OpenMod is a spiritual successor to *Rocket <doc_servers_rocket>* developed by one of Rocket's original maintainers. It has its own plugin framework, but supports compatibility with existing Rocket plugins by integrating with RocketMod and LDM.

49.1 Installation

49.1.1 Installing OpenMod Using the RocketMod Installer Plugin (recommended)

1. Download the latest OpenMod Installer Plugin for RocketMod from the OpenMod.Installer.RocketMod repository.
2. Move it to the `/Rocket/Plugins` folder and restart your server.
3. Run `/openmod install` and follow the instructions.
4. Done! Now you can start installing plugins.

49.1.2 Installing OpenMod Manually

1. Download the latest OpenMod.Unturned.Module-vX.X.X.zip from openmod repository.
2. Copy the "OpenMod.Unturned" folder into the "Modules" folder inside the Unturned installation directory.
3. Start your server. The first start will take a while since OpenMod will download its core components.
4. Done! Now you can start installing plugins.

49.2 Plugins

You can find open-source OpenMod plugins here.

49.2.1 Resources

- GitHub Repository
- Documentation

49.2.2 RocketMod Compatibility

OpenMod can be installed side-by-side with RocketMod, so you can use both RocketMod and OpenMod without any issue. OpenMod does not aim to replace RocketMod but to work with it together instead.

CHAPTER

FIFTY

PORT FORWARDING

Note: After the 3.23.14.0 update, port forwarding is no longer necessary to make a server accessible over the Internet. It's only required if you want players to be able to join directly by IP and/or domain name. For more information on the new alternatives to port forwarding, please refer to *Server Codes* and *Fake IP*.

When hosting a server on a home network **port forwarding** is required in order to direct traffic to the host computer. One way to think of it is that when there are multiple devices (e.g. computers and phones) connected to the LAN, the outside Internet does not know which device is the Unturned server. In this case port forwarding specifies which LAN device is the host.

Two pieces of information: the port range and local device address are required prior to port forwarding, and are described in detail below.

50.1 Port Range

Each Unturned server uses two consecutive ports while running. The first is for server list queries, and the second for in-game traffic.

By default 27015 and 27016 are used. Setting a different value with the `Port` command uses that value and plus one. Recommended `Port` command settings are 27015 for the first server, 27017 for the second server, 27019 for the third server, so on and so forth.

50.2 Local Device Address

Forwarding the ports directs them to a LAN address, i.e. the computer hosting the server. To determine the local IP on Windows:

1. Press Windows + R to open the Run dialog.
2. Type `cmd` and press enter.
3. Type `ipconfig` in the command prompt and press enter.
4. Find the `Wireless LAN adapter Wi-Fi` or `Ethernet adapter Ethernet` header.
5. Look for the `IPv4 Address` value and make note of it. This is the local address to forward the ports to. It likely looks something like `192.168.0.6`.

50.3 How to Forward Ports

Instructions vary by router, but should be doable from the web browser without any extra tools or software. This third-party website has a thorough list of routers with simple steps for each model: https://portforward.com/router.htm

In general the steps are along the lines of:

1. Connect to router via web browser.
2. Login with home admin credentials.
3. Find Port Forwarding menu.
4. Find the option to add a new rule.
5. Name the new rule something related to Unturned for reference.
6. Input 27015 as the starting port(s) and 27016 as the ending port(s).

Note: On some routers it might not be possible to input multiple ports within a single rule. In that case multiple rules can be setup; one for each of the two ports.

7. Enable UDP protocol.
8. Set destination internal IP to the local host address.
9. Save the new rule.

CHAPTER
FIFTYONE

ROCKET

SDG maintains a fork of **Rocket** called the Legally Distinct Missile (or LDM) after the resignation of its original community team. Using this fork is recommended because it preserves compatibility, and has fixes for important legacy Rocket issues like multithreading exceptions and teleportation exploits.

51.1 Installation

The dedicated server includes the latest version, so an external download is not necessary: 1. Copy the Rocket.Unturned module from the game's Extras directory. 2. Paste it into the game's Modules directory.

51.2 Resources

Browse the source code for the maintained version: Legally Distinct Missile Repository.

fr34kyn01535 has listed all of the original plugins in a post to the /r/RocketMod subreddit: List of plugins from the old repository.

Following closure of the original forum the recommended sites for developer discussion are the /r/UnturnedLDM subreddit, SDG Forum, or the Steam Discussions.

The RocketMod organization on GitHub hosts several related archived projects: RocketMod (Abandoned)

51.3 History

On the 20th of December 2019 Sven Mawby "fr34kyn01535" and Enes Sadık Özbek "Trojaner" officially ceased maintenance of Rocket. They kindly released the source code under the MIT license. Read their full farewell statement here.

Following their resignation SDG forked the repository to continue maintenance in sync with the game.

On the 2nd of June 2020 fr34kyn01535 requested the fork be rebranded to help distance himself from the project.

CHAPTER

FIFTYTWO

SERVER CODES

By default, your friends can join your server over the Internet using its **Server Code** in the Connect menu without *port forwarding*.

The downside of Server Codes is they are incompatible with the pre-joining server info screen. The info screen uses Steam's A2S protocol, which can only be queried by IP, so joining by Server Code enters the server immediately. You can also enable *Fake IP* to work around this limitation.

After startup, you can run the `CopyServerCode` command from the server console to copy the code to your clipboard.

Connecting by Server Code utilizes Steam Datagram Relay (SDR). To quote that page: "Relaying the traffic protects your servers and players from DoS attack, because IP addresses are never revealed. All traffic you receive is authenticated, encrypted, and rate-limited. Furthermore, for a surprisingly high number of players, we can also find a faster route through our network, which actually improves player ping times."

Note: Without a *Login Token* the Server Code will change each time your server restarts.

CHAPTER

FIFTYTHREE

SERVER HOSTING

All multiplayer servers are hosted using the Unturned Dedicated Server tool (sometimes abbreviated to U3DS). This tool can either be installed and updated using Valve's SteamCMD tool, or (not recommended) managed through your Steam Library. Using SteamCMD is ideal and has several benefits, but is not strictly necessary. If you are not using SteamCMD, some of the documentation may not apply to you.

Multiplatform:

- *How to Install Server using SteamCMD*
- *How to Install Server without SteamCMD*
- *How to Configure Server*
- *How to Host Curated Maps*
- *How to Host Over Internet*
- *Port Forwarding*
- *Rocket*
- *Login Tokens*
- *Update Notifications*
- *Rules and Guidelines*
- *Server Codes*
- *Fake IP*

Windows:

- *How to Install SteamCMD*
- *How to Launch Server*
- Video Tutorial

Linux:

- *How to Install SteamCMD*
- *How to Launch Server*

53.1 How to Install SteamCMD on Windows

1. Download From Here
2. Extract the contents of the zip somewhere you can find it again.
3. Run `steamcmd.exe`

Continue to: How to Install Server using SteamCMD

53.2 How to Install SteamCMD on Linux

Installation on Linux varies by distribution and your admin preferences, so refer to Valve's Linux Documentation. Once downloaded, run the `steamcmd.sh` script.

Continue to: *How to Install Server using SteamCMD*

53.3 How to Install Server using SteamCMD

1. Login to Steam anonymously.

```
login anonymous
```

2. Download the Unturned Dedicated Server application.

```
app_update 1110390
```

Tip: This command can also be used to update the server

3. Close SteamCMD.

```
quit
```

4. Server files have been installed in the default app install directory, which is `./steamapps/common/U3DS/`.

Continue to: *How to Launch Server on Windows* or *How to Launch Server on Linux*

53.4 How to Install Server without SteamCMD

The Unturned Dedicated Server tool can be installed and updated from your Steam Library. The tool is considered its own application, and is managed separately from the Unturned game itself. There are a few issues unique to those installing the Unturned Dedicated Server tool without SteamCMD, which should be considered before setting up your server.

1. It is not possible to run multiple servers at once.
2. The tool uses the same executable name as the game, which means that if the game is closed while the server is running then Steam will think the game is still running. This can cause issues such as Steam refusing to launch the game until the server as closed.

With these considerations in mind, it is recommended to install the Unturned Dedicated Server using SteamCMD instead. For those interested in installing the Unturned Dedicated Server tool without SteamCMD, navigate to your Steam Library. When using the default application filters for the Steam Library, tools (such as for launching dedicated servers) are not be visible in your Library.

To install the tool from your Steam Library either search for “Unturned Dedicated Server” via the search filter, or enable the “Tools” application type filter so that tools are visible. Select the “Unturned Dedicated Server” application in your Steam Library, and click the “Install” button.

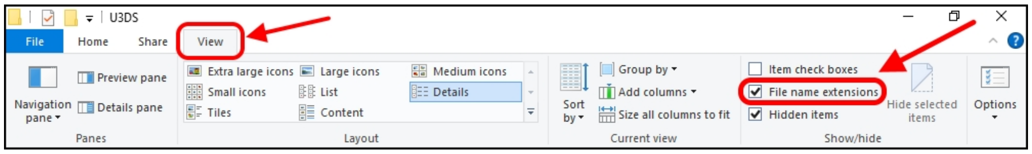
To navigate to the server files install directory: #. Right-click Unturned Dedicated Server in your Steam Library #. Select Properties… > Local Files > Browse…

The rest of the documentation assumes that the Unturned Dedicated Server tool was downloaded with SteamCMD, rather than through your Steam Library, so some of the documentation may differ slightly.

Continue to: *How to Launch Server on Windows* or *How to Launch Server on Linux*

53.5 How to Launch Server on Windows

1. Navigate to the `...\SteamCMD\steamapps\common\U3DS` directory.
2. Create a new text file by right-clicking an empty space within the U3DS directory, and selecting New > Text Document. This will create a new text file called “New Text Document.txt”.
   1. **If the file name does not display the `.txt` file extension, then you need to enable the viewing of “File name extensions”.**
   2. At the top of the File Explorer window, navigate to the View tab on the ribbon.
   3. In the Show/hide section of options, ensure that the “File name extensions” box is checked.

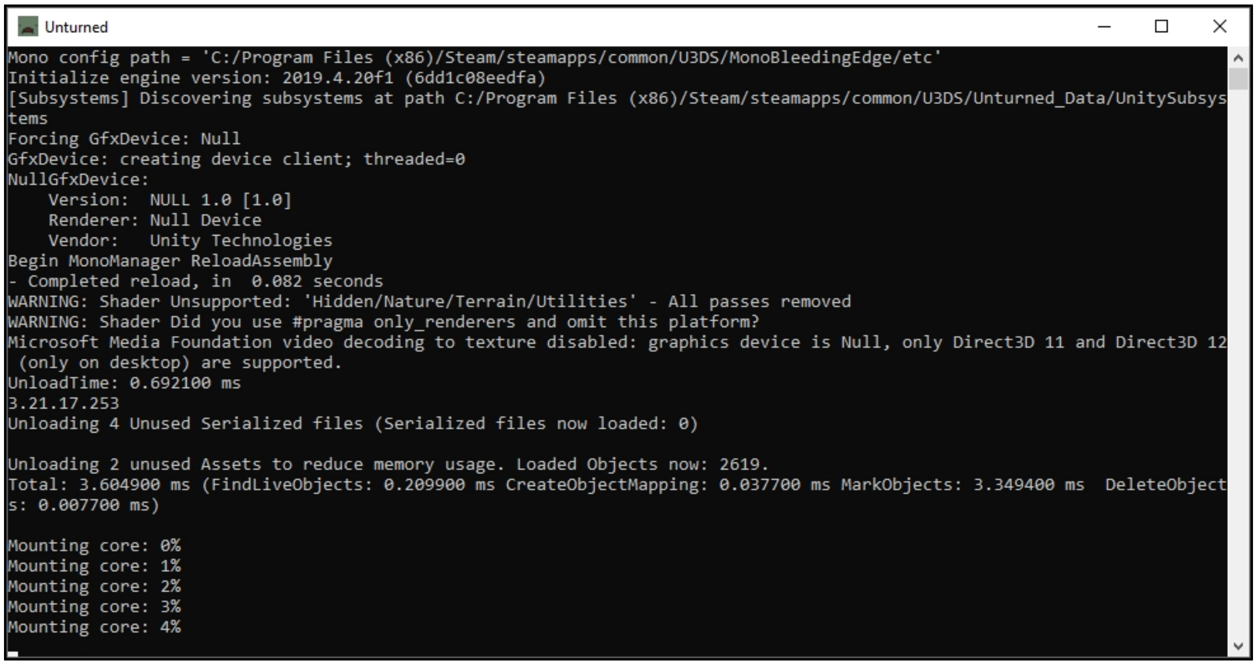


   4. File extensions should now be displayed at the end of file names.
3. Rename the “New Text Document.txt” file, and change it from a text file (.txt) to a batch script file (.bat). For example, “Tutorial.bat”.
4. Right-click on the batch script (`Tutorial.bat`) and select Edit. This will open the batch file in your default text editor, although any text editor (e.g., Notepad, WordPad, Notepad++) can be used.
5. Add the script that will start your server when the batch script is ran.

   a. For an Internet server, copy-and-paste the following text into the file: `start "" "%~dp0ServerHelper.bat" +InternetServer/MyServer`

   b. For a LAN server, copy-and-paste the following text into the file: `start "" "%~dp0ServerHelper.bat" +LanServer/MyServer`

   In this example “MyServer” is used as the ServerID for savedata and configuration purposes; you may choose to replace “MyServer” with a different name. For an example batch script, open the built-in `ExampleServer.bat` file in a text editor.
6. Save your changes to the file, and close the file.
7. Double-click the batch script to launch the server. A command-line interface should appear. Because this is the first time we have ran the batch file, it is going to generate a bunch of necessary server files.



```
Mono config path = 'C:/Program Files (x86)/Steam/steamapps/common/U3DS/MonoBleedingEdge/etc'
Initialize engine version: 2019.4.20f1 (6dd1c08eedfa)
[Subsystems] Discovering subsystems at path C:/Program Files (x86)/Steam/steamapps/common/U3DS/Unturned_Data/UnitySubsys
tems
Forcing GfxDevice: Null
GfxDevice: creating device client; threaded=0
NullGfxDevice:
    Version:  NULL 1.0 [1.0]
    Renderer: Null Device
    Vendor:   Unity Technologies
Begin MonoManager ReloadAssembly
- Completed reload, in  0.082 seconds
WARNING: Shader Unsupported: 'Hidden/Nature/Terrain/Utilities' - All passes removed
WARNING: Shader Did you use #pragma only_renderers and omit this platform?
Microsoft Media Foundation video decoding to texture disabled: graphics device is Null, only Direct3D 11 and Direct3D 12
 (only on desktop) are supported.
UnloadTime: 0.692100 ms
3.21.17.253
Unloading 4 Unused Serialized files (Serialized files now loaded: 0)

Unloading 2 unused Assets to reduce memory usage. Loaded Objects now: 2619.
Total: 3.604900 ms (FindLiveObjects: 0.209900 ms CreateObjectMapping: 0.037700 ms MarkObjects: 3.349400 ms  DeleteObject
s: 0.007700 ms)

Mounting core: 0%
Mounting core: 1%
Mounting core: 2%
Mounting core: 3%
Mounting core: 4%
```

8. When the command-line interface stops outputting new lines of text, it has finished loading (and finished generating all necessary files). You can safely close the server by executing (typing, and then pressing the " Enter" key on your keyboard) the following command on the command-line interface: `shutdown`
9. The batch script has created a new file directory located in `...\U3DS\Servers`, called "MyServer". This directory is where all the savedata and configuration files are kept. Changing the *MyServer* ServerID (from step 5) in the batch script to a different name will allow for keeping savedata separate across multiple servers, and for running multiple servers at once.
10. (optional) For your server to be visible on the in-game Internet server list you will need to set a *Login Token* and configure *Fake IP* or *Port Forwarding*.

53.6 How to Launch Server on Linux

1. Navigate to the `.../SteamCMD/steamapps/common\U3DS` directory.
2. To create our server, we need to execute a command.
   a. For an Internet server run the following command: `./ServerHelper.sh +InternetServer/MyServer`
   b. For a LAN server run the following command: `./ServerHelper.sh +LanServer/MyServer`

   In this example "MyServer" is used as the ServerID for savedata and configuration purposes; you may choose to replace "MyServer" with a different name. For an example script, open the built-in `ExampleServer.sh` file in a text editor.
3. You can safely close the server by executing (typing, and then pressing the " Enter" key on your keyboard) the following command on the command-line interface: `shutdown`
4. The executed command has created a new file directory located in `.../U3DS/Servers`, called "MyServer". This directory is where all the savedata and configuration files are kept. Changing the `MyServer` ServerID (from step 2) in the batch script to a different name will allow for keeping savedata separate across multiple servers, and for running multiple servers at once.
5. (optional) For your server to be visible on the in-game Internet server list you will need to set a *Login Token* and configure *Fake IP* or *Port Forwarding*.

53.7 How to Configure Server

Each individual ServerID has its own savedata and configuration.

1. Determine the ServerID. This is the name after the +InternetServer/ or +LanServer/ command.
2. Navigate to U3DS > Servers > ServerID.

Launch commands are setup in the Server > `Commands.dat` file. Each line should have one command.

Common useful commands are:

- **Map**: Specify the map to load by name, otherwise PEI is used.

Examples:

```
Map PEI
Map Washington
Map Russia
```

- **Port**: Running multiple servers simultaneously requires specifying different ports. Unturned uses two consecutive ports. The first is for server list queries, and the second for in-game traffic. Recommended port values are 27015 for the first server, 27017 for the second server, 27019 for the third server, so on and so forth.

Examples:

```
Port 27015
Port 27017
```

- **Name**: Name of the server on the server list; set as "Unturned" by default.
- **Password**: Requires password to join server. Note that password is only SHA1 hashed, so don't use the same password anywhere else.
- **Perspective**: Can be set to "First", "Third", "Both", or "Vehicle" to change camera options.
- **Cheats**: Allows admins to invoke cheat commands like spawning items or vehicles from the chat.

Game rules, listing display, and many other options are available in the `Config.json` file. Game options mirror the in-game Play > Singleplayer > Config menu. This file deserves further documentation, but is not officially documented yet.

Steam Workshop add-ons (e.g., maps, items, vehicles, collections) are setup in the `WorkshopDownloadConfig.json` file. To include a Workshop file on your server:

1. Browse to its web page, for example: Hawaii
2. Copy the file ID from the end of the URL.

```
URL: https://steamcommunity.com/sharedfiles/filedetails/?id=1753134636
ID: 1753134636
```

3. Insert the file ID into the File_IDs list:

```
"File_IDs":
[
        1753134636
],
```

Multiple file IDs should be separated by commas:

```
"File_IDs":
[
        1753134636,
        1702240229
],
```

4. During startup the files will be updated, and any dependencies detected. Players will have the files downloaded while connecting to the server.

53.7.1 How to Host Curated Maps

Curated maps are available as workshop items, so are configured in the `WorkshopDownloadConfig.json` file. During startup the Map command searches installed workshop items for a matching name.

Alphabetically sorted list of curated map file IDs:

- A6 Polaris: 2898548949
- Athens Arena: 1454125991
- Arid: 2683620106
- Belgium: 1727125581
- Buak: 3000549606
- Bunker Arena: 1257784170
- California: 1905768396
- Canyon Arena: 1850209768
- Carpat: 1497352180
- Cyprus Arena: 1647991167
- Cyprus Survival: 1647986053
- Dango: 1850228333
- Easter Island: 1983200271
- Elver: 2136497468
- France: 1975500516
- Greece: 1702240229
- Hawaii: 1753134636
- Ireland: 1411633953
- Kuwait: 2483365750
- Rio de Janeiro: 1821848824

53.8 How to Host Over Internet

By default, your friends can join your server over the Internet using its *Server Code* in the Connect menu without port forwarding.

For your server to be visible on the in-game Internet server list you will need to:

1. Set a *Login Token*.
2. Configure *Fake IP* or *Port Forwarding*.

Note: Without a *Login Token* the Server Code will change each time your server restarts.

CHAPTER

FIFTYFOUR

SERVER HOSTING RULES

Server hosts are able to customize their server in order to offer a tailored gameplay experience to their players. This often includes adding new features, disabling vanilla content, or using custom rulesets. However, all servers must adhere to the rules outlined below.

Servers that violate these rules may be temporarily or permanently banned. To report a server for rule violations, or appeal a moderation decision applied to your server, you may file a ticket with SDG Support:

- Report a server for breaking rules
- Appeal a server report

View Moderation List

54.1 Recent changes

2023-10-02: Clarified on how subscriptions interact with currency.

2023-08-17: Added reasoning behind flagging servers using an anycast proxy.

2023-02-15 revisions: Many of the rules have been revised to be clearer, with regards to what is (or isn't) currently allowed. "Consumable microtransaction" is better defined, there are a couple of new examples, and deceptive pricing has its own dedicated a section. The monetization filter section also includes more information about the filter, its purpose, and which of the four options (including a newer "Monetized" option) your server should use.

2022-10-16 clarification: Selling *vanilla* cosmetics, such as those available from the Stockpile or Steam Community Market, is not allowed. When offering cosmetics as a server microtransaction, the server network should should own (or have licensed) the rights to that content. Servers should not sell cosmetic content that they do not own the right to, such as vanilla cosmetics (either official, or community-contributed).

54.2 Monetization Types

Warnings for breaking the monetization rules first began being sent out on May 28, 2021. The monetization rules have now been in full effect since June 11, 2021.

Hosts are allowed to sell permanent benefits and monthly subscriptions. Consumable microtransactions are **not** allowed. A consumable microtransaction is anything that can be permanently consumed, lost, stolen, or destroyed. If it cannot be permanently lost, then it is not considered a consumable.

54.3 Monetization Examples

This section will provide *examples* of allowed/banned monetization options. It is not an exhaustive list of every possible monetization strategy.

54.3.1 Examples of allowed monetization:

- Accepting donations.
- Selling permanent or monthly subscription access to play on the server(s).
- Selling ranks, unlocks, benefits, etc. available permanently, or for the duration of the monthly subscription. Timers or cooldowns are fine.
- Selling reusable "kits"—which are reusable permanently, or for the duration of the monthly subscription—containing in-game items. Timers or cooldowns are fine.
- Selling **custom** cosmetics like custom skins, name tags, chat colors, etc. available permanently or for the duration of the monthly subscription.
- Selling services or commissions, for example a modder taking commissions for new content that gets added to the server for all players.

54.3.2 Examples of banned monetization:

- Selling individual in-game items like weapons, ammunition, supplies, bases, etc. that can be permanently lost, stolen, or destroyed.
- Selling individual in-game vehicles that can be permanently lost, stolen, or destroyed.
- Selling experience points.
- Selling currency. (As servers are allowed to have subscriptions, currency subscriptions are permitted so long as there is no more than 3 of these currency subscriptions available.)
- Selling ranks, kits, unlocks, benefits, etc. which stack with themselves as a loophole.
- Selling copies of **vanilla** cosmetics, such as those available on the Stockpile or Steam Community Market.

54.4 Monetization Filter

Players can filter the in-game server list by this field. It is not required to configure this field, but ideally it should be set to whichever value accurately describes your server's monetization practices. When configured, this field must be configured truthfully.

54.4.1 `Unspecified`

The “Monetization” field in each server’s Config.json file defaults to `Unspecified`. If you are unsure what to configure your server’s monetization type as, then you can leave it unspecified.

54.4.2 `None`

Servers that are entirely unmonetized, or only offer a donation option, can use the `None` value.

54.4.3 `NonGameplay`

Servers that only offer microtransactions that do not provide a gameplay advantage can use the `NonGameplay` value. For example, selling custom weapon skins and chat colors would not be a gameplay advantage.

54.4.4 `Monetized`

Servers that offer *any* “pay-to-win” microtransactions (i.e., those that provide a gameplay advantage)—such as selling “kits” containing items or vehicles—should use the `Monetized` option.

54.5 Deceptive Pricing

Fictitious and deceptive pricing is not allowed. For example: lying that a discount is nearly expired, or pretending the price is discounted when it has never been at the listed full price. We would strongly advise following Steam’s discounting rules to help avoid breaking any real-world laws.

54.6 Online Conduct

Repeated offenders of servers violating the Steam’s Online Conduct rules will be banned.

54.7 Anycast Proxies

If you use an anycast proxy, please consider submitting a support request here to ensure it is flagged correctly. Otherwise, players will almost certainly report the server.

Anycast proxies are a great protection mechanism, but they significantly affect the ping reported in the server browser. For example, a server hosted in Australia may have a ping of 40ms for players in the region but a ping of 300ms for players in Europe. Using an anycast proxy, in this case, would report a much lower ping (e.g., ~30ms) to players around the globe and incorrectly sort the server among those with the lowest ping.

This is frustrating for players looking for low-latency servers, as they may join one with a low ping only to find it is much higher in-game. Servers using an anycast proxy are flagged to sort like they have a higher ping to avoid this problem. It was implemented as a direct response to complaints from players.

Servers using a regular proxy with ping similar to the actual in-game ping are not flagged. Only proxies with a significant ping difference are flagged.

54.8 Workshop File Copyright Infringement

Mod authors can submit a notice of copyright infringement here: https://steamcommunity.com/dmca/create/

If you have submitted a notice of copyright infringement against a server host, please notify Smartly Dressed Games by email as well. We will keep a record of the server's workshop file usage. If there is a pattern of copyright infringement we will ban the server.

CHAPTER

FIFTYFIVE

SERVER UPDATE NOTIFICATIONS

Change logs are available on the Steam News Hub. For quick notifications with the game version number and Steam build ID consider one of the following:

55.1 Discord

Messages are posted to the #dedicated-server-updates text channel on the Official Discord Server via web hook. Each message has fields with the game version number and Steam build ID.

55.2 RSS

Each release is posted to an RSS feed on the Smartly Dressed Games website, with the game version number and Steam build ID: https://smartlydressedgames.com/rss/unturned-steam-dedicated-server-updates.xml

CHAPTER

FIFTYSIX

C# BUILT-IN TYPES

The following table lists the **C# built-in types**, along with common aliases and default values:

C# type	Alias(es)	Range	Default Value
bool	Boolean	`true` or `false`	`false`
int8	sbyte	`-128` to `127`	`0`
uint8	byte	`0` to `255`	`0`
int16	short	`-32768` to `32767`	`0`
uint16	ushort	`0` to `65535`	`0`
float32	float, single	precision up to seven digits	`0`
int32	int	`-2147483648` to `2147483647`	`0`
uint32	uint	`0` to `4294967295`	`0`
float64	double	precision up to fifteen digits	`0`
int64	long	`-9223372036854775808` to `9223372036854775807`	`0`
uint64	ulong	`0` to `18446744073709551615`	`0`
string		a sequence of zero or more Unicode characters	`null`

For more information about any of these built-in types, it is recommended to view Microsoft's "Built-in types (C# reference)" documentation page.

CHAPTER
FIFTYSEVEN

ASSET POINTER

When an asset refers to another it does so using an **Asset Pointer**. These identify the target asset by its *GUID*.

57.1 In *.dat files

Note that the GUID here is not wrapped in quotes because Unturned *.dat files are treated as pairs of strings.

```
MyAssetPtr ################################
```

57.2 In *.asset files

Two formats are supported in these files. The inline style was added later so you will see the older style used in many official assets.

```
"MyAssetPtr" "################################"
"MyAssetPtr" { "GUID" "################################" }
```

57.3 In JSON files

```
"MyAssetPtr": { "GUID": "################################" }
```

CHAPTER

FIFTYEIGHT

BITMASK

Recommended Wikipedia Article

58.1 Weather Example

Rain's default mask is 1 (0b01), and snow's default mask is 2 (0b10). An ambience volume could enable both with 3 (0b11), or neither with 0 (0b00).

CHAPTER

FIFTYNINE

COLOR

Colors can either be parsed as a single hexadecimal value (optionally prefixed with #), or from a dictionary with R, G, and B keys with *uint8* values.

For example, all three of these would be valid colors:

```
SkyColor 0000ff
GroundColor #00ff00
FogColor
{
        R 255
        G 0
        B 0
}
```

59.1 Legacy Parsing

Some older properties have support for the color data type, but can alternatively be written using the legacy method of three separate *float32* properties. Namely, the `Laser_Color` and `Nightvision_Color` properties have support for both.

For example, this would be valid for an older property that supports the legacy format:

```
Laser_Color_R 0.5
Laser_Color_G 1
Laser_Color_B 0
```

CHAPTER

SIXTY

ENUMERATED TYPES

60.1 EBatteryMode

The EBatteryMode enumerated type is used to determine how a vehicle battery should behave.

60.1.1 Enumerators

Named Value	Description
`None`	Battery charge remains unchanged.
`Burn`	Battery charge will deplete over time.
`Charge`	Battery charge will replenish over time.

60.2 EItemOrigin

The EItemOrigin enumerated type is used when spawning items.

60.2.1 Enumerators

Named Value	Description
`World`	Item origin is the world.
`Admin`	Item origin is an admin command.
`Craft`	Item origin is crafting.
`Nature`	Item origin is nature.

60.3 EItemRarity

The EItemRarity enumerated type consists of all the possible rarities of in-game content. Note that these are not the same as the item qualities used by Steam Economy items.

60.3.1 Enumerators

Named Value	Description
`Common`	Corresponds to the “Common” (white-colored) rarity.
`Uncommon`	Corresponds to the “Uncommon” (green-colored) rarity.
`Rare`	Corresponds to the “Rare” (blue-colored) rarity.
`Epic`	Corresponds to the “Epic” (purple-colored) rarity.
`Legendary`	Corresponds to the “Legendary” (pink-colored) rarity.
`Mythical`	Corresponds to the “Mythical” (red-colored) rarity.

60.4 EItemType

The EItemType enumerated type consists of all the possible item types.

60.4.1 Enumerators

Named Value	Description
`Hat`	Corresponds to the “Hat” item type.
`Pants`	Corresponds to the “Pants” item type.
`Shirt`	Corresponds to the “Shirt” item type.
`Mask`	Corresponds to the “Mask” item type.
`Backpack`	Corresponds to the “Backpack” item type.
`Vest`	Corresponds to the “Vest” item type.
`Glasses`	Corresponds to the “Glasses” item type.
`Gun`	Corresponds to the “Gun” item type.
`Sight`	Corresponds to the “Sight” item type.
`Tactical`	Corresponds to the “Tactical” item type.
`Grip`	Corresponds to the “Grip” item type.
`Barrel`	Corresponds to the “Barrel” item type.
`Magazine`	Corresponds to the “Magazine” item type.
`Food`	Corresponds to the “Food” item type.
`Water`	Corresponds to the “Water” item type.
`Medical`	Corresponds to the “Medical” item type.
`Melee`	Corresponds to the “Melee” item type.
`Fuel`	Corresponds to the “Fuel” item type.
`Tool`	Corresponds to the “Tool” item type.
`Barricade`	Corresponds to the “Barricade” item type.
`Storage`	Corresponds to the “Storage” item type.
`Beacon`	Corresponds to the “Beacon” item type.
`Farm`	Corresponds to the “Farm” item type.
`Trap`	Corresponds to the “Trap” item type.

continues on next page

Table 1 – continued from previous page

Named Value	Description
`Structure`	Corresponds to the “Structure” item type.
`Supply`	Corresponds to the “Supply” item type.
`Throwable`	Corresponds to the “Throwable” item type.
`Grower`	Corresponds to the “Grower” item type.
`Optic`	Corresponds to the “Optic” item type.
`Refill`	Corresponds to the “Refill” item type.
`Fisher`	Corresponds to the “Fisher” item type.
`Cloud`	Corresponds to the “Cloud” item type.
`Map`	Corresponds to the “Map” item type.
`Key`	Corresponds to the “Key” item type.
`Box`	Corresponds to the “Box” item type.
`Arrest_Start`	Corresponds to the “Arrest_Start” item type.
`Arrest_End`	Corresponds to the “Arrest_End” item type.
`Tank`	Corresponds to the “Tank” item type.
`Generator`	Corresponds to the “Generator” item type.
`Detonator`	Corresponds to the “Detonator” item type.
`Charge`	Corresponds to the “Charge” item type.
`Library`	Corresponds to the “Library” item type.
`Filter`	Corresponds to the “Filter” item type.
`Sentry`	Corresponds to the “Sentry” item type.
`Vehicle_Repair_Tool`	Corresponds to the “Vehicle_Repair_Tool” item type.
`Tire`	Corresponds to the “Tire” item type.
`Compass`	Corresponds to the “Compass” item type.
`Oil_Pump`	Corresponds to the “Oil_Pump” item type.

60.5 ELightingVision

The ELightingVision enumerated type is used to determine the lighting conditions when using certain items, such as *glasses*. Some assets may only support using specific enumerators.

60.5.1 Enumerators

Named Value	Description
`None`	There is no vision effect, and normal lighting is used.
`Military`	“Military” nightvision lighting is used. When supported by the asset, nightvision color is `#507814`, and nightvision fog intensity is `0.25`.
`Civilian`	“Civilian” nightvision lighting is used. When supported by the asset, nightvision color is `#666666`, and nightvision fog intensity is `0.5`.
`Headlamp`	“Headlamp” lighting is used. When supported by the asset, this will enable a toggleable light source and allow for using *PlayerSpotLightConfig* properties.

60.6 ENPCHoliday

The ENPCHoliday enumerated type consists of all of the game's recognized holidays or seasonal events. The duration of most seasonal events can be found in the `Status.json` file packaged with the game.

60.6.1 Enumerators

Named Value	Description
`None`	Represents no holiday or seasonal event.
`Halloween`	Corresponds to the Halloween holiday.
`Christmas`	Corresponds to the Christmas holiday, and other holidays within the festive season.
`April_Fools`	Corresponds to the April Fools' Day holiday.
`Valentines`	Corresponds to the Valentine's Day holiday.
`Pride_Month`	Corresponds to Pride Month, a month-long observance in June.
`Max`	Represents all holiday or seasonal events at the same time.

60.7 EObjectChart

The EObjectChart enumerated type is used to determine the how an asset should appear when generating a map's chart view. Most of the enumerators corresponds to a specific pixel coordinate on either the Height_Strip or Layer_Strip of a map's *Charts.unity3d file*.

60.7.1 Enumerators

Named Value	Description
`None`	Use the default for this asset.
`Ground`	Use (20, 0) from the Height_Strip.
`Ignore`	Skip this asset, and use whatever is underneath.
`Highway`	Use (0, 0) from the Layer_Strip.
`Street`	Use (1, 0) from the Layer_Strip.
`Road`	Use (2, 0) from the Layer_Strip.
`Path`	Use (3, 0) from the Layer_Strip.
`Large`	Use (15, 0) from the Layer_Strip.
`Medium`	Use (16, 0) from the Layer_Strip.
`Water`	Use (0, 0) from the Height_Strip.
`Cliff`	Use (4, 0) from the Layer_Strip.

60.8 ESlotType

The ESlotType enumerated type is used by equippable items. Some assets may only support using specific enumerators.

60.8.1 Enumerators

Named Value	Description
`None`	Does not correspond to any item slot.
`Primary`	Corresponds to the “Primary” item slot.
`Secondary`	Corresponds to the “Secondary” item slot.
`Tertiary`	Corresponds to the “Tertiary” item slot.
`Any`	Corresponds to any/all item slots.

CHAPTER

SIXTYONE

FLAG

Some asset types have properties that only look for the presence of a particular key, rather than a key-value pair. These are referred to as **flags**. Their value can be left empty, since flags do not have a value paired to them.

For example, item assets check for the Pro flag. An item asset that included this flag is marked as a Steam economy item.

```
Flag1
Flag2
Flag3
```

CHAPTER SIXTYTWO

GUID

Globally Unique Identifiers (**GUIDs**) are 32-digit hexadecimal values used to identify assets. They are preferable to file names because the files can be moved without redirectors.

A useful tool for manually generating GUIDs is guidgenerator.com. Note that if the GUID property is omitted from an *asset definition* file, then the game will automatically assign a random GUID during a successful load.

Legacy IDs are 16 bits with a [0, 65535] range, whereas GUIDs are 128 bits with an unimaginably huge range. This allows them to be generated without coordination or registration between developers.

MASTER BUNDLE POINTER

Identifies a Unity asset like a prefab, material or audio clip within a master bundle.

63.1 In *.asset files

`MasterBundle`: File name of the asset bundle exported from Unity. Should match the `Asset_Bundle_Name` configured in the `MasterBundle.dat` file.

`AssetPath`: File path of the Unity asset (e.g., *.prefab, *.mat, *.png, or *.ogg files). This path is relative to the `Asset_Prefix` path configured in the `MasterBundle.dat` file.

```
"MyMasterBundlePtr"
{
	"MasterBundle" "core.masterbundle"
	"AssetPath" "path/to/file.extension"
}
```

If the asset default master bundle should be used, then the path can be specified inline:

```
"MyMasterBundlePtr" "path/to/file.extension"
```

Alternatively, the name of a master bundle can be prefixed to the inline path:

```
"MyMasterBundlePtr" "core.masterbundle:path/to/file.extension"
```

CHAPTER

SIXTYFOUR

RICH TEXT

Certain text blocks support **rich text** markup. These are tags which modify the appearance of the text for example bold, italics, colors, etc.

Which tags are supported depends on the *Glazier* mode being used. Most players will be using the default, uGUI.

64.1 Extended Tags

These tags are specific to *Unturned*.

<br>: New line. Supported in most multi-line text boxes such as dialogue, signs/notes, item descriptions, etc.

<name_npc>: Insert the NPC character's name. Only supported in NPC dialogue.

<name_char>: Insert the player character's name. Only supported in NPC dialogue and signs/notes.

<pause>: Pause dialogue for 0.5 seconds before continuing. Only supported in NPC dialogue.

64.2 uGUI - TextMesh Pro (default)

For the full list of tags please refer to the TextMesh Pro documentation here: https://docs.unity3d.com/Packages/com.unity.textmeshpro@2.2/manual/RichText.html

64.3 IMGUI

For the full list of tags please refer to the Unity styled text documentation here: https://docs.unity3d.com/2018.3/Documentation/Manual/StyledText.html

CHAPTER

SIXTYFIVE

STRUCTS

65.1 PlayerSpotLightConfig

The PlayerSpotLightConfig struct contains properties for configuring player spot lights. Certain item assets are able to utilize these properties.

65.1.1 Properties

Property Name	Type	Default Value
SpotLight_Enabled	*bool*	`true`
SpotLight_Range	*float32*	`64`
SpotLight_Angle	*float32*	`90`
SpotLight_Intensity	*float32*	`1.3`
SpotLight_Color	*color*	`#f5df93`

65.1.2 Property Descriptions

SpotLight_Enabled bool `true`

When `true`, this item should have a toggleable light source.

SpotLight_Range float32 `64`

Range of the light source's beam, measured in meters.

SpotLight_Angle float32 `90`

Angle of the light source's beam, measured in degrees.

SpotLight_Intensity float32 `1.3`

Intensity of the light source's beam.

SpotLight_Color color `#f5df93`

Color of the light source's beam.

CHAPTER

SIXTYSIX

VECTOR3

Vector3 (or “3D vectors”) can either be parsed as a single value containing three *floats* (optionally surrounded by parenthesis), or from a dictionary with `X`, `Y`, and `Z` keys.

For example, all three of these would be valid 3D vectors:

```
Position 1, 2, 3
Offset (4, 5, 6)
Scale
{
        X 7
        Y 8
        Z 9
}
```

66.1 Legacy Parsing

Some older properties have support for the vector3 data type, but can alternatively be written using the legacy method of three separate *float32* properties. Namely, the `LOD_Center`, `LOD_Size`, `Explosion_Min_Force`, `Explosion_Max_Force`, and `Center_Of_Mass` properties have support for both.

For example, this would be valid for an older property that supports the legacy format:

```
LOD_Size_X 0
LOD_Size_Y -12
LOD_Size_Z -1
```

CHAPTER SIXTYSEVEN

UNTURNED DOCUMENTATION

This repository exists to document Unturned's modding features going forward. Originally map and asset documentation was created through Steam Guides. The main guide with links to related guides can be found here: Workshop: Creating, Publishing & Updating.

67.1 Upcoming Features

You can find modding features under consideration on this Trello board: Unturned Roadmap

The cards on the board aren't ordered in any particular way. i.e., they do not dictate the order of updates.

Miscellaneous requests and tasks that pop up take priority over the roadmap, so it may go a while between progress updates. For example, work for curated maps often takes priority.

Several high-priority ideas that don't yet have a solid plan, like car physics improvements or a crafting revamp, aren't listed on the board.

67.2 Video Tutorials

How to host a Dedicated Server on Windows

Several older tutorial videos are gradually becoming outdated and don't represent the current development direction, but are listed here for completeness:

- Uploading Skins to the Curated Workshop
- Creating Custom Songs
- Quick Overview of First Version of Foliage Upgrade
- Devkit 101 + Landscapes Overview
- Spawn Tables
- Building Models

67.3 Offline Downloads

PDF and Epub versions of the documentation can be downloaded for offline use.

67.4 Contributing

Anyone can contribute towards the *Unturned* modding documentation! To submit an issue, visit the GitHub repository. See the README for more details on how to contribute.