Unturned

Release 3.x

Smartly Dressed Games

Nov 12, 2026

ABOUT

1 Getting Started 1

2 Launch Options 5

3 Steam Workshop 9

4 Asset Bundles 11

5 Asset Definitions 15

6 Data File Format 17

7 Asset Validation 21

8 Asset Bundle Custom Data 23

9 Curated Items 25

10 Animation 31

11 Layers 33

12 Mod Hooks 37

13 Upgrading Unity Version 43

14 Introduction to Items 47

15 Blueprints 57

16 Actions 67

17 Arrest End Assets 71

18 Arrest Start Assets 73

19 Backpack Assets 75

20 Bag Assets 77

21 Barrel Assets 79

22 Barricade Assets 81

23 Beacon Assets 85

24 Blueprint Input Items 87

25 Blueprint Output Items 91

26 Box Assets 93

27 Caliber Assets 95

28 Charge Assets 99

29 Clothing Assets 101

30 Cloud Assets 105

31 Consumeable Assets 107

32 Detonator Assets 109

33 Farm Assets 111

34 Filter Assets 113

35 Fisher Assets 115

36 Fishing Catchable Properties 117

37 Food Assets 119

38 Fuel Assets 121

39 Gear Assets 123

40 Generator Assets 125

41 Glasses Assets 127

42 Grip Assets 129

43 Grower Assets 131

44 Gun Assets 133

45 Hat Assets 153

46 Key Assets 155

47 Library Assets 157

48 Magazine Assets 159

49 Map Assets 165

50 Mask Assets 167

51 Medical Assets 169

52 Melee Assets 171

53 Oil Pump Assets 173

54 Optic Assets 175

55 Pants Assets 177

56 Placeable Assets 179

57 Refill Assets 181

58 Sentry Assets 183

59 Shirt Assets 185

60 Sight Assets 187

61 Storage Assets 191

62 Structure Assets 193

63 Supply Assets 195

64 Tactical Assets 197

65 Tank Assets 201

66 Throwable Assets 203

67 Tire Assets 205

68 Tool Assets 207

69 Trap Assets 209

70 Vehicle Lockpick Tool Assets 211

71 Vehicle Paint Tool Assets 213

72 Vehicle Repair Tool Assets 215

73 Vest Assets 217

74 Water Assets 219

75 Weapon Assets 221

76 Vehicle Assets 225

77 Vehicle Physics Profile Assets 259

78 Vehicle Redirector Assets 261

79 Object Assets 263

80 Material Palette Assets 269

81 Introduction to NPCs 271

82 NPC Character Assets 273

83 Dialogue Assets 275
84 Quest Assets 277
85 Vendor Assets 279
86 Conditions 281
87 Rewards 287
88 Rewards List Assets 295
89 Currency Assets 297
90 Airdrop Assets 299
91 Animal Assets 301
92 Character Mesh Replacement 303
93 Crafting Assets 305
94 Crafting Blacklist Assets 307
95 Effect Assets 309
96 Foliage Assets 311
97 Road Assets 313
98 Landscape Material Asset 315
99 Level Assets 317
100 Mythical Effect Assets 323
101 Outfit Assets 325
102 Physics Material Assets 327
103 Redirector Assets 329
104 Resource Assets 331
105 Server Browser Curation Assets 333
106 Spawn Assets 335
107 Stereo Song Assets 339
108 Tag Assets 341
109 Weather Assets 343
110 Zombie Difficulty Assets 345
111 Charts 347
112 Curated Maps 349

113Editor Asset Redirectors 353

114Favorite Searches 355

115Level Batching 357

116Level Config 361

117Manual Object Culling 365

118Level Volumes 367

119Setting up a Server 371

120Using SteamCMD (Advanced Setup) 377

121Server Hosting Rules 381

122Bookmark Host 385

123Command IO 387

124Debugging Exceptions 389

125Dedicated Workshop Update Monitor 391

126Fake IP 393

127Game Server Login Tokens 395

128Glazier 397

129Modules 399

130NetMethods and NetIds 401

131OpenMod 407

132Port Forwarding 409

133Rocket 411

134Server Auto Restart 413

135Server Browser Curation 417

136Server Codes 423

137Server Configuration 425

138Server Update Notifications 427

139C# Built-in Types 429

140Asset Pointer 431

141Bitmask 433

142Color 435

143 Enumerated Types 437

144Flag 443

145GUID 445

146Master Bundle Pointer 447

147Rich Text 449

148 Structs 451

149Vector3 453

150FAQ 455

151Unity Project Overview 459

152Legacy ID Availability 461

153Upcoming Features 463

154Legacy Tutorials 465

155Offline Downloads 467

156Contributing 469

CHAPTER

ONE

GETTING STARTED

To get started with creating mods for *Unturned*, or hosting your own multiplayer server, certain tools need to be downloaded first. This page provides an explanation for the different types of tools you may need – depending on what you are trying to do.

1.1 Installing Unturned

Unturned must be downloaded in order to create, publish, and update mods. The game can be downloaded for free from Steam.

Not only do the game files include some of the tools necessary for creating your own custom content, but the game's official assets can also be used as an example when creating your own items, objects, or other game assets.

Adding *Unturned* to your Steam Library will also add the *Unturned Dedicated Server* application, which is necessary to host multiplayer servers. Although some server hosts may prefer to *download this tool through SteamCMD* instead.

1.2 Installing Unity

Installing the Unity Editor is required for exporting custom content for the game. We recommend using the same version as *Unturned*, which currently uses version **2022.3.62f3**. Unity can be downloaded from their website.

Although most 2022.3 LTS versions should be compatible, and some older LTS versions *can* be used with some additional setup, those versions might not function as intended.

When installing a Unity version, you will have the option to install some optional modules. You should at least install:

- **Linux Build Support (Mono)** – Adds support for your mod on Linux devices.
- **Mac Build Support (Mono)** – Adds support for your mod on macOS devices.

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.

1.3 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 alongside 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.

1.3.1 Project.unitypackage

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

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

1.3.2 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.

1.4 Other Tools

Modders will need a few more tools on hand to create custom content.

1.4.1 Text editors

A text editor (e.g., Notepad) or a code editor (e.g., Notepad++ or Visual Studio Code) is required to write the game data files used by assets. Code editors often include other useful tools, such as being able to search-and-replace content across multiple files at once.

We *do not* recommend using a word processor (e.g., Microsoft Word, LibreOffice, or WordPad). Such programs are not intended intended for writing plain text files, and it is easy to accidentally add unwanted characters when not used properly.

If you are unsure what you should use, Notepad comes installed on all Windows systems by default, and lacks any additional tools or features that (while helpful) may be confusing for an inexperienced user.

1.4.2 Image editing software

To create custom textures for your modded content – such as for new shirts or pants, materials for custom objects, or 2D effects – you will need an image editing software that supports transparency. Some free image editors include Paint.NET, GIMP, and Krita.

1.4.3 Blender

A 3D modeling tool such as Blender is required to create custom models (and animations). Blender is the same tool we use for *Unturned*, although it is not strictly required.

CHAPTER

TWO

LAUNCH OPTIONS

Launch options can be added to *Unturned* to change certain game settings before running the game. This allows for recovering from certain problems (such as an unwanted resolution or UI scale), troubleshooting a wide range of issues, or toggling settings not available from in-game.

This article lists the launch options available for *Unturned*. You can add launch options through your Steam Library.

1. Right-click **Unturned** in your Steam Library.
2. Select the **Properties...** button.
3. On the **General** tab, find the **Launch Options** field near the bottom.
4. Type options separated by spaces. For example, `-TimeOverlay -Width=1920 -Height=1080` will enable the TimeOverlay flag, set Width to 1920, and set Height to 1080.

2.1 Game Options

Some of the launch options are primarily intended for use with the Unturned Dedicated Server tool.

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

-Cinematic: Turns off many level-of-detail optimizations. This has a significant performance cost.

Effects include:

- 4 km draw distance and sun shadow range.
- LOD groups always show their highest quality. (High LOD bias.)
- Lights are always visible. (Turns off light LODs.)
- Objects and resources are always visible. (No landmarks or culling volumes.)
- Terrains always use splatmap shaders. (No basemap fallback texture.)
- Terrains always use max-quality heightmaps. (No mesh simplification.)
- Sun shadowmaps use GPU max supported resolution. (16,384 x 16,384 on some modern GPUs.)
- Planar reflections render at 100% resolution. (Rather than 50%.)

-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.

-EnableCharacterControllerOverlapRecovery: When enabled, `CharacterControllerExtension.CheckedMove` passes through to `CharacterController.Move`, and `CharacterController.enableOverlapRecovery = true`. Using this can improve performance – which can be useful for servers –

but it makes out-of-bounds exploits much easier. This can cause more problems than it solves so it's not enabled by default.

-EnableWheeledVehicleGizmos: Draw locally driven vehicle's wheel torque, RPM, slip, expected RPM, etc.

-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`, `UIToolkit`): Use a different UI system instead of the default uGUI. Accepted values are `IMGUI` (legacy) and `UIToolkit` (experimental). For more information, refer to: *Glazier*.

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

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

-Holiday= *enum* (`AprilFools`, `Christmas`, `Halloween`, `HW`, `PrideMonth`, `Valentines`, `XMAS`, `LunarNewYear`, `LNY`, `UnturnedAnniversary`): 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.

-LogBallisticDropConversion: Log automatic change of older gun assets' Ballistic_Drop property to Bullet_Gravity_Multiplier. Useful when manually updating older guns to the new property.

-LogGunSpreadConversion: Log automatic change of older gun assets' Spread_Hip property to Spread_Angle_Degrees. Useful when manually updating older guns to the new property.

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

-LogSpawnTablesAfterLoadingLevel: Log all spawn chances.

-LogVehicleWheelConfigurations: Log automatic creation of vehicle asset's `WheelConfigurations` property for older vehicles. Useful when converting vehicles to the new format.

-ModulesPath *string*: If set, search for `.dll` and `.module` files in this directory instead of in `Unturned/Modules`.

-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.

-NoPreserveMissingObjects: By default, the level editor keeps objects and foliage whose assets are missing. If this option is enabled, any missing assets are deleted instead.

-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.

-ParseAssetMetadata: Enables parsing asset file metadata like comments and line numbers. Useful for development (e.g., error messages) at the cost of slower loading and increased memory usage. Plugin developers building on this feature may be interested in doc_dat_editing_code.

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

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

-RazerChroma: Enable Razer Chroma integration on compatible devices.

-RefreshRate=: Monitor refresh rate override.

-ResaveAssets: Danger! Only use this if you have a backup of your custom assets, ideally in version control. Here be dragons:

Depends on the **-ParseAssetMetadata** launch option also being enabled.

This is our first experiment with automatically patching asset files. It will attempt to preserve comments and line numbers in your files (this is why asset metadata is needed). However, certain comments may not be preserved. In particular, comments with blank lines surrounding them.

At the time of writing (2025-05-06) the game will convert blueprints from the legacy (Blueprint_*** prefix) format to newer list-based format. It cannot yet auto-convert blueprints with NPC conditions or rewards because we haven't written conversion code for those yet, but it's on our wishlist.

-ResetSteamStatsAndAchievements: Reset all progress on Steam achievements and stats.

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

-ScrollViewSensitivity *float*: Multiplier for uGUI scroll view distance travelled when rolling the mouse wheel.

-SNS_DisableIPSocket: Skip Steam Networking Sockets creation of regular IP socket. Useful for limiting player connections to peer-to-peer (P2P).

-SNS_DisableP2PSocket: Skip Steam Networking Sockets creation of non-FakeIP P2P socket. Useful for limiting connections to only *Server Codes*.

-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`.

-UnlockSteamAchievements: Unlocks all Steam achievements. This is intended for achievement hunters who've moved on from the game but want to maintain their previous 100% completion status.

-UnredactedLogs: By default, player IPs in BattlEye's logging and the public IP for Workshop downloads are redacted. This option turns that off.

-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.

2.2 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

STEAM WORKSHOP

The **Steam Workshop** allows for sharing user-generated content (such as new maps, items, localizations, and other in-game content). This type of content is typically known as a *mod*, or *modification*. Players can download mods by clicking the Subscribe button on the detail page for the item.

Players can start exploring the Steam Workshop from its Home page, or learn more about the features utilized by *Unturned* from the About page.

3.1 Browsing

There are several tabs and additional filters in the Steam Workshop, to help players find the type of user-generated content they are interested in.

Content under the Mods tab can be downloaded by clicking the "Subscribe" button. This content will then be available the next time you launch the game. Some mods may conflict with each other, or may not be compatible with the latest version of the game. We recommend reading the description for each mod you are interested in, in case the mod author has included helpful information.

Content under the Stockpile Submissions tab cannot be downloaded. Instead, we may incorporate some of these submissions—such as new skins or cosmetics—into the actual game. Players can vote on whether they would like to see a specific submission added into the game. Accepted items may be made available for purchase, or otherwise be unlockable within the game.

The Collections tab contains groups of mods, which can be quickly and easily downloaded together. These are sometimes referred to as *modpacks*. For example, you may find a collection that adds several medieval-themed mods that are all compatible with each other.

3.2 Creating Mods

Refer to "*Getting Started*" for more information about what tools are needed to create mods. Afterwards, documentation for individual types of content can be found later on in the documentation.

3.3 Publishing Mods

When you are ready to publish your mod, launch the game and navigate to Workshop tab of the main menu. Click the "Submit" button to begin the submission process.

You will see the following fields when attempting to publish a mod:

1. **Name** – This will be displayed in the Steam Workshop, and some other locations, as the name of your mod.
2. **Collection Path** – Specify a valid file path to a folder containing your mod. For example, to upload a map you would copy your map's folder into a new folder, and then provide the path to that new folder in this field.

3. **Preview Image** – Specify a valid path to a .PNG or .JPG file, which will be used as the preview image for your mod. This is displayed in the Steam Workshop, and some other locations.
4. **Change Note** – This is an optional field. When a change note is included, that text will be added to the "Change Notes" tab of your mod's Workshop page. We recommend using this field when updating your mod.
5. **Asset Type** – Choose what type of mod you are uploading. Players will be able to filter for your mod based on its asset type. Some options have "sub-types", such as the specific item a skin is intended for.
6. **Visibility** – This defines who can view your mod once it has been published. You will usually set this to "Public", so that other players can see and download your mod. Although the other options can be helpful if you want to make a few changes before making the mod public.
7. **Allowed IPs** – This is an optional field. By including an IP address in this field, only the servers with that IP address will be able to automatically download and update the mod via the WorkshopDownloadConfig.json file available to dedicated servers. It is unlikely that you will want to use this field.
8. **Workshop Section** – Choose which section this mod should be published under. "Ready-to-Use" should be used for mods that other players can immediately download and enjoy. "Curated" should be used for skins or cosmetics that other players can vote on, and we may choose to incorporate into the game.

Before you can publish any mods, you must have accepted the Steam Subscriber Agreement and Supplemental Workshop Terms. If you have agreed to those terms, you will now be able to click the "Create" button and publish your mod.

3.4 Updating Mods

Mods can be updated from the same interface used for publishing mods.

Fill in the fields similar to before. We recommend including information about your update in the "Change Note" field. Note that the "Name" and "Preview Image" fields can be left blank while updating, which will leave them unchanged.

Once the fields have been filled out, select an existing upload from the list of your published mods at the bottom. Your mod will begin updating.

CHAPTER

FOUR

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.

4.1 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.

4.2 Master Bundles

Master bundles are the most efficient way for your assets to be packaged and distributed. They should be used for essentially any project that you plan to share with other players.

There are only a few asset types that are unsupported by master bundles.

4.2.1 File Setup

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

// If true, path within master bundle appends asset file name as subdirectory.
// For example:
// Guns/Eaglefire.asset → Guns/Eaglefire/Item.prefab
Bundle_Path_Include_Filename true
```

4.2.2 Tool Usage

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 platform-specific shaders are included, and exports a ".hash" file so the server can validate client asset bundle integrity.

4.2.3 Generated Files

After exporting a master bundle, the following files are generated in the export directory:

***_linux.masterbundle** and ***_mac.masterbundle** – These contain platform-specific shaders for Linux and macOS, respectively. These are only generated when "multiplatform" is enabled.

***.masterbundle.hash** – A hash file used by the server to verify asset bundle integrity for other platforms. For example, a server running Windows uses it to verify macOS and Linux asset bundle integrity.

> **Warning**
>
> Deleting the .hash file means cheaters can modify the asset bundle. For example, to make certain materials glow in the dark or to see through certain objects. Mods should always include the .hash file when redistributing asset bundles.

***.masterbundle.manifest** – A manifest file listing all assets in the bundle, their paths, and their hashes. When multiplatform is enabled, a manifest file is generated for each platform. Coincidentally, the manifest file is useful when debugging mods bundles. As it contains filepaths of all bundled assets, it can help confirm that assets were bundled as expected, and can also be useful when using master bundle pointers.

4.3 Individual Asset Bundles

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

- Per-map road textures.
- Colors used by charts.
- Level ambience.

4.3.1 Tool Usage

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.

4.3.2 Motivations

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.

4.4 Content Bundles (*.content)

Deprecated since version 3.22.4.0.

Content bundles were originally used by terrain, material palettes, and radio songs. Support for content bundles has been removed since February 25, 2022 and should be replaced by master 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 a *.masterbundle file instead. Then, add a corresponding MasterBundle.dat file as described in the file setup for master bundles.

CHAPTER

FIVE

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*.

5.1 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.

5.2 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 (items, animals, etc). Objects and vehicles 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
{
```

(continues on next page)

(continued from previous page)

```
        ID [...]
        Key1 Value
        Key2 Value
}
```

Is equivalent to:

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

5.3 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 LTS and 2019 LTS, `4` is 2020 LTS, `5` is 2021 LTS, and `6` is latest (Unity 2022 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.

5.4 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`.

5.5 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

SIX

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*.

6.1 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
        }
}
```

6.2 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
```

6.3 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
```

6.4 FAQ (Frequently Asked Questions)

Q. How do I write a multi-line value?

The \n escape sequence starts a new line. For example:

```
Text First line\nSecond line
```

Will set the value of Text to:

```
First line
Second line
```

Q. How do I write an in-line comment after a value containing quotation marks?

In-line comments require the value to enclosed in quotation marks, so quotation marks in the value must be escaped with \":

```
// The parser will read the comment as part of the value because it doesn't know where␣
↪the value ends.
Text Why use so-called "scare quotes" instead of /s? // Comment here

// The parser will exclude the comment from the value and replace the \" with quotation␣
↪marks.
Text "Why use so-called \"scare quotes\" instead of /s?" // Comment here
```

Q. Why can't I start a list or dictionary on the same line as a key?

This is—unfortunately—not supported because it would break backwards compatibility with the oldest .dat files. Older files may have [or { as the first letter of a value.

As an example of the problem:

```
SomeDictionary {
        SomeList [
        ]
}
```

Instead, the opening [or { must be placed on the next line:

```
SomeDictionary
{
        SomeList
        [
        ]
}
```

6.5 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

SEVEN

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

EIGHT

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.

8.1 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

NINE

CURATED ITEMS

Community-created items (such as skins and cosmetics) can be submitted to the *Steam Workshop* for consideration to be incorporated into the actual game. These items can be found under the Stockpile Submissions tab, where other players can vote on whether they would like to see a specific submission added into the game.

Accepted items may be made available for purchase, or otherwise be unlockable within the game. Most accepted items are sold in the Stockpile (also known as 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%. Your own split of that revenue share may be lower when an item has multiple contributors – such as when your item was created by multiple people, or if your item is added as part of bundle that contains many different peoples' items.

9.1 Requirements

For the tools necessary to start creating skins or cosmetics, refer to the *Getting Started* page. Some additional preparation is needed compared to creating mods:

1. Follow the submission guidelines.
2. Organize your Unity project.
3. Create your cosmetic or skin.
4. Specify a Mythical effect placement (if your item is a cosmetic).
5. Export a Unity package.
6. Submit your item to the *Steam Workshop*.

9.2 Guidelines

Most of these guidelines are intended to help promote consistency with *Unturned*'s art style. Here are the guidelines you should keep in mind:

1. Avoid high contrast colors. They're often painful to look at, especially when the item appears in harsh lighting.
2. Do not go darker than ■ #1e1e1e or brighter than □ #f0f0f0. Both extremes of brightness (and fully saturated colors) don't play nicely with the game's lighting. Most official content uses medium-intensity colors.
3. Edges of clothing should have a slightly darker border that is one pixel wide. This can be easily seen on official clothing items, along the cuffs and bottom hem of shirts.
4. Flat textures meant to blend into the terrain (e.g., similar to the in-game ghillie suit) should not be used. Instead, you can incorporated patterned camouflage into your items.

5. Textures should be kept to a reasonable resolution. Ideally, 2048x2048 scaled down to 1024x1024 for large items (e.g., the Maplestrike), and 1024x1024 scaled down to 512x512 for small items (e.g., the Cobra).
6. Avoid using high metallic or smoothness values. Unturned does not use reflection probes, and the only sources of metallic reflection data are limited to optional features like skybox reflections and screen-space reflections.
7. Corners of models should not be beveled. Most models have sharp edges (e.g., 90°). There is not a hard limit on vertex, triangle, or polygon count because anything matching the game's art style will naturally have a reasonable number.
8. Skins with custom models should generally respect the original item's silhouette. Be mindful that attachments (such as barrels, tacticals, sights, and grips) should still work on the custom model *and* look good.
9. Cosmetics should avoid potentially confusing players. For example: if a hat looks like hair, it should have some additional accessory or detail to help distinguish it as a cosmetic. Otherwise, it would look like the player isn't wearing any item at all.
10. Only submit content that you created yourself. Do not use copyrighted material or trademarks that you do not own or have permission to use.
11. We do not support custom shaders, i.e., shaders not included in the vanilla game cannot be used.
12. Unfortunately, physics do not play well with the Unturned character, and cannot be used on cosmetics at this time.

Remember: these are just guidelines. Except in specific cases (such as copyright infringement), we may occasionally accept items that break some of these guidelines. However, these are the things we are looking for in submissions. Sticking to these guidelines will help your chances of getting your item accepted.

9.3 Unity Project Organization

Organizing your project into two separate folders: one for your exported asset-bundled files (e.g., an `Item.prefab`) and one for your imported sources (e.g., a `.blend` file) is greatly appreciated. This makes it much easier for us to ensure only the necessary assets are included in the game.

For reference: all of the base game's asset-bundled files are in the `Assets/CoreMasterBundle` folder, and all of the source files are in the `Assets/Game/Sources` folder.

9.3.1 Cosmetics

Some additional notes exist for organizing cosmetic items.

- Map-related cosmetics are in per-map folders, and prefixed with the map's name. For example, Arid's "Arrowhead" exported files are located in `Assets/CoreMasterBundle/Items/Arid/Arid_Arrowhead` and the source files are in `Assets/Game/Sources/Items/Arid`.
- Outfits have their items in a per-outfit folder, and are prefixed with the outfit name. For example, the "Cultist's Mask" exported files 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 "Backpack Turtle" 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 game's core asset bundle, a `.unitypackage` file is required along with the regular `.dat` files for items. To export the package:

1. Select the folders containing your `Item.prefab` files (or equivalent asset-bundled files for other types of items). For example, if we were submitting our official Fedora item then we 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 source files that aren't directly placed in the asset bundles (i.e., the meshes, materials, textures, etc.).

> **Note**
>
> The Unity package is in *addition* to the regular asset `.dat` and `English.dat` files required for items to work. 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 Creating Cosmetics

In terms of setup, cosmetics are identical to actual in-game clothing items. The main difference between cosmetics and clothing is that the former does not offer any benefits to the player, appear over most worn clothing items, and can have their visibility toggled at will by the player.

Since these are otherwise identical, we recommend referencing the clothing items included in the ExampleAssets.unitypackage in order to create your cosmetic items.

9.5.1 Mythical Effect Placement

Most cosmetic items will want to support mythical effects. You will need to add an "Effect" child transform to both the Item.prefab and the clothing prefab (i.e., Backpack.prefab, Glasses.prefab, Hat.prefab, Mask.prefab, or Vest.prefab) for your item to support this.

The orientation is rather unfortunate: +Z is the mythical's up direction and +Y is the mythical's forward direction.

9.6 Creating Skins

Any item in *Unturned* could support skins, but not every item does at this time. We recommend only creating skins for items that are already skinnable. This includes most weapons, along with a few miscellaneous items such as Canned Beans and the Detonator. The unwrapped meshes are included as part of the ExampleAssets.unitypackage.

At the very least, you will want to create a custom albedo for your skin. You can also add custom metallic or emission textures. When eventually uploading your skin to the Steam Workshop, please make sure you follow the "Exporting Unity Package" steps to include your source files! This allows us to add any extra needed assets, or fix minor issues.

Unless your skin includes a custom mesh (as detailed in a later section), your asset-bundled files will not include a prefab. Instead, skins will include the material(s) used by the item and any attachments attached to it.

- **Primary**: The `Skin_Primary.mat` is the material used by the skinned item itself. Every skin should have a single primary material.

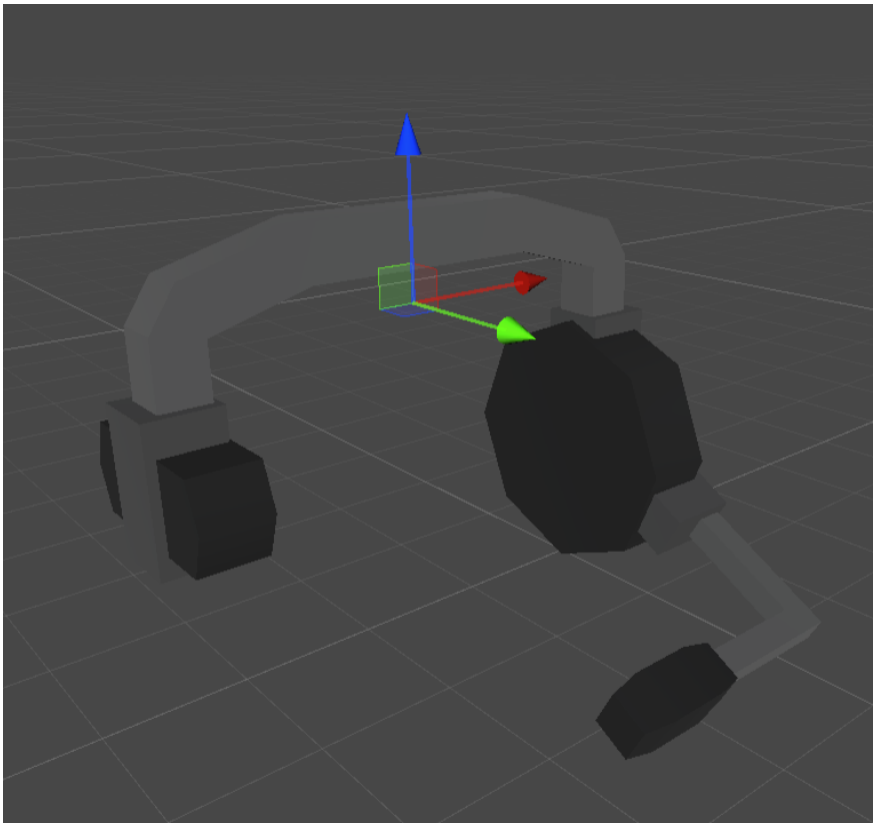


Fig. 1: Example “Effect” transform positioning and orientation.

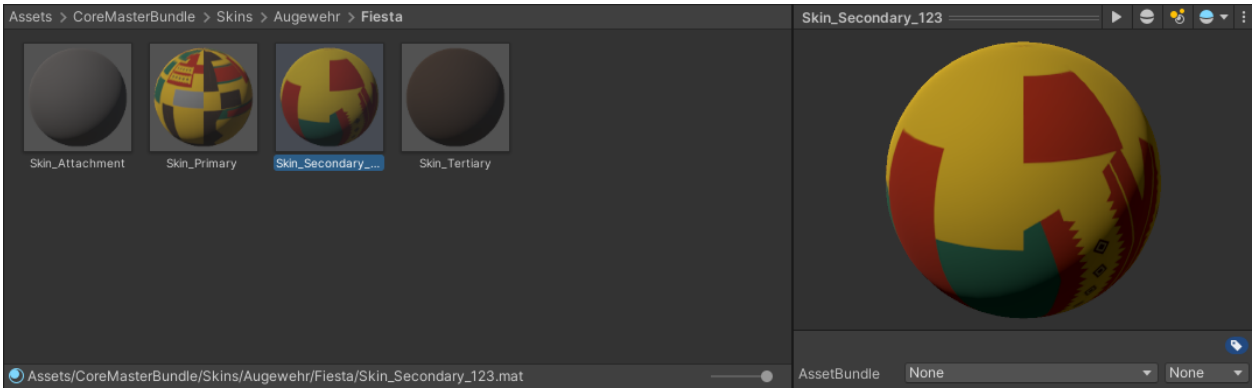


Fig. 2: The Fiesta Augewehr skin includes all four types of materials.

- **Secondary**: Each `Skin_Secondary_#.mat` included in the bundle files is for a specific attachment (where the # is the attachment's legacy ID). Skins can have multiple secondary materials. When creating skins for sniper rifles, you will usually want to include at least one secondary material for a scope.
- **Attachment**: The `Skin_Attachment.mat` (also called the "layered attachments material") is used when a secondary material has not been provided for an attachment that has texture masks. For example, the 8x Scope has its mount and knobs masked out.
- **Tertiary**: The `Skin_Tertiary.mat` (also called the "fallback attachments materials") is used when none of the other included materials are applicable for an attached attachment. You will usually want to include a tertiary material if your skin has an attachments material.

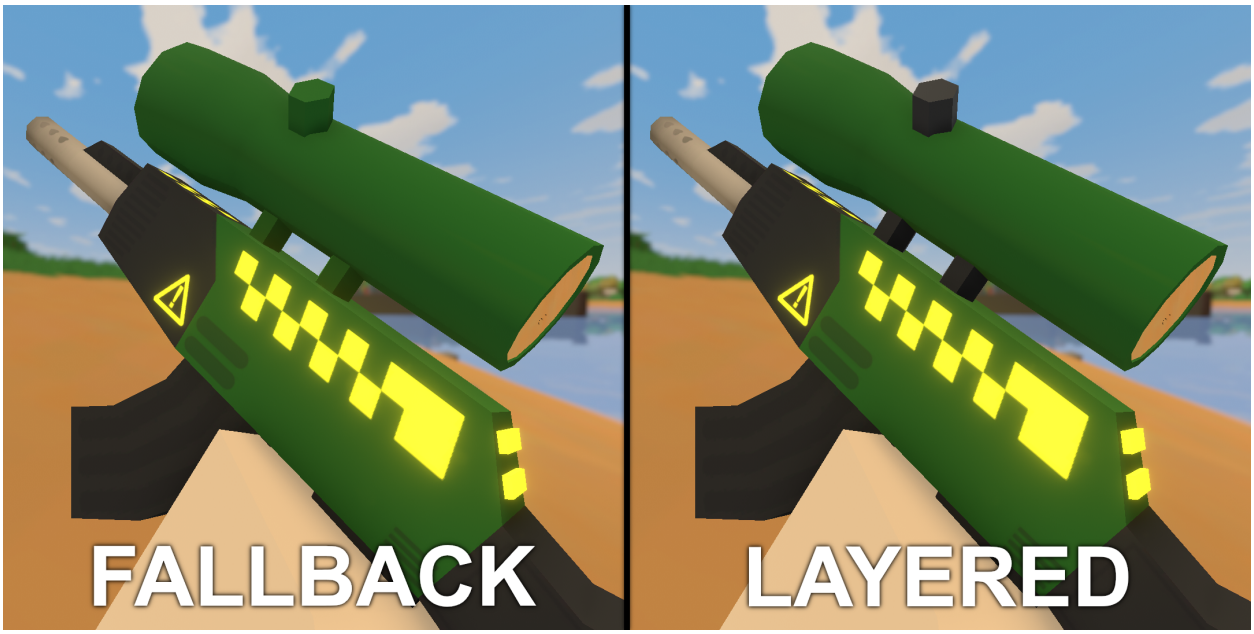


Fig. 3: Notice how some parts of the 8x Scope retains some of its original texture when a Layered Attachments material is included, while the Fallback Attachments material completely covers it.

Although most skins keep their layered (Skin_Attachment.mat) and fallback (Skin_Tertiary.mat) textures identical, this is not required. Some skins have fairly different secondary, attachment, and tertiary materials. The Bloodsport Calling Card, Bouquet Bluntforce, and Vortex Augewehr are good examples of this.

9.6.1 Custom Models

Skins can override an item's original model with a custom one. If you create a skin that does this, it's more likely to be accepted if your custom model still respects the original silhouette of the item. By extension, things like attachments and stat counters should still look good when attached to the item.

Setting this up in Unity is simple. Your asset-bundled files should include a `Override_Mesh_#.prefab` (where # is the LOD's index). For example, `Override_Mesh_0.prefab`. This prefab simply includes a Mesh Filter component that is linked to your custom model.

9.7 Using Collections

Collections allow for grouping multiple item submissions together, making it easier for users to discover and rate similar content when browsing the Steam Workshop. Creators often use collections to group skins that all share a similar pattern together.

Some creators may create collections when they want to showcase different versions of their item (e.g., palette swaps). While these *could* be combined into a single submission, having them separate allows for players to vote on the specific version(s) they would like to see accepted into the game.

Another way of using collections is to organize submissions that were designed to be used together. For example, a bunch of individual cosmetic items that would form a complete outfit. Creating a collection that contains each individual item allows players to vote on the specific item(s) they would like to see, and makes it easier for us to accept only a couple pieces from an outfit (e.g., when choosing items for a new box). You may also consider using an *OutfitAsset* to create a preview image that can be used on the Workshop page for the outfit's collection or its related items.

CHAPTER

TEN

ANIMATION

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

10.1 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

ELEVEN

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.

11.1 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.

11.2 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

TWELVE

MOD HOOKS

12.1 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

12.2 Event Listeners

12.2.1 Activation Event Hook

Events when a component or game object are enabled and disabled. Useful for extending toggleable actions in the base game.

12.2.2 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.

12.2.3 Collision Damage

Damages players when they overlap a trigger collider.

12.2.4 Collision Event Hook

Events for player overlaps with a trigger collider. This is primarily useful for server-side objects as collisions are not triggered by other players client-side.

On the client-side only the locally-controlled character is on the Player layer, whereas other player characters are on the Enemy layer. On the server all characters are on the Player layer. The collision event hook only reports collisions with colliders on the Player layer, which is why it is not as useful on the client.

12.2.5 Destroy Event Hook

Event when a component or game object is removed from the scene.

12.2.6 Explosion Spawner

Allows Unity events to apply damage in a sphere. (doesn't have any visual effects)

Intended to replace unsupported/unintentional use of `Grenade.cs` and `Rocket.cs` scripts.

12.2.7 Gun Attachment Event Hook

Allows gun item game objects (including children) to receive events when sights, tacticals, grips, barrels, and magazines are attached, replaced, and/or detached.

12.2.8 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.

12.2.9 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.

12.2.10 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.

12.2.11 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.

12.2.12 Timer Event Hook

Allows events to set or cancel a timer, and triggers an event when the timer expires.

12.2.13 Useable Event Hook

Events for EquipableItem prefab of any item type. These events are fired on server and client.

12.2.14 Useable Gun Event Hook

Events for EquipableItem prefab. Supersedes VehicleTurretEventHook. Will not have its events triggered when used by a sentry. These events are fired on server and client.

12.2.15 Vehicle Event Hook

Events for driver entering and exiting the vehicle. These events are fired on server and client.

12.2.16 Vehicle Gear Shift Event Hook

Events for vehicle gearbox entering and exiting a target gear.

12.2.17 Vehicle Health Event Hook

Events for vehicle health passing a comparison against a target number.

12.2.18 Vehicle Turret Event Hook

Events for Turret_# GameObjects in the vehicle when guns are used. These events are fired on server and client.

12.2.19 Weather Event Hook

Events for day, night, full moon, and weather. These events are fired on server and client.

> **Warning**
>
> **WeatherEventHook** is computationally expensive and can be reasonably expected to cause performance issues when overused. For example, on foliage assets—which are often placed in large quantities.

12.2.20 Custom Weather Event Hook

Events for a specific custom *Weather Asset*. Any map can have an unlimited number of weather types and weather listeners.

12.3 Event Instigators

12.3.1 Airdrop Spawner

Allows Unity events to call in an airdrop. Optionally overrides the cargo and destination.

12.3.2 Barricade Spawner

Allows Unity events to place barricades.

12.3.3 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 Configuration* before these can be triggered. This ensures hosts are aware of their usage. Single-player defaults to enabled.

12.3.4 Item Spawner

Allows Unity events to spawn dropped items.

12.3.5 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 Configuration* before these can be triggered. This ensures hosts are aware of their usage. Single-player defaults to enabled.

12.3.6 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.

12.3.7 Mob Alert Spawner

Allows Unity events to startle nearby animals and zombies.

Optionally uses a nearby player as the origin of the alert.

12.3.8 NPC Global Event Messenger

Allows Unity events to broadcast Event NPC rewards. The `NPC Global Event Hook` can then listen for these events.

12.3.9 Vehicle Spawner

Allows Unity events to spawn a vehicle. Optionally overrides the paint color.

12.4 Misc

12.4.1 Barricade Destroyer

Forcefully removes barricades within a sphere, optionally playing their Explosion effects and/or spawning their Item_Dropped_On_Destroy item(s).

12.4.2 Fall Damage Override

Allows any game object to override the fall damage when a character lands on it or one of its descendants.

12.4.3 Crafting Tag Provider

Allows the following entities to modify which crafting tags (workstations) are available to nearby players:

- Barricades
- Structures
- Vehicles
- Resources
- Objects

This component is used by vanilla for Heat Source backwards compatibility.

12.4.4 Crafting Tag Modifier

Linked from a Crafting Tag Provider. Allows Unity events to modify which crafting tags (workstations) are available to nearby players.

As an example: the automatic Heat Source backwards compatibility adds a Crafting Tag Modifier to the Fire game object with an Activation Requirement of Invert and Mode Remove. This removes the Heat Source tag while the Fire is inactive.

12.4.5 Engines Curves Component

Stores the curves that determine engine torque given at each engine speed (RPM) when attached to the root vehicle prefab.

12.4.6 Music Audio Source

Reassigns a sibling Audio Source's output audio mixer group to the vanilla Music mixer, respecting the player's volume preferences.

12.4.7 Repeat

Repeats an event a configurable or random number of times. Essentially a for-loop for Unity events.

CHAPTER

THIRTEEN

UPGRADING UNITY VERSION

This page covers the various engine upgrades that *Unturned* has undergone, and the important changes from them. While much of the older information is unlikely to still be relevant, it may provide some insight on the off chance that you are updating a particularly old mod, or referencing a tutorial created for a previous Unity version.

13.1 From Unity 5 LTS to 2017 LTS

The upgrade from Unity 5.5 to 2017.4 was a large change, and any mods created in Unity 5 are no longer compatible with the game. For archival purposes, the Unity 5.5 LTS version of the game remains available from the "unity-5.5" beta branch.

There were two key reasons for going through an engine upgrade. First, Unity 2017 marked the start of Unity's annual releases as they focused on stability, and performance could be improved by removing workarounds for some now-resolved Unity bugs. Second, Apple deprecated their OpenGL support, so supporting 2017.4's Metal graphics API would be important for macOS players.

Master bundles were introduced to make updating mods easier.

13.1.1 Master Bundles

Rather than exporting each bundle as individual .unity3d files, you can now export multiple bundles into a combined **.masterbundle**. Master bundles are not required, but they are more efficient when it comes to time and performance. However, they are not *required*, and you may find it easier to quickly iterate on a .unity3d bundle.

If the game detects a master bundle file, it will load bundles from sub-directories using their relative folder path.

For example, `MyModBundles/Items/Guns/MyGunItem.dat` could be the path to an item's .dat file. If `MyModBundles` contains a master bundle, it will try to load all Unity files using the relative path `/Items/Guns`. It is important for .dat folders and Unity folders to line up, 1:1.

13.1.2 Installing the Tools

Copy all of the files from `Unturned/Bundles/Sources/Tools` into the `.../Assets/Editor` directory of your Unity project. The **MasterBundleTool** will now be available from **Window > Unturned**. To use this tool, first assign your folder to an AssetBundle, and then export your bundles to a location that contains a corresponding MasterBundle.dat file.

Shaders can be imported from the `All_Shaders.unitypackage` (also located in a subdirectory of `Sources`). We recommend using official shaders to improve compatibility with base game content.

13.1.3 Exporting from Unity

Attention

This section of the documentation is admittedly rather old, and it should eventually be moved to a better spot. Fortunately—these steps are still accurate, and the information below has been revised recently. *[2024 November]*

Exporting requires some initial setup within Unity and the target location for your exported master bundle.

In Unity, select the folder within the Project Browser that contains your bundles. Assign it to an AssetBundle from the Inspector. If you haven't created one yet, this is the time to do so. For example, vanilla/official bundles are in an AssetBundle named "core.masterbundle", while the curated Hawaii map's bundles are in "hawaii.masterbundle".

Open the Master Bundle Tool. Select your AssetBundle from the dropdown. If your mod should be fully compatible with MacOS and Linux devices, you will need to check the "Multi-platform" option before exporting.

Tip

When you're still working on your mod, you don't need to enable multi-platform support! This option can be toggled at any time. By only enabling this for release candidates, you can cut down on build time while you're still iterating.

Clicking the "…" button will let you select a target location to export the master bundle into. Before you can proceed, you will need a MasterBundle.dat file in that location.

Navigate to your target location. Any folder that would contain a master bundle needs a **MasterBundle.dat** file. When the game traverses the file hierarchy, it will check for this file, and if present it will assume all sub-folders should be redirected into that master bundle.

Create this file at your target location. You should configure the following options in your MasterBundle.dat file:

- **Asset_Bundle_Name**: Filename of the Windows master bundle exported from Unity. For example, `hawaii.masterbundle`.
- **Asset_Prefix**: Filepath to the folder you selected as an AssetBundle in Unity. For example, your path could be `Assets/MyModBundles`. In which case, the game could look for an item in `Assets/MyModBundles/Items/Guns`.

Note

Bundled assets have a few optional properties they can use. The **Exclude_From_Master_Bundle** flag will cause that .dat file to ignore any parent MasterBundles in the folder hierarchy, and instead look for an individual .unity3d file. The **Master_Bundle_Override** property lets you specify the name of a master bundle to redirect to. **Bundle_Override_Path** can be used to have multiple items share the same setup in Unity (such as how the vanilla note objects share the same model), or combined with the previous property to use models from other master bundles (e.g., if you wanted to use the Eaglefire's model but give it custom stats).

Back in Unity, you can now export your master bundle.

13.1.4 Troubleshooting

Colors too light or dark in-game

Most likely the Color Space setting in your Unity project needs adjusting. Navigate to Edit > Project Settings > Player. Under the Other Settings section change Color Space to Linear.

If that doesn't work, double-check that sRGB is enabled on your textures.

Devkit Foliage

Materials default to instancing disabled now, so enable the Instancing flag in the Unity material inspector and rebuild the asset bundle.

13.2 From Unity 2017 LTS to 2018 LTS

For archival purposes, the Unity 2017.4 LTS version of the game remains available from the "unity-2017.4" beta branch.

13.2.1 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.

13.2.2 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.

13.2.3 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.

13.2.4 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.

13.2.5 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.

13.3 From Unity 2018 LTS to 2019 LTS

There are very few notable changes from this upgrade. For archival purposes, the Unity 2018 LTS version of the game remains available from the "unity-2018" beta branch.

By default, Unity no longer supports importing multiple animations from a single .blend file. Exporting to an exchange format like .fbx is recommended.

> **Tip**
>
> This is recommended for meshes/models as well! The base game has always handled its own assets like this, as well.

You can find more details in case #1186253 on the Unity issue tracker.

CHAPTER

FOURTEEN

INTRODUCTION TO ITEMS

Items are created from the ItemAsset class. They 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.

14.1 Unity Asset Bundle Contents

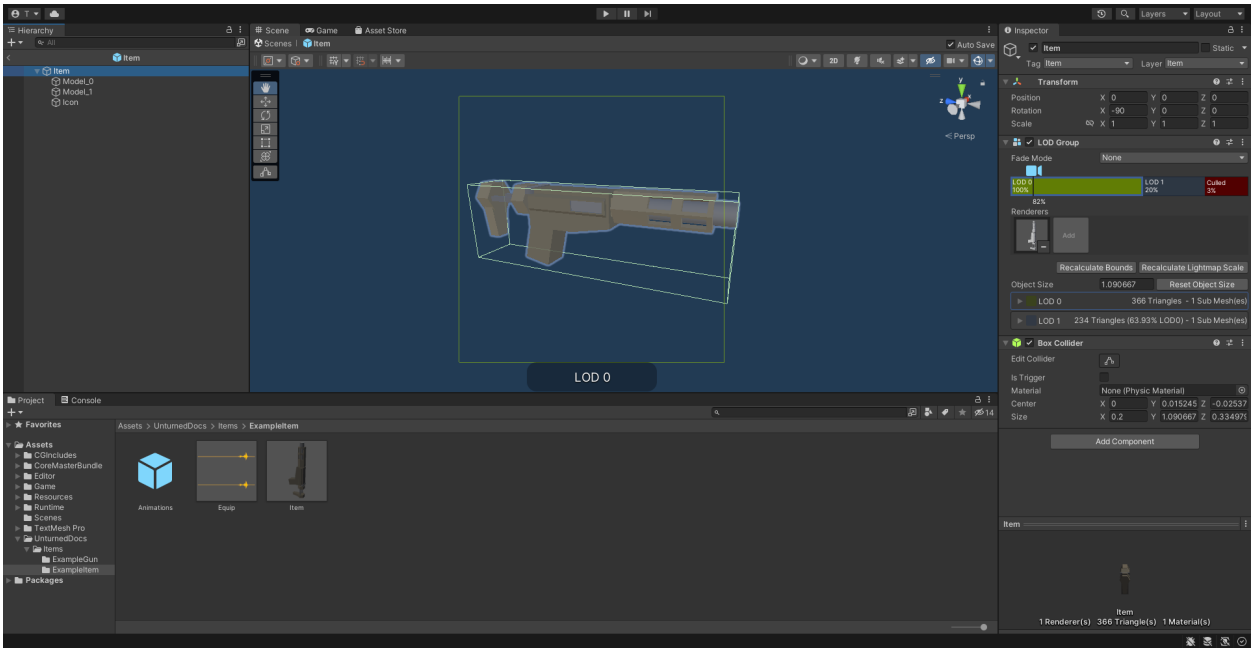


Fig. 1: An example of an item being set up in the Unity editor.

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.

14.1.1 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.

14.1.2 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.

14.1.3 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.

14.1.4 Skin Base Textures

Note

We should eventually add more documentation about this, but items support texture masking. This is rarely relevant for modders creating their own items, but it is important for skins. Items can optionally include `Albedo_Base.png`, `Metallic_Base.png`, or `Emission_Base.png` textures in their asset-bundled files. When using certain skins, those parts will be masked out and retain their original material.

14.2 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.

14.2.1 Properties

Property Name	Type	Default Value
Add_Default_Actions	*bool*	See description
Allow_Manual_Drop	*bool*	`true`
Amount	*uint8*	1
Backward	*flag*	*deprecated*

continues on next page

Table 1 – continued from previous page

Property Name	Type	Default Value
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`
Deleted_At_Zero_Quality_Effect	*Effect Assets*	`0`
Deleted_At_Zero_Quality_Rewards	*Rewards*	`0`
Destroy_Item_Colliders	*bool*	`true`
Equipable_Movement_Speed_Multiplier	*float32*	`1`
EquipableModelParent	*EEquipableModelParent*	See description
EquipablePrefab	*Master Bundle Pointer*	
EquipAudioClip	*Master Bundle Pointer*	`Equip`
Fishing_Catchable	*FishingCatchableProperties*	
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
Shared_Skin_Apply_Visuals	*bool*	`true`
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`

14.2.2 EEquipableModelParent Enumeration

Named Value	Description
`RightHook`	Item is attached to Right_Hook.
`LeftHook`	Item is attached to Left_Hook.
`Spine`	Item is attached to Spine.
`SpineHook`	Item is attached to Spine_Hook, an optional extra child bone of the Spine bone.

14.2.3 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.

14.2.4 Property Descriptions

Add_Default_Actions bool

If true, actions are automatically created for refill ammo, repair, and salvage blueprints. Defaults to true if no `Actions` are specified.

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`.

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

Controls whether or not this item can be equipped by the player. This property *technically* requires that `Useable` has been configured to any value other than `None`, as items are not equippable without the functionality provided by having a Useable class.

While the inclusion of this property may seem unorthodox, it does have some niche uses. For example, you could create a gun that can only be used by sentries.

This property defaults to `true` if the `Useable` property has been set. 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`.

Deleted_At_Zero_Quality_Effect Asset Pointer to Effect Assets

If assigned, and *Should_Delete_At_Zero_Quality* is `true`, this effect is played when the item breaks.

Deleted_At_Zero_Quality_Rewards Rewards

If assigned, and *Should_Delete_At_Zero_Quality* is `true`, these rewards are granted to the player when the item breaks.

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.

EquipableModelParent EEquipableModelParent

Overrides which transform to attach the item to when equipped by the player. Spine may be a better interpolation space for items with animations moving the model between hands.

Normally, this property defaults to `RightHook`. However, items using the deprecated `Backward` flag will cause this to instead use `LeftHook`.

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.

Fishing_Catchable Fishing Catchable Properties

Overrides settings when caught by a fishing rod. For more information, please refer to *Fishing Catchable Properties*.

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`.

Shared_Skin_Apply_Visuals bool `true`

If false, skin material and mesh are not applied when `Shared_Skin_Lookup_ID` is set. For example, a custom axe can transfer the kill counter and ragdoll effect from a vanilla item's skin without also transferring the material and mesh.

Should_Delete_At_Zero_Quality bool `false`

Item should be deleted when at 0% quality.

Should_Drop_On_Death bool `true`

When a player dies with this item, should an item drop be spawned?

> **Note**
>
> The item is not kept after respawning.

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`

Setting this property adds functionality from a corresponding Useable class. Unless `Can_Player_Equip` has been configured otherwise, this *at least* means the item is equippable.

This property is often used in conjunction with the `Slot` property, which determines where an item can be equipped from.

14.2.5 Blueprints and Actions

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

14.3 Localization

Name *string*: Item name in user interfaces.

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

CHAPTER

FIFTEEN

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.

15.1 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.

15.1.1 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_#_State_Transfer_Delete_Attachme	*bool*	`false`
Blueprint_#_Supplies	*uint8*	`0`
Blueprint_#_Supply_#_AllowEmpty	*bool*	`false`
Blueprint_#_Supply_#_Amount	*uint8*	`0`
Blueprint_#_Supply_#_Critical	*flag*	
Blueprint_#_Supply_#_ID	*uint16*	
Blueprint_#_Supply_#_Prioritization	*enum*	
Blueprint_#_Tool	*uint16*	`0`
Blueprint_#_Tool_Critical	*flag*	
Blueprint_#_Type	*EBlueprintType*	
Blueprints	*uint8*	`0`

15.1.2 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.

15.1.3 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.

15.1.4 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_#_State_Transfer_Delete_Attachments bool `false`

If `true` and `State_Transfer` is enabled, any output guns will have all of their attachments deleted.

Blueprint_#_Supplies uint8 `0`

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

Blueprint_#_Supply_#_AllowEmpty bool `false`

If true, items with an "amount" of zero—such as empty magazines—are treated as an amount of one. In vanilla this is used to enable salvaging empty magazines.

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`.

This property can also be set to a string value of `this`, which will use the the owning item's legacy ID. Useful for salvaging blueprints to avoid accidentally writing the wrong ID.

Blueprint_#_Supply_#_Prioritization enum

Controls which items are used first. Can be set to `LowestAmount` or `LowestQuality`.

- `LowestAmount` sorts items by their amount (e.g., number of bullets in magazine) from lowest to highest, and consumes the emptiest ones first.
- `LowestQuality` sorts items by their quality from lowest (0%) to highest (100%), and consumes those nearest 0% first.

`AMMO` type blueprints default to `LowestAmount`, otherwise defaults to `LowestQuality`.

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.

15.1.5 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`.

15.2 Game Data File v2

> **Warning**
>
> Under construction! We might move this to a different page.

The game can auto-convert most blueprints to this new format! For more information, please refer to *the ResaveAssets launch option*.

Starting with version 3.25.5.0, Blueprints can be specified as a *list* rather than prefixing each property with `Blueprint_#_`. For example:

```
Blueprints
[
        {
                // First blueprint properties
        }
        {
                // Second blueprint properties
        }
]
```

Each blueprint dictionary has the following properties:

15.2.1 Properties

Property Name	Type	Default Value
CategoryTag	*Asset Pointer* to *Tag Assets*	
Conditions	*Conditions*	
Effect	*Asset Pointer* to *Effect Assets*	
InputItems	*list* of *Blueprint Input Items*	
Map	*string*	""
Name	*string*	""
Operation	*EBlueprintOperation*	`None`
OutputItems	*list* of *Blueprint Output Items*	
RequiresNearbyCraftingTags	*list* of *Asset Pointer* to *Tag Assets*	
RequiresStaticTags	*list* of *Asset Pointer* to *Tag Assets*	
Rewards	*Rewards*	
Searchable	*bool*	`true`
Skill	*EBlueprintSkill*	`None`
Skill_Level	*int32*	`0`
StateTransfer	*bool*	`false`
StateTransfer_DeleteAttachments	*bool*	`false`
Type	*EBlueprintType*	Deprecated.
VisibleWithUnmetConditions	*bool*	`false`

15.2.2 EBlueprintOperation Enumeration

Named Value	Description
`None`	No special modification.
`RepairTargetItem`	Restore target item to full quality.
`FillTargetItem`	Transfer amount from first input item to target item.

CategoryTag Asset Pointer to Tag Assets

Determines which category the blueprint appears under in the crafting menu. Enables the creation of custom categories.

This replaces the `Type` property which acted as both a hardcoded category and modified behavior of crafting.

Conditions Conditions

NPC conditions which must be met before the blueprint can be crafted.

> **Note**
>
> By default, the blueprint will be hidden until all of the conditions are met. If you have configured display text for your conditions you can enable *VisibleWithUnmetConditions*.

Effect Asset Pointer to Effect Assets

An effect to play upon successfully crafting.

InputItems list of Blueprint Input Items

Required ingredients/supplies needed to craft this 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.

Name string ""

Optional case-sensitive identifier in list of blueprints. Can be used, for example, to reference this blueprint from a context menu action or a list of prohibited blueprints.

Operation EBlueprintOperation `None`

Controls what blueprint does with input items.

OutputItems list of Blueprint Output Items

Items created by this blueprint.

RequiresNearbyCraftingTags list of Asset Pointer to Tag Assets

Tag Assets that must be available from nearby crafting tag providers (workstations).

For example, to require both Chemical Mixing and Workbench:

```
RequiresNearbyCraftingTags
[
        // Chemical Mixing
        99896da563a748148460c67b9962874f

        // Workbench
        7b82c125a5a54984b8bb26576b59e977
]
```

RequiresStaticTags list of Asset Pointer to Tag Assets

Similar to `RequiresNearbyCraftingTags`, but only checked once during level startup.

Can test for *Level Assets* `Tags`. For example, rather than using the `Map` property to test for a level by name, the level can signal it supports a set of blueprints with a tag.

Vanilla makes the following tags available:

```
"73eb818d1aa044c7bb4e61b8f9b37a3c" // Building In Safezones Allowed
"f663677b88de40ec80ff36b0c1cae544" // Not-singleplayer
"d7bd989414644b19b3299be0c6fab5f0" // Singleplayer
```

Rewards Rewards

NPC rewards granted when crafting this 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.

Skill EBlueprintSkill `None`

The player must have leveled the specified skill in order to craft this blueprint. This property is used in conjunction with `Skill_Level`.

Skill_Level int32 `0`

If the blueprint requires a skill, its level must be greater than or equal to this value. This property is used in conjunction with `Skill`.

StateTransfer bool `false`

Transfer the current state the first input item 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).

StateTransfer_DeleteAttachments bool `false`

If `true` and `StateTransfer` is enabled, any output guns will have all of their attachments deleted.

Type EBlueprintType

Should not be used. Type acted as both a hardcoded category and modified behavior of crafting. These have been separated into the `CategoryTag` and `Operation` properties instead.

VisibleWithUnmetConditions bool `false`

If true, blueprint can become visible in the crafting list even when NPC conditions are not met. This should typically only be enabled if all conditions are configured to be visible in the details panel. Otherwise, the default "conditions unmet" label isn't very informative for players.

CHAPTER

SIXTEEN

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.

16.1 Game Data File

16.1.1 Properties

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

16.1.2 EActionType Enumeration

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

16.1.3 Property Descriptions

Action_#_Blueprint_#_Index uint8 `0`

Index of the blueprint that action should perform.

> **Note**
>
> `Action_#_Blueprint_#_Name` should be used instead of `Action_#_Blueprint_#_Index` where possible because the index may change if the item's Blueprints list is reorganized. This requires specifying a `Name` in the

blueprint.

Action_#_Blueprint_#_Name string ""

Name of the blueprint that action should perform. This requires specifying a `Name` in the blueprint, however.

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`, `Stack`, `Store`, `Take`, and `Unstack`.

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

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

Action_#_Text string

Context button name. If a key matching this value exists in the per-item localization file then that text is shown, otherwise this value is shown as-is.

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

Action_#_Tooltip string

Context button tooltip. If a key matching this value exists in the per-item localization file then that tooltip is shown, otherwise this value is shown as-is.

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.

16.2 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.

CHAPTER

SEVENTEEN

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.

17.1 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`

17.1.1 Properties

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

17.1.2 Property Descriptions

Recover uint16 `0`

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

CHAPTER

EIGHTEEN

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.

18.1 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`

18.1.1 Properties

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

18.1.2 Property Descriptions

Strength uint16 `0`

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

CHAPTER

NINETEEN

BACKPACK ASSETS

Backpacks are created from the ItemBackpackAsset class. They can be worn by players and zombies, and occupy the player's "backpack" slot.

19.1 Game Data File

The ItemBackpackAsset class inherits properties from the *ItemBagAsset* class. Properties that are required (or recommended) are listed in the table below.

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

Some inherited properties behave differently when used by this class. Notably, these are:

1. *Armor* (from *ItemClothingAsset*). Backpacks do not cover any body part(s) when worn, so this property has no effect.
2. *InventoryAudio* (from *ItemAsset*). Defaults to `Sounds/Inventory/LightMetalEquipment.asset` when *Width* or *Height* are less than `3`. To `Sounds/Inventory/MediumMetalEquipment.asset` when less than `6`. Otherwise, to `Sounds/Inventory/HeavyMetalEquipment.asset`.

19.1.1 Properties

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

CHAPTER

TWENTY

BAG ASSETS

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

20.1 Game Data File

The ItemBagAsset class inherits properties from the *ItemClothingAsset* class.

20.1.1 Properties

Property Name	Type	Default Value
Height	*uint8*	`0`
Width	*uint8*	`0`

20.1.2 Property Descriptions

Height uint8 `0`

Number of rows (vertical storage space).

Width uint8 `0`

Number of columns (horizontal storage space).

CHAPTER

TWENTYONE

BARREL ASSETS

Barrels (or "barrel attachments") are created from the ItemBarrelAsset class. Barrel attachments are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

21.1 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`

21.1.1 Properties

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

21.1.2 Property Descriptions

Ballistic_Drop float32 1

Note

Moved to *Caliber Asset*.

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.

CHAPTER

TWENTYTWO

BARRICADE ASSETS

Barricades are created from the ItemBarricadeAsset class. They can be placed by players or in the level editor.

This inherits the *PlaceableAsset* class.

22.1 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.

22.2 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`.

> **Note**
>
> `Armor_Falloff*` properties are applicable to Animals, Objects, Resources, Structures, and Vehicles as well. (TODO: move into their own doc?)

Armor_FalloffMaxRange *float*: Ranged damage (guns) from greater than this distance finishes decreasing toward `Armor_FalloffMultiplier`. Defaults to -1, in which case armor falloff is ignored.

Armor_FalloffRange *float*: Ranged damage (guns) from greater than this distance begins decreasing toward `Armor_FalloffMultiplier`. Defaults to `Armor_FalloffMaxRange` (immediate decrease, no transition).

Armor_FalloffMultiplier *float*: [0, 1] normalized percentage of incoming damage to apply past `Armor_FalloffMaxRange`.

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 *bool*: When `true`, this can be placed inside someone else's claimed area. This property used to be a *flag*, and using it as such is still supported. Defaults to false, except when using `Build Charge`.

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.

CanVehicleHookWhileAttached *bool*: By default, vehicles with "hooks" (such as the Skycrane) cannot pick up vehicles with barricades attached. If all barricades on the vehicle set this to `true` then the vehicle *can* be picked up. Defaults to `false`.

> **Warning**
>
> In the 3.24.7.0 update notes **CanVehicleHookWhileAttached** was mistakenly referred to as **CanParentVehicleBePickedUp**. In the next update both properties will work, but for the meantime **CanVehicleHookWhileAttached** should be used instead.

Can_Zombies_Target *bool*: If true, this item is eligible for zombies to detect and attack when stuck. 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.

RequiresHeatSourceCraftingTagConversion *bool*: Applicable to `Oven`, `Torch`, and `Campfire` Build types. Defaults to true. Performs the following modifications during load for backwards compatibility:

1. Adds the vanilla Heat Source tag (`20f30322bbcc4b01a4f116d22b24c21a`) to PlaceableProvidesCraftingTags if empty.
2. Adds a Crafting Tag Modifier component to the Fire child game object with Mode Remove, Activation Requirement Invert, and Tag `20f30322bbcc4b01a4f116d22b24c21a`. This removes the Heat Source tag when Fire is inactive.
3. Adds a Crafting Tag Provider component to the Barricade game object with Modifiers set to the component added to Fire.

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.

CHAPTER

TWENTYTHREE

BEACON ASSETS

Beacons are created from the ItemBeaconAsset class. Placing the beacon will start a zombie horde event, which can be completed by killing a predetermined number of zombies before the beacon is destroyed.

This inherits the *BarricadeAsset* class.

23.1 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.

23.2 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.

CHAPTER

TWENTYFOUR

BLUEPRINT INPUT ITEMS

Configuration for a required ingredient item in a *blueprint*.

24.1 Value Format

For straightforward cases, a simple one-liner format is supported:

- {ID}
- {ID} x {Amount}
- this
- this x {Amount}

For examples:

```
// Canned Beans:
78fefdd23def4ab6ac8301adfcc3b2d4

// Or, using Canned Beans' legacy ID:
13

// Two cans of beans:
13 x 2

// Owning asset:
this

// Two of owning asset:
this x 2
```

24.2 Dictionary Format

Unlike the shorter format above, using these properties requires the { } *dictionary* format. For example:

```
InputItems
[
        {
                // First input item properties
        }
        {
```

(continues on next page)

(continued from previous page)

```
                // Second input item properties
        }
]
```

24.2.1 Properties

Property Name	Type	Default Value
Amount	*uint8*	1
AllowEmpty	*bool*	`false`
AllowFull	*bool*	`true`
AllowMaxQuality	*bool*	`true`
CountEmptyAsOne	*bool*	`false`
CountingMethod	*ECraftingInputCountingMethod*	See description
Critical	*bool*	`false`
Delete	*bool*	`true`
ID	*Asset Pointer* to *Item*	
Prioritization	*ECraftingInputPrioritization*	See description

24.2.2 ECraftingInputPrioritization Enumeration

Named Value	Description
`LowestAmount`	Sort items with lowest "amount" to front of list.
`HighestAmount`	Sort items with highest "amount" to front of list.
`LowestQuality`	Sort items with lowest quality% to front of list.
`HighestQuality`	Sort items with highest quality% to front of list.

24.2.3 ECraftingInputCountingMethod Enumeration

Named Value	Description
`TotalItems`	Sum up number of items found, ignoring amount.
`TotalAmount`	Sum up "amount" of each item. Optionally counting zero as one.

Amount uint8 `1`

Quantity of this item needed to craft the blueprint. Minimum value of one.

AllowEmpty bool `false`

If true, items with an "amount" of zero are included in eligible supplies. Otherwise, they are ignored (default).

Defaults to `true` if `CountEmptyAsOne` is true.

This option exists primarily for the `FillTargetItem` operation's internal use.

AllowFull bool `true`

If true, items with an "amount" greather than or equal to their maximum amount are ignored. Otherwise, they are eligible (default).

This option exists primarily for the `FillTargetItem` operation's internal use.

AllowMaxQuality bool `true`

If true, items with quality of 100% are eligible (default). Otherwise, they are ignored.

This option exists primarily for the `RepairTargetItem` operation's internal use.

CountEmptyAsOne bool `false`

If true, items with an "amount" of zero are included in eligible supplies as amount 1.

This option is used in vanilla for salvaging empty magazines.

CountingMethod ECraftingInputCountingMethod

Determines how available amount of this input is calculated.

Defaults to `TotalItems` unless `Operation` is `FillTargetItem` in which case `TotalAmount` is used.

Critical bool `false`

If true, the blueprint is hidden from the crafting menu if this input item is missing.

Useful for blueprints that aren't relevant to the player without a specific item. For example, the first input item of a `FillTargetItem` blueprint is often marked `Critical`.

Delete bool `true`

If true, this item is consumed when crafting the blueprint.

If false, this item is marked as a "Tool" in the crafting menu. (For example, a Saw item required to cut wood.)

ID Asset Pointer to Item Asset

This property can also be set to a string value of `this`, which will use the the owning asset's ID. Useful for Salvage blueprints to avoid accidentally writing the wrong ID.

Prioritization ECraftingInputPrioritization

Controls which items are used first. For example, whether to use the lowest quality items first.

Defaults to `LowestQuality` unless `Operation` is `FillTargetItem` in which case `LowestAmount` is used.

CHAPTER

TWENTYFIVE

BLUEPRINT OUTPUT ITEMS

Configuration for a produced item in a *blueprint*.

25.1 Value Format

For straightforward cases, a simple one-liner format is supported:

- {ID}
- {ID} x {Amount}
- this
- this x {Amount}

Examples:

```
// Canned Beans:
78fefdd23def4ab6ac8301adfcc3b2d4

// Or, using Canned Beans' legacy ID:
13

// Two cans of beans:
13 x 2

// Owning asset:
this

// Two of owning asset:
this x 2
```

25.2 Dictionary Format

Unlike the shorter format above, using these properties requires the `{ }` *dictionary* format. For example:

```
OutputItems
[
        {
                // First output item properties
        }
        {
```

(continues on next page)

(continued from previous page)

```
                // Second output item properties
        }
]
```

25.2.1 Properties

Property Name	Type	Default Value
Amount	*uint8*	1
ID	*Asset Pointer* to *Item*	
Origin	*EItemOrigin*	Craft

Amount uint8 1

Quantity of this item created by the blueprint. Minimum value of one.

ID Asset Pointer to Item Asset

This property can also be set to a string value of `this`, which will use the the owning asset's ID. Useful for blueprints creating the owning asset to avoid accidentally writing the wrong ID.

Origin EItemOrigin Craft

A kludge to override certain spawning properties. For example, setting the origin to `Admin` will cause items to spawn at full quality.

CHAPTER

TWENTYSIX

BOX ASSETS

Boxes are created from the ItemBoxAsset class. They are used to visualize certain information from the game's Steam Economy integration. Since this asset is not useful to modders, this documentation merely exists for completeness.

This inherits the *ItemAsset* class.

26.1 Item Asset Properties

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

Type *enum* (`Box`)

ID *uint16*: Must be a unique identifier.

26.2 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.

CHAPTER

TWENTYSEVEN

CALIBER ASSETS

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

27.1 Game Data File

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

27.1.1 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
Ballistic_Drop	*float32*	1
Calibers	*uint8*	`0`
Caliber_#	*uint16*	`0`
Damage	*float32*	*deprecated*
Destroy_Attachment_Colliders	*bool*	`true`
Firerate	*uint8*	`0`
Instantiated_Attachment_Name_Override	*string*	See description
Invulnerable	*bool*	`false`
Paintable	*flag*	
Recoil_X	*float32*	1
Recoil_Y	*float32*	1
Shake	*float32*	1
Spread	*float32*	1
Sway	*float32*	1

27.1.2 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.

Ballistic_Drop float32 `1`

Gravity acceleration multiplier for bullets in flight.

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.

Instantiated_Attachment_Name_Override string

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

For example, a magazine attachment with GUID `dbfb1d0d11ca438e9dffb95f76e61274` will instantiate Magazine.prefab as (Gun)/Magazine/dbfb1d0d11ca438e9dffb95f76e61274 by default. With `Instantiated_Attachment_Name_Override` set to "Example" it would instead spawn as (Gun)/Magazine/Example.

Invulnerable bool `false`

If true, gun can damage entities with Invulnerable tag.

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.

CHAPTER

TWENTYEIGHT

CHARGE ASSETS

Charges (or "remote explosives") are created from the ItemChargeAsset class. They can be placed and then remotely detonated with a *remote trigger*.

This inherits the *BarricadeAsset* class.

28.1 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.

28.2 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.

CHAPTER

TWENTYNINE

CLOTHING ASSETS

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

Items that use a class derived from this class are known as clothing items. Clothing can be worn by players and zombies, although they may function slightly differently between the two.

29.1 Game Data File

The ItemClothingAsset class inherits properties from the *ItemAsset* class. Clothing items will always display a quality value.

29.1.1 Properties

Property Name	Type	Default Value
Armor	*float32*	1
Armor_Explosion	*float32*	See description
Beard_Visible	*bool*	`true`
Destroy_Clothing_Colliders	*bool*	`true`
Falling_Damage_Multiplier	*float32*	1
Hair_Visible	*bool*	`true`
Mirror_Left_Handed_Model	*bool*	`true`
Movement_Speed_Multiplier	*float32*	1
Prevents_Falling_Broken_Bones	*bool*	`false`
Priority_Over_Cosmetic	*bool*	See description
Proof_Fire	*flag*	
Proof_Radiation	*flag*	
Proof_Water	*flag*	
Skin_Override	*string*	
Visible_On_Ragdoll	*bool*	`true`
WearAudio	*Master Bundle Pointer*	See description

29.1.2 Property Descriptions

Armor float32 1

Multiplier on damage received to the covered body part(s). The affected body part(s) will differ depending on the child class.

Armor_Explosion float32

Multiplier on the damage received from area-of-effect explosions. Defaults to the value of `Armor`.

Beard_Visible bool `true`

When `true`, the character's facial hair should be visible (unless disabled by some other condition).

Destroy_Clothing_Colliders bool `true`

When `false`, colliders are *not* destroyed when the clothing is attached to the player character. For example, equipped vanilla clothing do not have any colliders. But some mods (e.g., armor with a hitbox) may have relied on child colliders not being destroyed.

Falling_Damage_Multiplier float32 `1`

Multiplier on damage received from falling.

Hair_Visible bool `true`

When `true`, the character's hair should be visible (unless disabled by some other condition).

Mirror_Left_Handed_Model bool `true`

Clothing should be mirrored when the player is using the left-handed setting. This property only affects 3D clothing items, being: vests, backpacks, masks, glasses, and hats.

Movement_Speed_Multiplier float32 `1`

Multiplier on movement speed.

Prevents_Falling_Broken_Bones bool `false`

When `true`, the player will never receive the Broken Bones debuff from falling.

Priority_Over_Cosmetic bool

This property can be set to override the default cosmetic override behavior. The default behavior will differ depending on the asset. For most assets, cosmetics are displayed over clothing. For glasses using the `Vision` property, the clothing item has priority over the cosmetic.

> **Note**
>
> This only has an effect in PvP mode servers.

Proof_Fire flag

When this flag is included, this clothing item is considered fireproof. When a fireproof shirt and fireproof pants are worn together, the player becomes immune to fire damage. This property has no effect on other clothing types.

Proof_Radiation flag

When this flag is included, this clothing item is considered radiation-proof. When a radiation-proof mask is worn, the player will not be affected by standard deadzones. When radiation-proof pants, a radiation-proof shirt, and a radiation-proof mask are all worn together, the player will not be damaged by full-suit deadzones. This protection only lasts for as long as the radiation-proof mask's item quality is greater than 0%. The mask's quality will deplete over time while inside of a deadzone, but can be replenished with *radiation filters*.

Proof_Water flag

When this flag is included, this clothing item is considered waterproof. When waterproof glasses are worn, the player's screen is no longer blurred when underwater. When waterproof glasses and a waterproof backpack are worn together, the player's oxygen will deplete at a greatly reduced rate while underwater. This property has no effect on other clothing types.

Skin_Override string

Optional name of a renderer that should use the player's skin material. For example, the Conflicting Conscience cosmetic adds miniature versions of the player sitting on their shoulder.

Visible_On_Ragdoll bool `true`

When `true`, this clothing item should be visible on ragdolls.

WearAudio Master Bundle Pointer

AudioClip or OneShotAudioDefinition to play upon equipping this clothing item. Default value is dependent on the child asset. Backpacks and vests will use `Sounds/Zipper.mp3` by default. Otherwise, `Sounds/Sleeve.mp3` is used.

CHAPTER

THIRTY

CLOUD ASSETS

Clouds (or "parachutes") are created from the ItemCloudAsset class. They can affect a player's gravity when held.

This inherits the *ItemAsset* class.

30.1 Item Asset Properties

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

Type *enum* (`Cloud`)

Useable *enum* (`Cloud`)

ID *uint16*: Must be a unique identifier.

30.2 Cloud Asset Properties

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

CHAPTER

THIRTYONE

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.

31.1 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.

Randomize_Consume_Audio_Pitch *bool*: If false, `ConsumeAudioClip` always plays with `1.0` pitch. Defaults to true.

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.

31.2 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`.

CHAPTER

THIRTYTWO

DETONATOR ASSETS

Detonators (or "remote triggers") are created from the ItemDetonatorAsset class. They can be used to detonate *remote explosives*.

This inherits the *ItemAsset* class.

32.1 Item Asset Properties

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

Type *enum* (`Detonator`)

Useable *enum* (`Detonator`)

ID *uint16*: Must be a unique identifier.

32.2 Detonator Asset Properties

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

CHAPTER

THIRTYTHREE

FARM ASSETS

Farms (or "plants") are created from the ItemFarmAsset class. They are 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.

33.1 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.

33.2 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.

CHAPTER

THIRTYFOUR

FILTER ASSETS

Filters (or "radiation filters") are created from the ItemFilterAsset class. They can be used to replenish the quality of radiation-proof *masks*.

This inherits the *ItemAsset* class.

34.1 Item Asset Properties

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

Type *enum* (`Filter`)

Useable *enum* (`Filter`)

ID *uint16*: Must be a unique identifier.

34.2 Filter Asset Properties

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

CHAPTER

THIRTYFIVE

FISHER ASSETS

Fishers (or "fishing poles") are created from the ItemFisherAsset class. They are useables that allow for catching fish.

This inherits the *ItemAsset* class.

35.1 Item Asset Properties

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

Type *enum* (`Fisher`)

Useable *enum* (`Fisher`)

ID *uint16*: Must be a unique identifier.

35.2 Fisher Asset Properties

Reward_Experience_Min *int32*: Minimum amount (inclusive) of experience to grant upon successfully catching something. Defaults to 3.

Reward_Experience_Max *int32*: Maximum amount (inclusive) of experience to grant upon successfully catching something. Defaults to 3.

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

Fishing poles can use quest rewards. Refer to *Rewards* documentation for additional documentation. These rewards are prefixed with `Quest_`. For example, `Quest_Rewards 1`.

Fish_Bite_Interval_Multiplier *float*: Multiplier for interval before a fish takes the bait. Defaults to 1.

FishingRewardMode *enum*: `Rod` or `WaterVolumes`. Defaults to `Rod` for backwards compatibility.

`Rod`: Fishing rod itself defines rewards. Ignore per-volume rewards.

`WaterVolumes`: Use per-volume (or per-level if volume unspecified) rewards. If level doesn't support volume rewards, fallback to `Rod` rewards.

See also

Level Asset fishing properties.

CatchChallenge_Enabled *bool*: If true, player must complete a challenge when fish takes the bait before catching. Defaults to false for backwards compatibility.

CatchChallenge_CursorSize *float*: Size of window item must be within to catch. Defaults to 0.2.

CatchChallenge_Gravity *float*: Downward acceleration while input is not held. Defaults to 1.

CatchChallenge_Acceleration *float*: Upward acceleration while input is held. Defaults to 1.

CatchChallenge_UpperRestitution *float*: How much velocity to preserve when bouncing off the top. Defaults to 0.5 (50%).

CatchChallenge_LowerRestitution *float*: How much velocity to preserve when bouncing off the bottom. Defaults to 0.5 (50%).

CatchChallenge_CaptureSpeed *float*: Multiplier for how quickly item is caught while within cursor. Defaults to 1 (100%).

CatchChallenge_EscapeSpeed *float*: Multiplier for how quickly the player loses while item is outside the cursor. Defaults to 1 (100%).

CHAPTER

THIRTYSIX

FISHING CATCHABLE PROPERTIES

These options are specified in an item's *Fishing_Catchable Dictionary*.

Capture_Duration *float*: How long the player must keep the item within the cursor before it's caught. Defaults to 2 seconds.

Escape_Duration *float*: How long before the item gets away. Defaults to 2 seconds.

Spring_Stiffness *float*: Scales acceleration toward target position (without exceeding limits). Defaults to 16.

Spring_Damping *float*: Reduces speed toward target position. Defaults to 4.

Min_Relocate_Interval *float*: Minimum seconds before target position changes. Defaults to 1.5 seconds.

Max_Relocate_Interval *float*: Maximum seconds before target position changes. Defaults to 2 seconds.

Max_Upward_Acceleration *float*: Upward acceleration cannot increase beyond this limit. Defaults to 1.5.

Max_Downward_Acceleration *float*: Downward acceleration cannot increase beyond this limit. Defaults to 1.2.

Max_Upward_Speed *float*: Upward speed cannot increase beyond this limit. Defaults to 0.6.

Max_Downward_Speed *float*: Downward speed cannot increase beyond this limit. Defaults to 0.45.

Upper_Restitution *float*: How much velocity to preserve when bouncing off the top. Defaults to 0.6 (60%).

Lower_Restitution *float*: How much velocity to preserve when bouncing off the bottom. Defaults to 0.4 (40%).

Min_Target_Delta *float*: When choosing a new position, it will be at least this far away. Defaults to 0.3

Max_Target_Delta *float*: When choosing a new position, it will be at most this far away. Defaults to 0.4.

Min_Target_Position *float*: [0, 1] Minimum choosable position. Defaults to 0.1.

Max_Target_Position *float*: [0, 1] Maximum choosable position. Defaults to 0.9.

Tip

For example, the vanilla Lobster's configuration is:

```
Fishing_Catchable
{
        Min_Relocate_Interval 0.3
        Max_Relocate_Interval 1
        Max_Upward_Acceleration 1.3
        Max_Downward_Acceleration 1.75
        Max_Upward_Speed 0.5
        Max_Downward_Speed 1.2
        Upper_Restitution 0
        Lower_Restitution 2
```

```
        Min_Target_Delta 0.35
        Max_Target_Delta 0.4
        Min_Target_Position 0
        Max_Target_Position 0.4
        Capture_Duration 2.75
        Escape_Duration 1.25
        Spring_Stiffness 20
        Spring_Damping 4
}
```

CHAPTER

THIRTYSEVEN

FOOD ASSETS

Food items are created from the ItemFoodAsset class. They can be consumed by the player.

This inherits the *ConsumeableAsset* class.

37.1 Item Asset Properties

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

Type *enum* (`Food`)

Useable *enum* (`Consumeable`)

ID *uint16*: Must be a unique identifier.

37.2 Food Asset Properties

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

CHAPTER

THIRTYEIGHT

FUEL ASSETS

Fuel items (or "fuel canisters") are created from the ItemFuelAsset class. They are useables able to siphon, store, and deposit fuel.

This inherits the *ItemAsset* class.

38.1 Item Asset Properties

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

Type *enum* (`Fuel`)

Useable *enum* (`Fuel`)

ID *uint16*: Must be a unique identifier.

38.2 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.

CHAPTER

THIRTYNINE

GEAR ASSETS

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

39.1 Game Data File

The ItemGearAsset class inherits properties from the *ItemClothingAsset* class.

39.1.1 Properties

Property Name	Type	Default Value
Beard	*flag*	
Beard_Override	*string*	
Beard_Override_NonGoldColor	*Color*	
Hair	*flag*	
Hair_Override	*string*	
Hair_Override_NonGoldColor	*Color*	

Some inherited properties behave differently when used by this class. Notably, these are:

1. *Beard_Visible* (from *ItemClothingAsset*). This property's default behavior is altered. Refer to this class's `Beard` property instead.
2. *Hair_Visible* (from *ItemClothingAsset*). This property's default behavior is altered. Refer to this class's `Hair` property instead.

39.1.2 Property Descriptions

Beard flag

When this flag is not included, the parent class's *Beard_Visible* property is set to false. This flag must be included if the character's facial hair should be visible.

Beard_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 hbeardair material. This property is used by certain cosmetics that entirely cover the character's beard, so that the player's selected beard color can still be used for customization.

Beard_Override_NonGoldColor Color

For items using BeardOverride, the beard material color will default to this for players without the Gold Upgrade. (Since the Gold Upgrade is required for full RGB control, the default beard colors may look boring for items that cover the beard but aren't beards in of themselves.) Also used as the color in the cosmetic preview.

Hair flag

When this flag is not included, the parent class's *Hair_Visible* property is set to false. This flag must be included if the character's 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.

Hair_Override_NonGoldColor Color

For items using hairOverride, the hair material color will default to this for players without the Gold Upgrade. (Since the Gold Upgrade is required for full RGB control, the default hair colors may look boring for items that cover the hair but aren't hair in of themselves.) Also used as the color in the cosmetic preview.

CHAPTER

FORTY

GENERATOR ASSETS

Generators are created from the ItemGeneratorAsset class. They are a placeable power sources. Players can deposit fuel into generators with a *fuel canister*.

This inherits the *BarricadeAsset* class.

40.1 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.

40.2 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.

CHAPTER

FORTYONE

GLASSES ASSETS

Glasses are created from ItemGlassesAsset. They are clothing items that can be worn by players and zombies.

This inherits the *GearAsset* class.

41.1 Item Asset Properties

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

Type *enum* (`Glasses`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

41.2 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 *float32*: Intensity of fog while nightvision is active.

Vision *ELightingVision*: Determines the type of lighting vision to use. When looking to assign a custom nightvision color via the `Nightvision_Color` property, you should use the `Military` enumerator. When the `Headlamp` enumerator is used, you can also specify properties from the *PlayerSpotLightConfig* struct. Defaults to `None`.

CHAPTER

FORTYTWO

GRIP ASSETS

Grips (or "grip attachments") are created from the ItemGripAsset class. They are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

42.1 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`

42.1.1 Properties

Property Name	Type	Default Value
Bipod	*flag*	

42.1.2 Property Descriptions

Bipod flag

Stat-changing properties should only take effect while prone.

CHAPTER

FORTYTHREE

GROWER ASSETS

Growers (or "growth supplements") are created from the ItemGrowerAsset class. They can be used to instantly finish growing a *plant*.

This inherits the *ItemAsset* class.

43.1 Item Asset Properties

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

Type *enum* (`Grower`)

Useable *enum* (`Grower`)

ID *uint16*: Must be a unique identifier.

43.2 Grower Asset Properties

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

CHAPTER

FORTYFOUR

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.

44.1 Unity Asset Bundle Contents

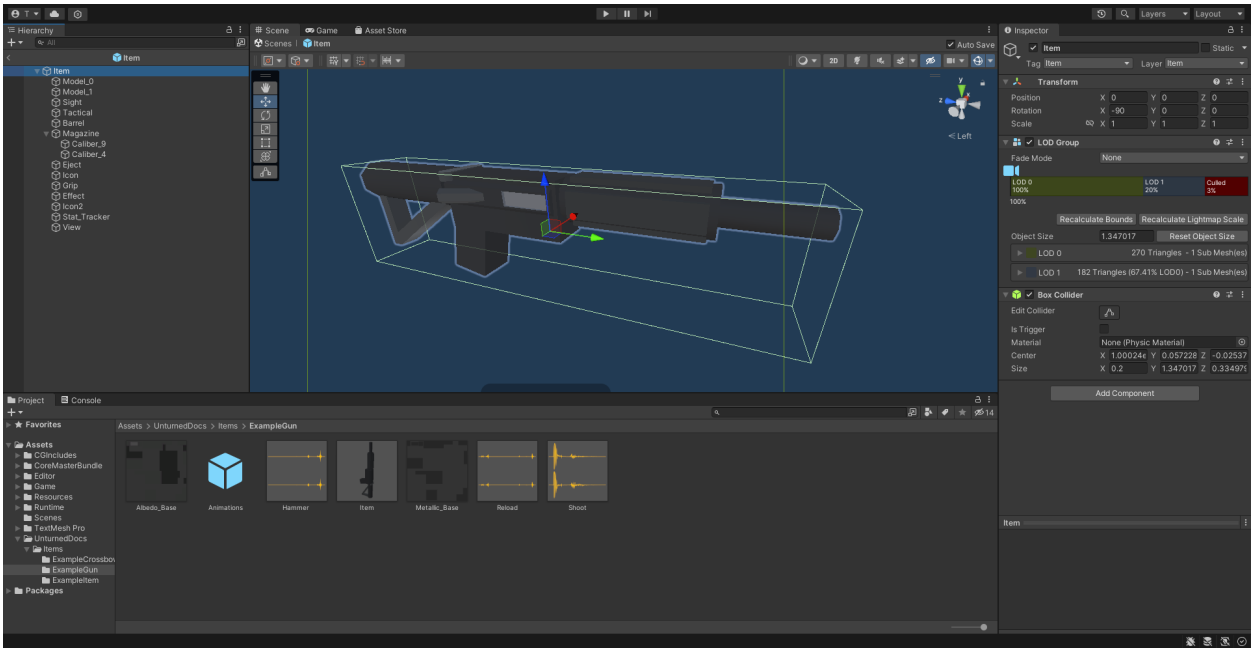


Fig. 1: An example of a gun being set up in the Unity editor.

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.

44.1.1 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. (This is supported for the other attachment types as well.)

Additional Setup for Bows

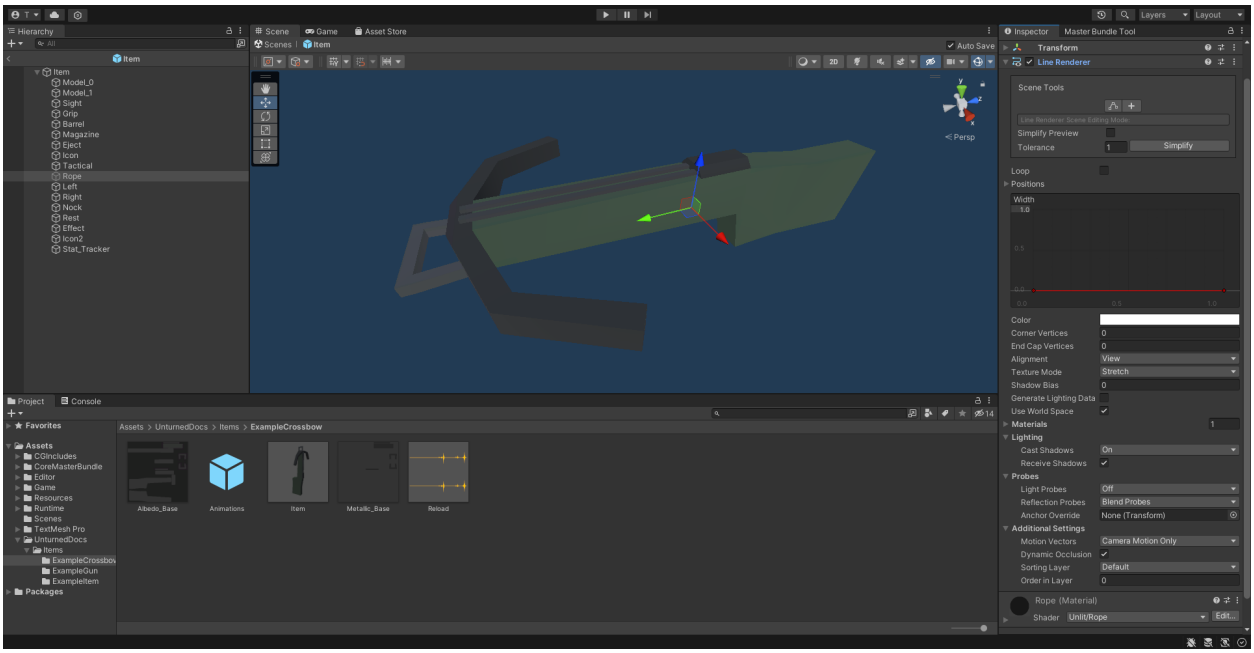


Fig. 2: An example of a crossbow being set up in the Unity editor.

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.

44.1.2 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.

44.1.3 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.

44.2 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.

44.2.1 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 1: 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
Must_Aim_To_Shoot	*bool*	See description
Range_Rangefinder	*float32*	See description
Scale_Aim_Animation_Speed	*bool*	`true`
Stop_Aiming_After_Shooting	*bool*	`false`
DriverTurretViewmodelMode	*EDriverTurretViewmodelMode*	`OffscreenWhileAiming`

Table 2: 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 3: 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 4: Effects

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

Table 5: 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 6: 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 7: 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 8: 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 9: 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 10: 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 11: Rechambering Properties

Property Name	Type	Default Value
CasingEjectCountAfterRechamberingAfterShooting	*int32*	`1`
CasingEjectCountAfterReload	*int32*	See description
EjectAfterHammerDelay	*float32*	`0.45`
EjectAfterReloadDelay	*float32*	`0.5`
EjectCasingAfterShooting	*bool*	See description
RechamberAfterMagazineAttached	*ERechamberGunAfterReloadMode*	`IfAmmoWasEmpty`
RechamberAfterMagazineDetached	*ERechamberGunAfterReloadMode*	`Always`
RechamberAfterShotCount	*int32*	See description
RechamberAfterShotDelay	*float32*	`0.25`

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_Midair	*float32*	`1.0`
Recoil_Prone	*float32*	`0.7`
Recoil_Sprint	*float32*	`1.25`
Recoil_Swimming	*float32*	`1.1`
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_Midair	*float32*	`1.5`
Spread_Prone	*float32*	`0.7`
Spread_Sprint	*float32*	`1.25`
Spread_Swimming	*float32*	`1.1`

44.2.2 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.

44.2.3 EDriverTurretViewmodelMode Enumeration

Named Value	Description
`OffscreenWhileAiming`	Default. Pushes first-person arms off-screen while aiming. Originally implemented for the Fighter Jet where it looks weird if your arms are still visible when the camera zooms in while "aiming."
`AlwaysOffscreen`	Push first-person arms off-screen when equipped.
`AlwaysOnscreen`	Included for completeness.

44.2.4 ERechamberGunAfterReloadMode Enumeration

Named Value	Description
`IfAmmoWasEmpty`	Default. Plays "Hammer" animation if ammo count was zero.
`Never`	Regardless of ammo, does not play "Hammer" animation after reloading.
`Always`	Regardless of ammo, will play "Hammer" animation after reloading.

44.2.5 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. Running the game with the `-LogBallisticDropConversion` *launch option* will log the equivalent `Bullet_Gravity_Multiplier` value.

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.

CasingEjectCountAfterRechamberingAfterShooting int32 `1`

Only applicable if *RechamberAfterShotCount* is non-zero.

If greater than zero, emit this many bullet casing particles after *EjectAfterHammerDelay* seconds pass.

CasingEjectCountAfterReload int32

If greater than zero, emit this many bullet casing particles after *EjectAfterReloadDelay* seconds pass.

Defaults to *Ammo_Max* for `Break` Action guns. Zero otherwise.

EjectCasingAfterShooting bool

If true, emit a bullet casing particle when a shot is fired.

Defaults to true for `Trigger` and `Minigun` action guns.

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.

EjectAfterHammerDelay float32 `0.45`

How long in seconds after hammering to eject a bullet casing.

Only applicable if *RechamberAfterShotCount* is non-zero.

EjectAfterReloadDelay float32 `0.5`

How long in seconds after reloading to eject bullet casings.

Only applicable if *CasingEjectCountAfterReload* is greater than zero.

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`.

Must_Aim_To_Shoot bool

If true, the gun cannot shoot unless the player is aiming.

Defaults to true for `Minigun` *Action EAction*. False otherwise.

`String` *Action EAction* overrides `Must_Aim_To_Shoot` functionality.

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.

RechamberAfterMagazineAttached ERechamberGunAfterReloadMode `IfAmmoWasEmpty`

Determines whether "Hammer" animation plays after attaching a magazine. This happens when a magazine replaces another OR fills previously empty slot.

RechamberAfterMagazineDetached ERechamberGunAfterReloadMode `Always`

Determines whether "Hammer" animation plays after detached a magazine. This happens when a magazine is removed from the gun without a replacement.

RechamberAfterShotCount int32

If non-zero, hammer animation plays after shooting after shooting this many shots and *RechamberAfterShotDelay*: seconds pass.

Shot count resets after reloading, hammering, or dequipping the gun.

Defaults to `1` for `Bolt` and `Pump` Action guns. Zero otherwise.

RechamberAfterShotDelay float32 `0.25`

How long in seconds after firing to rechamber the gun by playing the Hammer animation.

Only applicable if *RechamberAfterShotCount* is non-zero.

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_Midair float32 `1.0`

Multiplier on camera recoil while jumping and/or falling.

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`.

Recoil_Swimming float32 `1.1`

Multiplier on camera recoil while swimming.

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 `-LogGunSpreadConversion` *launch option* will log the equivalent `Spread_Angle_Degrees` value.

Spread_Midair float32 `1.5`

Multiplier on the bullet spread while jumping and/or falling.

Spread_Prone float32 `0.7`

Multiplier on the bullet spread while prone.

Spread_Sprint float32 `1.25`

Multiplier on the bullet spread while sprinting.

Spread_Swimming float32 `1.1`

Multiplier on the bullet spread while swimming.

Stop_Aiming_After_Shooting bool `false`

If true, the gun will stop aiming regardless of player input.

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.

DriverTurretViewmodelMode EDriverTurretViewmodelMode

Controls how first-person arms are moved for turrets operated from the driver's seat.

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.

44.3 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`.

44.4 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*.

CHAPTER

FORTYFIVE

HAT ASSETS

The ItemHatAsset class is used by clothing items occupying the "hat" slot. Hats can be worn by players and zombies, and cover their head.

45.1 Game Data File

The ItemHatAsset class inherits properties from the *ItemGearAsset* class. Properties that are required (or recommended) are listed in the table below.

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

45.1.1 Properties

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

CHAPTER

FORTYSIX

KEY ASSETS

Keys are created from the ItemKeyAsset class. They 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.

46.1 Item Asset Properties

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

Type *enum* (`Key`)

ID *uint16*: Must be a unique identifier.

46.2 Key Asset Properties

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

CHAPTER

FORTYSEVEN

LIBRARY ASSETS

Libraries are created from the ItemLibraryAsset class. They are placeable storage containers for experience points.

This inherits the *BarricadeAsset* class.

47.1 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.

47.2 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.

CHAPTER

FORTYEIGHT

MAGAZINE ASSETS

Magazines (or "magazine attachments") are created from the ItemMagazineAsset class. They can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

48.1 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`

48.1.1 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
Explosion_Penetrate_Buildables	*bool*	`false`
Explosion_Plays_Impact_Effects	*bool*	`true`
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`

48.1.2 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`.

Explosion_Penetrate_Buildables bool `false`

If true, explosion damage passes through buildable items.

Explosion_Plays_Impact_Effects bool `true`

If true, per-surface effects like blood splatter are created. Defaults to true, but can be disabled particularly if a performance issue (e.g., an explosive many-pellet shotgun shell).

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.

48.2 Unity_Setup

Projectile.prefab: Optional for projectile-launching guns. If included, overrides the projectile instantiated when the gun is fired.

CHAPTER

FORTYNINE

MAP ASSETS

Maps and compasses are created from the ItemMapAsset class. They 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.

49.1 Item Asset Properties

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

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

ID *uint16*: Must be a unique identifier.

49.2 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.

CHAPTER

FIFTY

MASK ASSETS

Masks are created from the ItemMaskAsset class. They can be worn by players and zombies.

This inherits the *GearAsset* class.

50.1 Item Asset Properties

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

Type *enum* (`Mask`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

50.2 Mask Asset Properties

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

FilterDegradationRateMultiplier *float32*: Multiplier for how quickly deadzones deplete a gasmask's filter quality. For example, 2 is faster (2x) and 0.5 is slower.

CHAPTER

FIFTYONE

MEDICAL ASSETS

Medical items (or "medicine") are created from the ItemMedicalAsset class. They are irreversibly consumed by the player on use, and directly affect a player's stats such as health or immunity.

This inherits the *ConsumeableAsset* class.

51.1 Item Asset Properties

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

Type *enum* (`Medical`)

Useable *enum* (`Consumeable`)

ID *uint16*: Must be a unique identifier.

51.2 Medical Asset Properties

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

CHAPTER

FIFTYTWO

MELEE ASSETS

Melees (or "melee weapons") are created from the ItemMeleeAsset class. They can be used as a source of damage. Melee weapons always show quality.

This inherits the *WeaponAsset* class.

52.1 Item Asset Properties

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

Type *enum* (`Melee`)

Useable *enum* (`Melee`)

Slot *ESlotType* (`None`, `Primary`, `Secondary`, `Any`): Most melee weapons, including vanilla items, use `Secondary`. This allows the weapon to be used from either the character's primary or secondary slot.

ID *uint16*: Must be a unique identifier.

52.2 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.

52.3 NPC Rewards

For more information, refer to the *Rewards* documentation.

Rewards can be granted for weak and/or strong attacks. `Weak_Attack_Quest_Rewards # with Weak_Attack_Quest_Reward_` prefix or `Strong_Attack_Quest_Rewards # with Strong_Attack_Quest_Reward_` prefix.

CHAPTER

FIFTYTHREE

OIL PUMP ASSETS

Oil pumps are created from the ItemOilPumpAsset class. They are placeables capable of creating fuel. When powered, oil pumps generate fuel over time.

This inherits the *BarricadeAsset* class.

53.1 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.

53.2 Oil Pump Asset Properties

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

CHAPTER

FIFTYFOUR

OPTIC ASSETS

Optics are created from the ItemOpticAsset class. They can modify a player's view.

This inherits the *ItemAsset* class.

54.1 Item Asset Properties

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

Type *enum* (`Optic`)

Useable *enum* (`Optic`)

ID *uint16*: Must be a unique identifier.

54.2 Sight Asset Properties

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

CHAPTER

FIFTYFIVE

PANTS ASSETS

Pants are created from the ItemPantsAsset class. They can be worn by players and zombies.

This inherits the *BagAsset* class.

55.1 Item Asset Properties

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

Type *enum* (`Pants`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

55.2 Pants Asset Properties

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

CHAPTER

FIFTYSIX

PLACEABLE ASSETS

The ItemPlaceableAsset class is a base class that other classes are derived from. Placeables are able to be placed by players.

This inherits the *ItemAsset* class.

56.1 Placeable Asset Properties

ExplosionEffect_CopyModelPosition *bool*: If true, effect spawns exactly at the model position without any offset. Defaults to false for backwards compatibility.

ExplosionEffect_CopyModelRotation *bool*: If true, effects spawns with same rotation as the model. Defaults to false for backwards compatibility.

Item_Dropped_On_Destroy *Asset Pointer*: Item asset or spawn table for items dropped when destroyed.

This property can also be set to a string value of `this`, which will use the the owning item's GUID. Useful to avoid accidentally writing the wrong ID.

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.

Items_Dropped_On_Destroy *int*: Shorthand for setting both `Min_Items_Dropped_On_Destroy` and `Max_Items_Dropped_On_Destroy`.

Min_Items_Recovered_On_Salvage *int*: Minimum number of items to receive when picked up below 100% health. Defaults to 1.

Max_Items_Recovered_On_Salvage *int*: Maximum number of items to receive when picked up below 100% health. Defaults to 1.

Items_Recovered_On_Salvage *int*: Shorthand for setting both `Min_Items_Recovered_On_Salvage` and `Max_Items_Recovered_On_Salvage`.

Min_Items_Recovered_On_Salvage_Full_Health *int*: Minimum number of items to receive when picked up at 100% health. Defaults to 1.

Max_Items_Recovered_On_Salvage_Full_Health *int*: Maximum number of items to receive when picked up at 100% health. Defaults to 1.

Items_Recovered_On_Salvage_Full_Health *int*: Shorthand for setting both `Min_Items_Recovered_On_Salvage_Full_Health` and `Max_Items_Recovered_On_Salvage_Full_Health`.

PlaceableProvidesCraftingTags list of *Asset Pointer*: *Tag Assets* available to nearby players for blueprint requirements. Tags are listed in the item description as "crafting capabilities."

For example, the vanilla Brick Oven provides two tags:

```
PlaceableProvidesCraftingTags
[
        // Heat Source (for backwards compatibility)
        20f30322bbcc4b01a4f116d22b24c21a
        // Enclosed Heat Source
        d2cc65b749e5477f95103601df89cdbc
]
```

SalvageItem *Asset Pointer*: Override the default salvaging behavior by pointing to a specific item or spawn table that should be added when salvaging a placeable that is below 100% health. By default, this property will choose a random item used in the placeable's blueprints.

SalvageItem_FullHealth *Asset Pointer*: Defaults to self. Overrides the default pickup behavior by pointing to a specific item or spawn table that should be added when salvaging a placeable that is at 100% health.

This property can also be set to a string value of `this`, which will use the the owning item's GUID. Useful to avoid accidentally writing the wrong ID.

CHAPTER

FIFTYSEVEN

REFILL ASSETS

Refills (localized as "water canisters") are created from the ItemRefillAsset class. They 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.

57.1 Item Asset Properties

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

Type *enum* (`Refill`)

Useable *enum* (`Refill`)

ID *uint16*: Must be a unique identifier.

57.2 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.

CHAPTER

FIFTYEIGHT

SENTRY ASSETS

Sentries (localized as "robotic turrets") are created from the ItemSentryAsset class. They 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.

58.1 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.

58.2 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_Animals *bool*: If true, this sentry can attack animals. Defaults to true.

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`).

Target_Players *bool*: If true, this sentry can attack players. Defaults to true.

Target_Vehicles *bool*: If true, this sentry can attack vehicles. Defaults to true.

Target_Zombies *bool*: If true, this sentry can attack zombies. Defaults to true.

Sentry_Bypasses_PvE *bool*: If true, sentry can damage players and vehicles in PvE mode. Defaults to false.

React_To_Attacks *bool*: If true, sentry immediately focuses on attacking player. Defaults to false.

Sweep_Yaw *float*: Yaw range the sentry sweeps left/right. Defaults to 120.

Sweep_Period *float*: How long in seconds for the sentry to complete a sweep from left to right and back. Defaults to 6.3 seconds.

CHAPTER

FIFTYNINE

SHIRT ASSETS

Shirts are created from the ItemShirtAsset class. They can be worn by players and zombies.

This inherits the *BagAsset* class.

59.1 Item Asset Properties

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

Type *enum* (`Shirt`)

Useable *enum* (`Clothing`)

ID *uint16*: Must be a unique identifier.

59.2 Shirt Asset Properties

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

59.3 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.

CHAPTER

SIXTY

SIGHT ASSETS

Sights (or "sight attachments") are created from the ItemSightAsset class. They are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

60.1 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`

60.1.1 Properties

Property Name	Type	Default Value
AimAlignment_LocalOffset	*Vector3*	`(0, 0, 0)`
AimAlignment_Owner	*EAimAlignmentTransformOwner*	`Sight`
AimAlignment_Path	*string*	`Model_0/Aim`
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`

60.1.2 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`

60.1.3 EAimAlignmentTransformOwner Enumeration

Named Value	Description
`Sight`	Look for aim alignment transform relative to sight model. Defaults to Model_0/Aim.
`Gun`	Look for aim alignment transform relative to equipable prefab. Requires setting AimAlignment_Path.

60.1.4 ESide Enumeration

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

60.1.5 Property Descriptions

AimAlignment_LocalOffset Vector3 `(0, 0, 0)`

Position offset relative to Aim transform or transform specified by `AimAlignment_Path`.

AimAlignment_Owner EAimAlignmentTransformOwner `Sight`

Controls where to find `AimAlignment_Path`.

AimAlignment_Path string `Model_0/Aim`

While aiming down sights, the camera's position is aligned with this transform. Relative to `AimAlignment_Owner`.

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.

60.1.6 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.

CHAPTER

SIXTYONE

STORAGE ASSETS

Storages (localized as "item storages") are created from the ItemStorageAsset class. They are placeables used to store items.

This inherits the *BarricadeAsset* class.

61.1 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.

61.2 Storage Asset Properties

Can_Players_Open *bool*: If true, players can interact with the barricade to open storage. Defaults to true. Useful for pre-placed sentries to prevent stealing their guns.

Delete_Contained_Items_On_Destroy *bool*: If true, any stored items are despawned rather than dropped when destroyed. Defaults to false.

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.

Default_Contained_Items *list of dictionaryies*: Items to create inside storage when first spawned. Each item can contain the following properties:

- **Asset**: *Asset Pointer*: Item or spawn table to grant an item from.
- **Amount** *int*: Number of times to grant this item. Defaults to 1.
- **Origin** *EItemOrigin*: Determines starting state of the item. Defaults to World.

CHAPTER

SIXTYTWO

STRUCTURE ASSETS

Structures are created from the ItemStructureAsset class. They can be placed by players. Some structure pieces require another structure piece in order to be placed.

This inherits the *PlaceableAsset* class.

62.1 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.

62.2 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.

Can_Zombies_Target *bool*: If true, this item is eligible for zombies to detect and attack when stuck. 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.

CHAPTER

SIXTYTHREE

SUPPLY ASSETS

Supplies (or "crafting supplies") are created from the ItemSupplyAsset class. They are items primarily intended to be used as ingredients in crafting blueprints. They can neither be held nor equipped.

This inherits the *ItemAsset* class.

63.1 Item Asset Properties

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

Type *enum* (`Supply`)

ID *uint16*: Must be a unique identifier.

63.2 Supply Asset Properties

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

CHAPTER

SIXTYFOUR

TACTICAL ASSETS

Tacticals (or "tactical attachments") are created from the ItemTacticalAsset class. They are inventory items that can be attached to ranged weapons.

This inherits the *CaliberAsset* class.

64.1 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`

64.1.1 Properties

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

64.1.2 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.

64.1.3 Melee-Specific Property Descriptions

> **Note**
>
> These properties are only applicable if the `Melee` flag is set.

> **Warning**
>
> This is mostly copied 1:1 from *Weapon Assets* and should be tidied up if/when that is updated.

Melee_Range *float32*: The maximum distance in meters before damage is no longer possible.

Melee_Player_Damage *float32*: Amount of damage that should be dealt to player entities, prior to modifiers such as limb multipliers.

Melee_Player_Leg_Multiplier *float32*: Multiplier on damage targeted against a player's legs.

Melee_Player_Arm_Multiplier *float32*: Multiplier on damage targeted against a player's arms.

Melee_Player_Spine_Multiplier *float32*: Multiplier on damage targeted against a player's torso.

Melee_Player_Skull_Multiplier *float32*: Multiplier on damage targeted against a player's head.

Melee_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.

Melee_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.

Melee_Zombie_Damage *float32*: Amount of damage that should be dealt to zombie entities, prior to modifiers such as limb multipliers.

Melee_Zombie_Leg_Multiplier *float32*: Multiplier on damage targeted against a zombie's legs.

Melee_Zombie_Arm_Multiplier *float32*: Multiplier on damage targeted against a zombie's arms.

Melee_Zombie_Spine_Multiplier *float32*: Multiplier on damage targeted against a zombie's torso.

Melee_Zombie_Skull_Multiplier *float32*: Multiplier on damage targeted against a zombie's head.

Melee_Zombie_Ragdoll_Force_Multiplier *float32*: Scales force applied to zombie ragdoll. Defaults to 1.

Melee_Stun_Zombie_Always *flag*: Specified if a zombie should always be stunned when targeted by the weapon.

Melee_Stun_Zombie_Never *flag*: Specified if a zombie should never be stunned when targeted by the weapon.

Melee_Animal_Damage *float32*: Amount of damage that should be dealt to animal entities, prior to modifiers such as limb multipliers.

Melee_Animal_Leg_Multiplier *float32*: Multiplier on damage targeted against a animal's limbs.

Melee_Animal_Spine_Multiplier *float32*: Multiplier on damage targeted against a animal's torso.

Melee_Animal_Skull_Multiplier *float32*: Multiplier on damage targeted against a animal's head.

CHAPTER

SIXTYFIVE

TANK ASSETS

Tanks (localized as "liquid storages") are created from the ItemTankAsset class. They 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.

65.1 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.

65.2 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.

CHAPTER

SIXTYSIX

THROWABLE ASSETS

Throwables are created from the ItemThrowableAsset class. They can be thrown by players. Throwables cannot be used in any safezones that disallow weapons.

This inherits the *WeaponAsset* class.

66.1 Item Asset Properties

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

Type *enum* (`Throwable`)

Useable *enum* (`Throwable`)

ID *uint16*: Must be a unique identifier.

66.2 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.

CHAPTER

SIXTYSEVEN

TIRE ASSETS

Tires (localized as "tools") are created from the ItemTireAsset class. They are useables that allow for adding and removing tires from vehicles.

This inherits the *VehicleRepairToolAsset* class.

67.1 Item Asset Properties

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

Type *enum* (`Tire`)

Useable *enum* (`Tire`)

ID *uint16*: Must be a unique identifier.

67.2 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.

CHAPTER

SIXTYEIGHT

TOOL ASSETS

Tools are created from the ItemToolAsset class. Their functionality depends on the configured `Useable` property.

This inherits the *ItemAsset* class.

68.1 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.

68.2 Tool Asset Properties

Tools have no unique asset properties. Instead, refer to `Useable` for relevant configuration options. Refer to parent classes for additional properties.

CHAPTER

SIXTYNINE

TRAP ASSETS

Traps are created from ItemTrapAsset. They are placeable damage sources.

This inherits the *BarricadeAsset* class.

69.1 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.

69.2 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.

Requires_Power *bool*: Whether or not the trap requires power from a generator. Defaults to `false`. Optionally, a "Powered" GameObject can be included in Unity. This GameObject is activated when powered, and deactivated when not powered.

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.

CHAPTER

SEVENTY

VEHICLE LOCKPICK TOOL ASSETS

Vehicle lockpick tools are created from the ItemVehicleLockpickToolAsset class. They are useables for unlocking a vehicle.

This inherits the *ToolAsset* class.

70.1 Vehicle Lockpick Tool Asset Properties

FailureProbability *float*: Normalized percentage chance of lockpicking failing. Defaults to zero. The "Use_Failure" animation is played instead of "Use" if lockpicking fails.

FailureEffect *Asset Pointer*: Effect to play when lockpicking fails.

CHAPTER

SEVENTYONE

VEHICLE PAINT TOOL ASSETS

Vehicle paint tools are created from the ItemVehiclePaintToolAsset class. They are useables for changing the vehicle paint color.

This inherits the *ToolAsset* class.

71.1 Item Asset Properties

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

Type *enum* (`Vehicle_Paint_Tool`)

Useable *enum* (`Vehicle_Paint`)

ID *uint16*: Must be a unique identifier.

71.2 Vehicle Paint Tool Asset Properties

PaintColor *color*: Vehicle's color will be replaced with this when used.

CHAPTER

SEVENTYTWO

VEHICLE REPAIR TOOL ASSETS

Vehicle repair tools (localized as "tools") are created from the ItemVehicleRepairTool class. They are useables for replacing vehicle batteries.

This inherits the *ToolAsset* class.

72.1 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.

72.2 Vehicle Repair Tool Asset Properties

Vehicle repair tools have no unique asset properties. Refer to parent classes for additional properties.

CHAPTER

SEVENTYTHREE

VEST ASSETS

The ItemVestAsset class is used by clothing items occupying the "vest" slot. Vests can be worn by players and zombies, and cover their torso.

73.1 Game Data File

The ItemVestAsset class inherits properties from the *ItemBagAsset* class. Properties that are required (or recommended) are listed in the table below.

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

73.1.1 Properties

Vests have no unique properties. Refer to parent classes for additional properties instead.

CHAPTER

SEVENTYFOUR

WATER ASSETS

Drinks are created from the ItemWaterAsset class. They are consumed by the player on use, and directly affect a player's stats such as water or stamina.

This inherits the *ConsumeableAsset* class.

74.1 Item Asset Properties

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

Type *enum* (`Water`)

Useable *enum* (`Consumeable`)

ID *uint16*: Must be a unique identifier.

74.2 Water Asset Properties

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

CHAPTER

SEVENTYFIVE

WEAPON ASSETS

The ItemWeaponAsset class is a base class that other classes are derived from. This asset provides various properties related to damaging players, structures, and other entities – but its specific behavior depends on the child class being used.

This inherits the *ItemAsset* class.

75.1 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.

75.1.1 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).

75.1.2 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.

Zombie_Ragdoll_Force_Multiplier *float32*: Scales force applied to zombie ragdoll. Defaults to 1.

75.1.3 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.

75.1.4 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

SEVENTYSIX

VEHICLE ASSETS

The **VehicleAsset** class is used by vehicles. These can be driven by players, have support for gun turrets, can function as a storage container for items, and more.

76.1 Game Data File

The `GUID` and `Type` properties are required by all vehicle assets. Many vehicle assets will want to include `Engine` and `Rarity` as well. The `ID` property used to be required, but this is no longer necessary.

Any properties from parent classes that are required—or recommended—are listed in the table below.

Class	Property Name	Required Value
Asset	*GUID*	
Asset	*ID*	
Asset	*Type*	`Vehicle`
VehicleAsset	*Engine*	

76.1.1 Properties

Table 1: Uncategorized

Property Name	Type	Default Value
AdditionalTransparentSections	*list of PaintableVehicleSection*	
Bicycle	*flag*	
Bicycle_Anim_Speed	*float32*	`0`
Buildable_Placement_Rule	*EVehicleBuildablePlacementRule*	`None`
Bypass_Hash_Verification	*flag*	
Cam_Driver_Offset	*float32*	`0`
Cam_Follow_Distance	*float32*	`5.5`
Cam_Passenger_Offset	*float32*	`0`
Can_Be_Locked	*bool*	`true`
Crawler	*flag*	*deprecated*
CrawlerTrackTilingMaterials	*list of CrawlerTrackTilingMaterial*	
Drops_Max	*uint8*	`7`
Drops_Min	*uint8*	`3`
Drops_Table_ID	*uint16*	`962`
Engine	*EEngine*	`Car`
Exit	*float32*	`2`
GUID	*GUID*	

continues on next page

Table 1 – continued from previous page

Property Name	Type	Default Value
Has_Clip_Prefab	*bool*	`true`
Has_Horn	*bool*	See description
HornAudioClip	*Master Bundle Pointer*	
ID	*uint16*	`0`
IgnitionAudioClip	*Master Bundle Pointer*	
LockMouse	*flag*	
Num_Steering_Tires	*int32*	*deprecated*
Rarity	*EItemRarity*	`Common`
Reclined	*flag*	
Should_Spawn_Seat_Capsules	*bool*	`false`
Steering_Tire_#	*int32*	*deprecated*
Tire_ID	*uint16*	`1451`
Trunk_Storage_X	*uint8*	`0`
Trunk_Storage_Y	*uint8*	`0`
Valid_Speed_Down	*float32*	
Valid_Speed_Horizontal	*float32*	
Valid_Speed_Up	*float32*	
Zip	*flag*	

Table 2: Handling

Property Name	Type	Default Value
Air_Steer_Max	*float32*	See description
Air_Steer_Min	*float32*	See description
Air_Turn_Responsiveness	*float32*	`2`
Brake	*float32*	
Center_Of_Mass	*vector3*	
Carjack_Force_Multiplier	*float32*	`1.0`
CrawlerTrackSteering_Torque	*float32*	`0.0`
CrawlerTrackSteering_SidewaysFrictionMultiplier	*float32*	`1.0`
CrawlerTrackSteering_MaxSpeedScale	*float32*	`1.0`
Engine_Force_Multiplier	*float32*	`1.0`
Lift	*float32*	`0`
Override_Center_Of_Mass	*bool*	`false`
Physics_Profile	*GUID*	See description
RollAngularVelocityDamping	*float32*	`-1.0`
Sleds	*flag*	
Speed_Max	*float32*	`0`
Speed_Min	*float32*	`0`
Steering_Angle_FullSpeed_Factor	*float32*	`0`
Steering_Angle_Max	*float32*	`0`
Steering_Angle_Turn_Speed	*float32*	See description
Steering_LeaningForceMultiplier	*float32*	`-1.0`
Steering_LeaningForce_ScaleWithSpeed	*bool*	`false`
Steering_LeaningForce_SpeedExponent	*float32*	`1.0`
Traction	*flag*	
Wheel_Collider_Mass_Override	*float32*	`null`
WheelBalancing_ForceMultiplier	*float32*	`-1.0`
WheelBalancing_UprightExponent	*float32*	`1.5`
WheelConfigurations	*list of VehicleWheelConfiguration*	

Table 3: Engine RPM and Gears

Property Name	Type	Default Value
EngineIdleRPM	*float32*	`1000.0`
EngineMaxRPM	*float32*	`7000.0`
EngineMaxTorque	*float32*	`1.0`
EngineRPM_DecreaseRate	*float32*	`-1.0`
EngineRPM_IncreaseRate	*float32*	`-1.0`
EngineRPMMismatch_TorqueReduction_Enabled	*bool*	`false`
EngineRPMMismatch_TorqueReduction_Threshold	*float*	`0.0`
EngineRPMMismatch_GearShift_PreventShifting	*bool*	`false`
EngineRpmMismatch_GearShift_UpMinThreshold	*float*	`0.0`
EngineRpmMismatch_GearShift_UpMaxThreshold	*float*	`0.0`
EngineRpmMismatch_GearShift_DownMinThreshold	*float*	`0.0`
EngineRpmMismatch_GearShift_DownMaxThreshold	*float*	`0.0`
ForwardGearRatios	*list of float32*	
GearShift_AllowSkippingGears	*bool*	`true`
GearShift_DownThresholdRPM	*float32*	`1500.0`
GearShift_Duration	*float32*	`0.5`
GearShift_Interval	*float32*	`1.0`
GearShift_UpThresholdRPM	*float32*	`5500.0`
GearShift_VisibleInHUD	*bool*	`true`
ReverseGearRatio	*float32*	`1.0`

Table 4: Engine Sound

Property Name	Type	Default Value
EngineSound	*RpmEngineSoundConfiguration*	
EngineSound_Type	*EVehicleEngineSoundType*	`Legacy`
Pitch_Drive	*float32*	
Pitch_Idle	*float32*	

Table 5: Health and Armor

Property Name	Type	Default Value
Bumper_AnimalDamage	*float32*	`15.0`
Bumper_Invulnerable	*flag*	
Bumper_Multiplier	*float32*	`1.0`
Bumper_ObjectDamage	*float32*	`30.0`
Bumper_PlayerDamage	*float32*	`10.0`
Bumper_ResourceDamage	*float32*	`85.0`
Bumper_SelfDamageMultiplier	*float32*	`1.0`
Bumper_SpeedDamageThreshold	*float32*	`3.0`
Bumper_ZombieDamage	*float32*	`15.0`
Can_Repair_While_Seated	*bool*	`false`
Child_Explosion_Armor_Multiplier	*float32*	`0.2`
Environment_Invulnerable	*flag*	
Explosions_Invulnerable	*flag*	
Health	*uint16*	`0`
Health_Max	*uint16*	`0`
Health_Min	*uint16*	`0`
Invulnerable	*flag*	
Passenger_Explosion_Armor	*float32*	`1`
Tires_Invulnerable	*flag*	

Table 6: Fuel

Property Name	Type	Default Value
Fuel	*uint16*	`0`
Fuel_Burn_Rate	*float32*	See description
Fuel_Min	*uint16*	`0`
Fuel_Max	*uint16*	`0`

Table 7: Battery

Property Name	Type	Default Value
Battery_Burn_Rate	*float32*	`20`
Battery_Charge_Rate	*float32*	`20`
Battery_Powered	*flag*	
Battery_Spawn_Charge_Multiplier	*float32*	`1`
BatteryMode_Driving	*EBatteryMode*	`Charge`
BatteryMode_Empty	*EBatteryMode*	`None`
BatteryMode_Headlights	*EBatteryMode*	`Burn`
BatteryMode_Sirens	*EBatteryMode*	`Burn`
Can_Steal_Battery	*bool*	`true`
Cannot_Spawn_With_Battery	*flag*	
Default_Battery	*GUID*	`098b13be34a7411db7736b7f866ada6`

Table 8: Stamina

Property Name	Type	Default Value
Stamina_Boost	*float32*	
Stamina_Powered	*flag*	

Table 9: Paintability

Property Name	Type	Default Value
DefaultPaintColor_Configuration	*VehicleRandomPaintColorConfiguration*	
DefaultPaintColor_Mode	*EVehicleDefaultPaintColorMode*	See description
DefaultPaintColors	*list of colors*	
IsPaintable	*bool*	
PaintableSections	*list of PaintableVehicleSection*	

Table 10: Explosion

Property Name	Type	Default Value
Explosion	*GUID* or *uint16*	
ExplosionBurnMaterialSections	*list of PaintableVehicleSection*	
Explosion_Force_Multiplier	*float32*	`1.0`
Explosion_Max_Force	*vector3*	`(0, 1024, 0)`
Explosion_Min_Force	*vector3*	`(0, 1024, 0)`
ShouldExplosionBurnMaterials	*bool*	See description
ShouldExplosionCauseDamage	*bool*	See description

Table 11: Turret

Property Name	Type	Default Value
Turret_#_Ignore_Aim_Camera	*flag*	
Turret_#_Item_ID	*uint16*	`0`
Turret_#_Pitch_Max	*float32*	`0`
Turret_#_Pitch_Min	*float32*	`0`
Turret_#_Seat_Index	*uint8*	`0`
Turret_#_Yaw_Max	*float32*	`0`
Turret_#_Yaw_Min	*float32*	`0`
Turrets	*uint8*	`0`

Table 12: Train

Property Name	Type	Default Value
Train_Car_Length	*float32*	`0`
Train_Track_Offset	*float32*	`0`
Train_Wheel_Offset	*float32*	`0`

Table 13: Economy

Property Name	Type	Default Value
Shared_Skin_Lookup_ID	*GUID* or *uint16*	See description
Shared_Skin_Name	*string*	
Size2_Z	*float32*	`0`

76.1.2 EEngine Enumeration

Named Value	Description
`Car`	This vehicle is part of the Car category.
`Plane`	This vehicle is part of the Plane category.
`Blimp`	This vehicle is part of the Blimp category.
`Boat`	This vehicle is part of the Boat category.
`Train`	This vehicle is part of the Train category.

76.1.3 EVehicleBuildablePlacementRule Enumeration

Named Value	Description
`None`	Vehicle does not override placement. This means that barricades can be attached *unless* the barricade sets *Allow_Placement_On_Vehicle* to `false` (e.g., beds and sentry guns are often set to `false`).
`AlwaysAllow`	Vehicle allows any barricade to be placed on it, regardless of the barricade's *Allow_Placement_On_Vehicle* setting. Trains used to use this option, but it can be exploited to move beds into out-of-bounds areas (e.g., into other objects).
`Block`	Vehicle prevents any barricade form being placed on it.

76.1.4 EVehicleDefaultPaintColorMode

Named Value	Description
`None`	Not configured.
`List`	Pick from the *DefaultPaintColors* list.
`RandomHueOrGrayscale`	Pick a random HSV using *DefaultPaintColor_Configuration*.

76.1.5 EVehicleEngineSoundType Enumeration

Named Value	Description
`Legacy`	Default.
`EngineRPMSimple`	Set pitch and volume of a single clip according to engine RPM.

76.1.6 EWheelMotionEffectsMode Enumeration

Named Value	Description
`None`	Turn off motion effects. Default for wheels not using collider pose.
`BothDirections`	Enable motion effects. Default for wheels using collider pose.
`ForwardOnly`	Enable motion effects, but turn them off while moving backward.
`BackwardOnly`	Enable motion effects, but turn them off while moving forward.

76.1.7 EWheelSteeringMode Enumeration

Named Value	Description
`None`	Wheel does not affect steering.
`SteeringAngle`	Set WheelCollider steering angle according to Steer_Min and Steer_Max.
`CrawlerTrack`	Increase or decrease motor torque to rotate vehicle in-place.

76.1.8 ECrawlerTrackForwardMode Enumeration

Named Value	Description
`Auto`	Assigns a forward mode based on wheel collider position. Wheels on the left side are `Clockwise` and wheels on the right side are `CounterClockwise`.
`Clockwise`	Positive motor torque on this wheel rotates the vehicle clockwise.
`CounterClockwise`	Positive motor torque on this wheel rotates the vehicle counter-clockwise.

76.1.9 CrawlerTrackTilingMaterial Dictionary

Property Name	Type	Default Value
Path	*string*	
MaterialIndex	*int32*	`0`
WheelIndices	*list of int32*	
RepeatDistance	*float32*	`0.0`
UV_Direction	*Vector2*	`(0.0, 0.0)`

76.1.10 PaintableVehicleSection Dictionary

Property Name	Type	Default Value
Path	*string*	
MaterialIndex	*int32*	`0`
AllMaterials	*bool*	`false`

76.1.11 RpmEngineSoundConfiguration Dictionary

Property Name	Type	Default Value
IdlePitch	*float32*	`0.0`
IdleVolume	*float32*	`0.0`
MaxPitch	*float32*	`0.0`
MaxVolume	*float32*	`0.0`

76.1.12 VehicleRandomPaintColorConfiguration Dictionary

Property Name	Type	Default Value
MinSaturation	*float32*	`0.0`
MaxSaturation	*float32*	`0.0`
MinValue	*float32*	`0.0`
MaxValue	*float32*	`0.0`
GrayscaleChance	*float32*	`0.0`

76.1.13 VehicleWheelConfiguration Dictionary

Property Name	Type	Default Value
CanExplode	*bool*	`true`
CopyColliderRpmIndex	*int32*	`-1`
CrawlerTrackForwardMode	*ECrawlerTrackForwardMode Enumeration*	`Auto`
IsColliderPowered	*bool*	`false`
IsModelSteered	*bool*	`false`
ModelPath	*string*	
ModelRadius	*float32*	`-1.0`
ModelSuspensionOffset	*float32*	`0.0`
ModelSuspensionSpeed	*float32*	`-1.0`
ModelUseColliderPose	*bool*	`false`
MotionEffects	*EWheelMotionEffectsMode Enumeration*	See description
SteeringAngleMultiplier	*float32*	`1.0`
SteeringMode	*EWheelSteeringMode Enumeration*	`None`
WheelColliderPath	*string*	

76.1.14 Property Descriptions

AdditionalTransparentSections list of PaintableVehicleSection

Materials to register as needing transparent sorting. Their render queue is periodically updated according to whether their pivot point is underwater.

Air_Steer_Max float32

The angle to turn when moving quickly, when using `Engine Plane`. Defaults to the value of `Steer_Max`.

Air_Steer_Min float32

The angle to turn when moving slowly, when using `Engine Plane`. Defaults to the value of `Steer_Min`.

Air_Turn_Responsiveness float32 `2`

Sensitivity on steering while airborne, when using `Engine Plane`.

Battery_Burn_Rate float32 `20`

This controls the rate at which battery charge decreases per second.

Battery_Charge_Rate float32 `20`

This controls the rate at which battery charge increases per second.

Battery_Powered flag

The vehicle does not use fuel. For example, this flag is useful for creating electric vehicles.

Battery_Spawn_Charge_Multiplier float32 `1`

Battery charge on a newly-spawned vehicle is multiplied by this [0, 1] number. Setting this to a number less than 1 will result in the vehicle spawning with less battery charge than normal.

BatteryMode_Driving EBatteryMode `Charge`

How the vehicle battery should behave when a player is driving it.

BatteryMode_Empty EBatteryMode `None`

How the vehicle battery should behave when the vehicle is empty.

BatteryMode_Headlights EBatteryMode `Burn`

How the vehicle battery should behave when the headlights are on.

BatteryMode_Sirens EBatteryMode `Burn`

How the vehicle battery should behave when the siren is on.

Bicycle flag

Player character should use a bicycling animation.

Bicycle_Anim_Speed float32 `0`

Multiplier on the speed of the bicycling animation.

Brake float32

The amount of braking force to apply.

Buildable_Placement_Rule EVehicleBuildablePlacementRule `None`

This property overrides how barricades can be attached to the vehicle. View the *EVehicleBuildablePlacementRule* documentation for more information about the behavior of each option. Note that the `Bypass_Buildable_Mobility` gameplay config option, when `true`, will always take priority over this property.

The `Supports_Mobile_Buildables` flag predates this property, and has since been deprecated. Its behavior can be replicated by using this property with the `AlwaysAllow` value instead.

Bumper_AnimalDamage float32 `15`

Base damage to animals when traveling at 1 m/s (before speed and other multipliers apply).

Bumper_Invulnerable flag

The vehicle cannot be damaged by collisions (such as with other vehicles, objects, placeables, or entities).

Bumper_Multiplier float32 `1.0`

Multiplies vehicle speed (in m/s) at time of collision. If result is less than `Bumper_SpeedDamageThreshold`, no damage is applied.

The multiplied speed is applied to outgoing damage. For example: if speed is 4 m/s with a `Bumper_Multiplier` of 2 and a base damage of 10, 80 damage is applied.

Bumper_ObjectDamage float32 `30`

Base damage to objects when traveling at 1 m/s (before speed and other multipliers apply).

Bumper_PlayerDamage float32 10

Base damage to players when traveling at 1 m/s (before speed and other multipliers apply).

Bumper_ResourceDamage float32 85

Base damage to resources when traveling at 1 m/s (before speed and other multipliers apply).

Bumper_SelfDamageMultiplier float32 1.0

Multiplier for damage inflicted to the vehicle by crashing into things. Not applicable if the `Bumper_Invulnerable` flag is applied.

Bumper_SpeedDamageThreshold float32 3.0

If speed (in m/s) multiplied by `Bumper_Multiplier` is less than this threshold, no damage is applied.

Bumper_ZombieDamage float32 15

Base damage to zombies when traveling at 1 m/s (before speed and other multipliers apply).

Bypass_Hash_Verification flag

Disable hash verification check, and allow for mismatched files.

Cam_Driver_Offset float32 0

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 float32 5.5

The distance behind the player the third-person camera should be placed at, in meters.

Cam_Passenger_Offset float32 0

The vertical offset for any passenger's (including the driver's) first-person camera, in meters.

Can_Be_Locked bool `true`

Whether or not the vehicle can be locked by a player.

Can_Repair_While_Seated bool `false`

When `true`, this vehicle can be repaired by seated players.

Can_Steal_Battery bool `true`

Whether or not the vehicle battery can be removed from the vehicle by a player.

Cannot_Spawn_With_Battery flag

The vehicle does not spawn with a vehicle battery.

Carjack_Force_Multiplier float32 `1.0`

This is a multiplier on the force applied when using a Carjack on this vehicle. It is recommended that this property scales based on your vehicle's mass.

Although this property was originally intended for modded vehicles, many official vehicles use this property as well. If you are creating a custom vehicle and using one of the example assets provided as a template (or have a mass that is similar to official content), you will likely want to use a value of 2 for this property.

The mass of official vehicles may be revisited in the future, to make collisions feel more reasonable. If this happens, the recommended value could be increased again.

CrawlerTrackSteering_Torque float32 `1.0`

Added or subtracted from wheel motor torque in `CrawlerTrack` mode.

For example, if the vehicle is attempting to turn left and the wheel's `CrawlerTrackForwardMode` is `Clockwise` this value is subtracted from the motor torque.

CrawlerTrackSteering_SidewaysFrictionMultiplier float32 `1.0`

For a wheel in `CrawlerTrack` mode, the wheel collider's sideways friction stiffness is multiplied by this value while a steering input is applied. Potentially useful when turning in place to overcome the sideways friction.

CrawlerTrackSteering_MaxSpeedScale float32 `1.0`

Multiplier for crawler track steering torque and sideways friction multiplier while driving at maximum speed, similar to how steering angle is reduced at maximum speed (`Steer_Max`). Useful to keep the steering controllable while driving.

CrawlerTrackTilingMaterials list of CrawlerTrackTilingMaterial

If set, offsets a crawler track's material UVs in sync with wheels rolling.

Center_Of_Mass vector3

Overrides the vehicle's center of mass on the -, -, and -axis, when using `Override_Center_Of_Mass true`. This allows for modifying a vehicle's center of gravity without needing to move the "Cog" GameObject in Unity.

For example:

```
Override_Center_Of_Mass true
Center_Of_Mass (0, -50, 0)
```

Child_Explosion_Armor_Multiplier float32 `0.2`

This is a multiplier on the damage that barricades (and other buildables) placed on the vehicle receive, when damaged by explosions.

Crawler flag

Deprecated since version 3.23.4.0: Replaced by the `WheelConfigurations` property.

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.

Default_Battery GUID `098b13be34a7411db7736b7f866ada69`

Battery item given to the player when a specific battery hasn't been manually installed yet. Defaults to the Vehicle Battery used by official vehicles.

DefaultPaintColor_Configuration VehicleRandomPaintColorConfiguration

Determines the potential colors of a newly-spawned vehicle. Can be overridden by a *Vehicle Redirector*'s *LoadPaintColor* and *LoadPaintColor* properties.

This property is used with `DefaultPaintColor_Mode RandomHueOrGrayscale`. For example:

```
DefaultPaintColor_Mode RandomHueOrGrayscale
DefaultPaintColor_Configuration
{
        MinSaturation 0.15
        MaxSaturation 0.7
        MinValue 0.15
        MaxValue 0.9
        GrayscaleChance 0.1
}
```

DefaultPaintColor_Mode EVehicleDefaultPaintColorMode

This property controls the mode that should be used when randomly picking a paint color for a newly-spawned vehicle. Defaults to `List` if *DefaultPaintColors* has been configured. Otherwise, defaults to `None`. This can be manually set to `RandomHueOrGrayscale` to pick a random HSV.

DefaultPaintColors list of colors

List of random colors to pick from when spawning a new vehicle. Can be overridden by a *Vehicle Redirector*'s *LoadPaintColor* and *LoadPaintColor* properties.

This property is used with `DefaultPaintColor_Mode List`. For example:

```
DefaultPaintColor_Mode List
DefaultPaintColors
[
        "#353535" // Classic Black
        "#37658c" // Classic Blue
        "#2e642e" // Classic Green
        "#bd6e27" // Classic Orange
        "#6a466d" // Classic Purple
        "#9a2525" // Classic Red
        "#d4d4d4" // Classic White
        "#cdaa1e" // Classic Yellow
]
```

Drops_Max uint8 7

Maximum amount of item drops to spawn when the vehicle is destroyed.

Drops_Min uint8 3

Minimum amount of item drops to spawn when the vehicle is destroyed.

Drops_Table_ID uint16 962

ID of the item spawn table to use when the vehicle is destroyed. The default spawn table is Destroyed_Vehicle_Default.

Engine EEngine Car

The `Engine` property determines the type of vehicle (e.g., car, plane, boat). Some vehicle properties are only usable depending on the vehicle's `Engine`.

Engine_Force_Multiplier float32 `1.0`

This is a multiplier on otherwise not-yet-configurable plane/heli/boat/etc. forces. It is recommended that this property scales based on your vehicle's mass.

EngineMaxTorque float32 `1.0`

Multiplier for the amount of torque provided to the wheels. Understanding how engine RPM is translated to wheel torque is crucial for tuning the physics:

1. Engine RPM is normalized into a 0 to 1 range according to *EngineIdleRPM* and *EngineMaxRPM*. For example, an Engine RPM of 2000 with Idle RPM of 1000 and Max RPM of 5000 would be 0.25.
2. Vehicle root needs an `EngineCurvesComponent` attached. This allows you to map normalized engine RPM to a normalized torque multiplier. Typically, the multiplier should be closest to 1 in the middle range (e.g., 0.3 to 0.8) and drop off toward 0 and 1.
3. Torque curve is sampled using the normalized engine RPM.
4. Sampled torque is multiplied by `EngineMaxTorque`.
5. If changing gears, torque is zero.
6. If reversing, torque is multiplied by *ReverseGearRatio*.
7. Otherwise, torque is multiplied by the active *ForwardGearRatio*.
8. Each *Powered Wheel* gets an equal share of the torque. To clarify, the per-wheel torque is equal to the engine output torque divided by the number of powered wheels.

EngineRPM_DecreaseRate float32 `-1.0`

How quickly engine RPM can decrease in RPM/s. For example, 1000 will take 2 seconds to go from 4000 to 2000 RPM.

Defaults to -1 which instantly changes RPM.

> **Note**
>
> Originally, I thought this might come in handy, but in practice tuning the torque and gear ratios worked better. Kept in case it comes in useful for somebody.

EngineSound RpmEngineSoundConfiguration

When *EngineSound_Type* is set to `EngineRPMSimple`, this should be set to an *RpmEngineSoundConfiguration dictionary*.

EngineSound_Type EVehicleEngineSoundType `Legacy`

Defaults to `Legacy`. In that mode, `Pitch_Idle` and `Pitch_Drive` are used to control engine audio pitch.

EngineIdleRPM float32 `1000.0`

Engine RPM will never drop below this value regardless of whether wheel RPM * gear ratio is lower. Otherwise, the engine wouldn't be able to start the wheels rolling from zero.

EngineRPM_IncreaseRate float32 `-1.0`

How quickly engine RPM can increase in RPM/s. For example, 1000 will take 2 seconds to go from 2000 to 4000 RPM.

Defaults to -1 which instantly changes RPM.

> **Note**
>
> Originally, I thought this might come in handy, but in practice tuning the torque and gear ratios worked better. Kept in case it comes in useful for somebody.

EngineRPMMismatch_TorqueReduction_Enabled float32 `false`

If true, wheel RPM is reduced according to the difference between expected and actual wheel RPM divided by torque reduction threshold.

EngineRPMMismatch_TorqueReduction_Threshold float32 `0.0`

If torque reduction is enabled, torque is reduced to zero when difference between expected and actual RPM is greater than this threshold.

For non-linear reduction, you can enable **Use Engine RPM Mismatch Torque Reduction Curve** on the Engine Curves Component.

EngineRPMMismatch_GearShift_PreventShifting float32 `-1.0`

If true, prevent changing gears when the difference between expected and actual wheel RPM exceeds threshold.

EngineRpmMismatch_GearShift_UpMinThreshold float32 `0.0`

If prevent shifting is enabled, prevent changing gears up when the difference between expected and actual wheel RPM is less than this threshold.

I.e., if `expected - actual < min` it cannot shift up.

EngineRpmMismatch_GearShift_UpMaxThreshold float32 `0.0`

If prevent shifting is enabled, prevent changing gears up when the difference between expected and actual wheel RPM is greater than this threshold.

I.e., if `expected - actual > max` it cannot shift up.

EngineRpmMismatch_GearShift_DownMinThreshold float32 `0.0`

If prevent shifting is enabled, prevent changing gears down when the difference between expected and actual wheel RPM is less than this threshold.

I.e., if `expected - actual < min` it cannot shift down.

EngineRpmMismatch_GearShift_DownMaxThreshold float32 `0.0`

If prevent shifting is enabled, prevent changing gears down when the difference between expected and actual wheel RPM is greater than this threshold.

I.e., if `expected - actual > max` it cannot shift down.

EngineMaxRPM float32 `7000.0`

Engine RPM will never exceed this value regardless of whether wheel RPM * gear ratio is higher. It should be kept to a reasonable value because the normalized engine RPM is used in a variety of places like sampling the torque curve and network replication.

Environment_Invulnerable flag

This vehicle cannot be damaged by animals, zombie melee attacks, or boulders thrown by mega zombies. Zombies and animals will still pursue the vehicle, and attempt to attack any passengers directly. Other damage sources can still damage the vehicle.

> **Tip**
>
> The original reason zombies attack passengers directly was for stationary turrets in boss encounters. The (vehicle) turret can be used to attack the boss without risk of getting destroyed and despawned, but the player is unsafe inside.

Exit float `2`

Distance away from the vehicle to teleport when exiting.

Explosion GUID or uint16

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

Explosion_Force_Multiplier float32 `1.0`

This is a multiplier on the force applied when the vehicle explodes. It is recommended that this property scales based on your vehicle's mass.

Many official vehicles use this property. If you are creating a custom vehicle and using one of the example assets provided as a template (or have a mass that is similar to official content), you will likely want to use a value of `2` for this property.

Explosion_Max_Force vector3 `(0, 1024, 0)`

Maximum amount of force applied on the -, -, and -axis, when the vehicle explodes.

Explosion_Min_Force vector3 `(0, 1024, 0)`

Minimum amount of force applied on the -, -, and -axis, when the vehicle explodes.

Explosions_Invulnerable flag

The vehicle cannot be damaged by explosions.

ForwardGearRatios list of float32

Ratio between engine RPM and wheel RPM in a given gear. For example, if the wheel RPM is 6 and the gear ratio is 5 then the engine RPM is 30.

> **Note**
>
> When converting vanilla cars to gear ratios, the approach I used was to calculate the gear ratio for a desired speed and engine RPM. Suppose you're targeting 80 kph with a wheel radius of 0.6 m:
>
> 1. Convert 80 kph to m/s, in this case, 22.2 m/s.
> 2. Calculate wheel circumference with 2 * pi * r, in this case 3.77 m.
> 3. Calculate how far the vehicle would travel in a minute. 22.2 m/s * 60 s/min is 1,333.2 m/min.
> 4. Divide the distance per minute by the circumference to get the wheel RPM of 353.6776.
>
> Supposedly (I'm still learning as I go) engines work most efficiently around the upper-middle of their RPM range. For example, 3500 RPM for an engine with 1000 idle RPM and 6000 max RPM. Using 3500 as our target engine RPM we can divide it by the wheel RPM to get a good starting point for the gear ratio tuning: 9.89.

Fuel uint16 `0`

Total fuel capacity.

Fuel_Burn_Rate float32

This controls the rate at which fuel decreases per second. Defaults to `2.05` when using `Engine Car`, or to `4.2` otherwise.

Fuel_Max uint16 `0`

Maximum possible fuel to spawn with.

Fuel_Min uint16 `0`

Minimum possible fuel to spawn with.

GearShift_AllowSkippingGears bool `true`

If true, engine can skip from (for example) 1st to 3rd gear if it keeps RPM within the acceptable range.

GearShift_DownThresholdRPM float32 `1500.0`

When engine RPM is below this value and a lower gear is available the car will shift gears down.

GearShift_Duration float32 `0.5`

How long it takes to shift gears, measured in seconds. Wheels do not provide any torque for this duration.

GearShift_Interval float32 `1.0`

How long to wait since the last gear change before shifting gears, measured in seconds. It can take a moment for the engine RPM to adjust after a gear change, so without a delay the RPM would still exceed the threshold.

GearShift_UpThresholdRPM float32 `5500.0`

When engine RPM is above this value and a higher gear is available the car will shift gears up.

GearShift_VisibleInHUD bool `true`

If gears are configured and this is true, RPM and gear number will be shown in the user interface.

GUID GUID

Refer to *GUID* documentation. Vehicles 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.

Has_Clip_Prefab bool `true`

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`.

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`.

Health uint16 `0`

Total health value.

Health_Max uint16 `0`

Minimum possible health to spawn with.

Health_Min uint16 `0`

Maximum possible health to spawn with.

HornAudioClip Master Bundle Pointer

AudioClip to play when using the horn.

ID uint16 `0`

Must be a unique identifier. This property used to be required by vehicles, but this is no longer necessary.

The range reserved for official content is [1, 2000).

IgnitionAudioClip Master Bundle Pointer

AudioClip to play after entering the driver's seat.

Invulnerable flag

The vehicle cannot be damaged by lower-power *Weapon Assets* that do not have the `Invulnerable` flag.

IsPaintable bool

If `true`, *Vehicle Paint Tools* can be used on this vehicle. Defaults to `true` if *PaintableSections* has been configured.

Lift float32 `0`

The amount of upwards lift force to apply, when using `Engine Plane`.

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

Deprecated since version 3.23.4.0: Replaced by the `WheelConfigurations` property.

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.

Override_Center_Of_Mass bool `false`

If `true`, overrides the vehicle's center of mass with the values from the `Center_Of_Mass` property. This allows for modifying a vehicle's center of gravity without needing to move the "Cog" GameObject in Unity.

PaintableSections list of PaintableVehicleSection

If set, the vehicle can be painted with a *Vehicle Paint Tool*. Each section's material's `_PaintColor` property is set to the vehicle's paint color.

Passenger_Explosion_Armor float32 `1`

Multiplier on the damage taken by players sitting in the vehicle, by explosions.

Physics_Profile GUID

GUID of a *VehiclePhysicsProfileAsset* to use. Physics profiles allow for increased control over vehicle settings in bulk, but are not required for anything.

There are several default profiles. These are used when the vehicle's *Engine* property has been set to `Boat`, `Car`, `Helicopter`, or `Plane`, when certain conditions are met. The Vehicle.prefab's root RigidBody component must have a mass equal to 1.0, and any of its Tires' WheelColliders must also have a mass equal to 1.0. Otherwise, nothing is used by default.

- `Boat` defaults to `47258d0dcad14cb8be26e24c1ef3449e`.

- `Car` defaults to `6b91a94f01b6472eaca31d9420ec2367`.
- `Helicopter` defaults to `bb9f9f0204c4462ca7d976b87d1336d4`.
- `Plane` defaults to `93a47d6d40454335b4784e803628ac54`.

Other vehicle types do not have a default profile.

Pitch_Drive float32

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". Otherwise, defaults to `0.05`.

Pitch_Idle float32

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". Otherwise, defaults to `0.5`.

Rarity EItemRarity `Common`

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

Reclined flag

Player character should use a reclined idle animation.

ReverseGearRatio float32 `1.0`

Gear ratio to use when reversing. Please refer to *ForwardGearRatios* for more details on gear ratios.

RollAngularVelocityDamping float32 `-1.0`

If greater than zero, an acceleration is applied to angular velocity on -axis toward zero.

Shared_Skin_Lookup_ID GUID or uint16

GUID or legacy ID of another vehicle, which this vehicle should share skins with. This property was used by some official vehicles (such as the Rally Car), as each paint color used to be a separate vehicle. This is no longer necessary, but some modded vehicles may still rely on this functionality. Defaults to the value of this vehicle's configured `GUID`.

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`.

Should_Spawn_Seat_Capsules bool `false`

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, such as bicycles.

ShouldExplosionBurnMaterials bool

When `true`, the materials of the vehicle's `Model_#` GameObjects will be tinted black when the vehicle is destroyed. Defaults to `true` if the `Explosion` property is configured.

ExplosionBurnMaterialSections list of PaintableVehicleSection

Only used if `ShouldExplosionBurnMaterials` is `true`.

If set, manually specifies which materials should be darkened after an explosion.

ShouldExplosionCauseDamage bool

When `true`, the explosion caused by the vehicle being destroyed will deal damage to nearby entities, and kill any passengers. Defaults to `true` if the `Explosion` property is configured.

Size2_Z float32 `0`

Orthogonal camera size for economy icons.

Sleds flag

Tires should easily roll. For example, most planes will want to use this property.

Speed_Max float32 `0`

The vehicle's maximum velocity to aim for while accelerating forward, 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).

Speed_Min float32 `0`

The vehicle's maximum velocity to aim for while accelerating in reverse, 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).

Stamina_Boost float32

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.

Steering_Angle_FullSpeed_Factor float32 `0`

Multiplier for steering angle range at target maximum speed (for the current forward/backward direction). Reducing steering range at higher speeds keeps the vehicle controlable with digital (non-analog) input.

Steering_Angle_Max float32 `0`

Steering angle range at zero speed (idle/parked). For example, 45 means the wheels connected to steering can rotate ±45 degrees.

Steer_Max float32 `0`

> **Tip**
>
> This property was replaced by *Steering_Angle_Max* which **isn't** multiplied by 0.75.

Steering angle range at zero speed (idle/parked). This value is multiplied by 0.75.

Steer_Min float32 `0`

> **Tip**
>
> This property was replaced by *Steering_Angle_FullSpeed_Factor*.

Steering angle range at target maximum speed (for the current forward/backward direction).

Steering_Angle_Turn_Speed float32

How quickly wheels can turn to meet player input, measured in degrees per second. Defaults to the value of `Steer_Max * 5.0`.

Steering_LeaningForceMultiplier float32 `-1.0`

If greater than zero, torque is applied on -axis according to steering input for bikes and motorcycles.

Steering_LeaningForce_ScaleWithSpeed bool `false`

If true, leaning force is multiplied by normalized speed to the power of Steering_LeaningForce_SpeedExponent. Defaults to false.

Steering_LeaningForce_SpeedExponent float32 `1.0`

Refer to Steering_LeaningForce_ScaleWithSpeed.

Steering_Tire_# int32

Deprecated since version 3.23.4.0: Replaced by the `WheelConfigurations` property.

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.

Tire_ID uint16 `1451`

ID of the item that should be 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`.

Tires_Invulnerable flag

Tires cannot be damaged.

Traction flag

Tires should have traction in snowy positions.

Train_Car_Length float32 `0`

The distance between each train car on the train, in meters. This property is used with `Engine Train`.

Train_Track_Offset float32 `0`

The offset the train car is above the track, in meters. This property is used with `Engine Train`.

Train_Wheel_Offset float32 `0`

The offset between the wheels, in meters. This property is used with `Engine Train`.

Trunk_Storage_X uint8 `0`

Number of columns (horizontal storage space).

Trunk_Storage_Y uint8 `0`

Number of rows (vertical storage space).

Turret_#_Ignore_Aim_Camera flag

Normally, the player's camera is positioned at the "Aim" GameObject. Including this flag will disable this feature.

Turret_#_Item_ID uint16 `0`

Legacy ID of the item usable from the turret seat. This is often used with a *GunAsset* that has the `Turret` property. However, other items can be used – although most will function in unintended ways.

Turret_#_Pitch_Max float32 `0`

Maximum allowed rotation of the turret through the elevation, in degrees.

Turret_#_Pitch_Min float32 `0`

Minimum allowed rotation of the turret through the elevation, in degrees.

Turret_#_Seat_Index uint8 `0`

Index of the "Seat_#" GameObject that this turret is usable from. For example, `0` would correspond to "Seat_0".

Turret_#_Yaw_Max float32 `0`

Maximum allowed rotation of the turret through the azimuth, in degrees. If this is set to `360`, it can rotate rightward forever.

Turret_#_Yaw_Min float32 `0`

Minimum allowed rotation of the turret through the azimuth, in degrees. If this is set to `-360`, it can rotate leftward forever.

Turrets uint8 `0`

Total number of turrets on the vehicle. All other turret-related properties require that this property has been configured.

For example, this is how the Fighter Jet is configured:

```
Turrets 1
Turret_0_Seat_Index 0
Turret_0_Item_ID 1471
Turret_0_Yaw_Min -135
Turret_0_Yaw_Max 135
Turret_0_Pitch_Min 85
Turret_0_Pitch_Max 180
Turret_0_Ignore_Aim_Camera
```

Type EAssetType

Designates the vehicle's class. Vehicle assets are required to have this property.

Valid_Speed_Down float32

Configuring this will override the downward vertical speed sanity check on the Y-axis, measured in m/s (meters per second). Multiplayer checks if speed exceeds this value, and the movement is marked as invalid if so.

Defaults to `25` when using `Engine Car` or `Engine Boat`, or to `100` otherwise.

Valid_Speed_Horizontal float32

Configuring this will override the horizontal speed sanity check on the XZ plane, measured in m/s (meters per second). Multiplayer checks if speed exceeds this value, and the movement is marked as invalid if so.

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 float32

Configuring this will override the upward vertical speed sanity check on the Y-axis, measured in m/s (meters per second). Multiplayer checks if speed exceeds this value, and the movement is marked as invalid if so.

Defaults to 12.5 when using `Engine Car`, to 3.25 when using `Engine Boat`, or to 100 otherwise.

Wheel_Collider_Mass_Override float32 `null`

Override the mass of the vehicle's WheelCollider 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`).

WheelBalancing_ForceMultiplier float32 `-1.0`

If greater than zero, torque is applied on the Z axis multiplied by this factor to align vehicle up with ground up.

> **Note**
>
> *RollAngularVelocityDamping* is critical for damping this force.

WheelBalancing_UprightExponent float32 `1.5`

Exponent on the 0 to 1 factor representing how aligned the vehicle is with the ground up vector. For example, a value of 2 would apply much less force while nearly aligned with up, whereas a value of 0.5 would apply more force even while nearly aligned with up.

WheelConfigurations list of VehicleWheelConfiguration

Controls WheelCollider components and their corresponding visual models. When converting older vehicles, enable the `-LogVehicleWheelConfigurations` command-line flag to output an equivalent wheel configuration.

For example, this is how the Ambulance is configured:

```
WheelConfigurations
[
        {
                WheelColliderPath Tires/Tire_0
                IsColliderSteered true
                IsColliderPowered true
                ModelPath Wheels/Wheel_0
                ModelUseColliderPose true
        }
        {
                WheelColliderPath Tires/Tire_1
                IsColliderSteered true
                IsColliderPowered true
                ModelPath Wheels/Wheel_1
                ModelUseColliderPose true
        }
        {
                WheelColliderPath Tires/Tire_2
                IsColliderSteered false
                IsColliderPowered false
                ModelPath Wheels/Wheel_2
```

(continues on next page)

(continued from previous page)

```
                ModelUseColliderPose true
        }
        {
                WheelColliderPath Tires/Tire_3
                IsColliderSteered false
                IsColliderPowered false
                ModelPath Wheels/Wheel_3
                ModelUseColliderPose true
        }
]
```

Zip flag

Player character should use a handlebar idle animation.

76.1.15 CrawlerTrackTilingMaterial Dictionary Descriptions

Path string

Scene hierarchy path to a Renderer component relative to the vehicle's root transform.

MaterialIndex int32

Index into Renderer component's Materials list. For example, `0` is the 1st material, `1` is the 2nd material, and so forth.

WheelIndices list of int32

Index into wheel configurations list. For example, `0` is the 1st wheel, `1` is the 2nd wheel, and so forth.

The average of these wheels' RPM is used to calculate how fast crawler track is moving.

RepeatDistance float32

How far to travel to offset UV 1x.

You can calculate RepeatDistance by selecting an edge parallel to the crawler track and dividing the UV distance by the physical 3D distance. For example, if the UV length is 2 and the 3D length is 1.5 m then the texture repeats 1.33 UV/m.

UV_Direction Vector2

For example, `(-1.0, 0.0)` will move the UV offset left as the vehicle travels forward.

76.1.16 PaintableVehicleSection Dictionary Descriptions

Path string

Scene hierarchy path to a Renderer component relative to the vehicle's root transform.

MaterialIndex int32 `0`

Index into Renderer component's Materials list. For example, `0` is the 1st material, `1` is the 2nd material, and so forth.

AllMaterials bool `false`

If true, apply to all materials in Renderer component's Materials list, rather than a specific index.

76.1.17 RpmEngineSoundConfiguration Dictionary Descriptions

IdlePitch float32 `0.0`

AudioSource pitch when engine RPM is at *Idle RPM*.

IdleVolume float32 `0.0`

AudioSource volume when engine RPM is at *Idle RPM*.

MaxPitch float32 `0.0`

AudioSource pitch when engine RPM is at *Max RPM*.

MaxVolume float32 `0.0`

AudioSource volume when engine RPM is at *Max RPM*.

76.1.18 VehicleRandomPaintColorConfiguration Dictionary Descriptions

MinSaturation float32 `0.0`

Minimum random saturation in HSV color to generate.

MaxSaturation float32 `0.0`

Maximum random saturation in HSV color to generate.

MinValue float32 `0.0`

Minimum value or brightness in HSV color to generate.

MaxValue float32 `0.0`

Maximum value or brightness in HSV color to generate.

GrayscaleChance float32 `0.0`

[0, 1] color will have zero saturation if random value is less than this. For example, 0.2 means 20% of vehicles will be grayscale.

76.1.19 VehicleWheelConfiguration Dictionary Descriptions

CanExplode bool `true`

If true, wheel should fly off when vehicle explodes. Defaults to true. Used to simplify destroying vehicles with crawler tracks.

CopyColliderRpmIndex int32 `-1`

If set, visual-only wheels without a collider (like the back wheels of the snowmobile) can copy RPM from a wheel that does have a collider. Requires *ModelRadius* to also be set.

CrawlerTrackForwardMode ECrawlerTrackForwardMode `Auto`

For `CrawlerTrack` `SteeringMode`, indicates how a positive motor torque (forward) rotates the vehicle.

IsColliderPowered bool `false`

If `true`, collider is connected to the engine and responds to player's acceleration input.

IsColliderSteered bool `false`

If `true`, collider's steering angle responds to player input.

Deprecated since version 3.23.7.0: Replaced by the `SteeringMode` property.

IsModelSteered bool `false`

If `true`, model is rotated according to steering input.

Only kept for backwards compatibility. Prior to wheel configurations, only certain WheelColliders actually received steering input, while multiple models would appear to steer. For example, the APC's front 4 wheels appeared to rotate but only the front 2 actually affected physics.

ModelPath string

Scene hierarchy path of visual representation of wheel relative to the vehicle's root transform.

ModelRadius float32 `-1.0`

If greater than zero, visual-only wheels (without a collider) like the extra wheels of the Snowmobile use this radius to calculate their rolling speed.

ModelUseColliderPose bool `false`

If `true`, model ignores `IsModelSteered` and instead uses the wheel collider state (or an approximation when not simulating).

Prior to wheel configurations, some high-fidely modded cars used a separate set of physics constraints to animate the wheel models as if they had suspension. Constraint setups like this should be completely superseded by the `ModelUseColliderPose` property. The constraints were awful for performance because physics for every purely-visual wheel were simulated on every client, and even then they didn't actually match the real wheel state.

ModelSuspensionOffset float `0.0`

Vertical offset of model from simulated suspension position. Used by crawler tracks because the visual wheel rests on the tread above the physics wheel.

ModelSuspensionSpeed float `-1.0`

How quickly to interpolate model toward suspension position in meters per second. If negative, position teleports immediately.

MotionEffects EWheelMotionEffectsMode

Controls whether wheel creates particle kickup effects for the ground surface material underneath.

Defaults to `BothDirections` if `ModelUseColliderPose` is true, `None` otherwise.

SteeringAngleMultiplier float `1.0`

Target steering angle is multiplied by this value. For example, can be set to a negative number for rear-wheel steering.

SteeringMode EWheelSteeringMode `Auto`

Controls whether wheel contributes to steering and if so, which kind of steering to simulate.

WheelColliderPath string

Scene hierarchy path of a WheelCollider component relative to the vehicle's root transform.

76.2 Localization

Name *string*: Vehicle name in user interfaces.

CHAPTER

SEVENTYSEVEN

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.

77.1 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
```

77.2 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.

77.3 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

SEVENTYEIGHT

VEHICLE REDIRECTOR ASSETS

Vehicle Redirector Assets help consolidate legacy colored vehicle variants into a single vehicle asset.

Prior to the addition of paintable vehicles, it was common to create duplicates of a vehicle asset with the color being the only difference. One of many downsides with this approach was the increased hassle of keeping changes consistent between all of the variants; for example, when tuning physics.

Vehicle redirector assets ensure compatibility with existing saves and content while merging colored vehicle variants into one unified asset.

78.1 Game Data File

Note that the `TargetVehicle` property is required for this asset to function.

78.1.1 Properties

Property Name	Type	Default Value
TargetVehicle	*Asset Pointer*	
LoadPaintColor	*color*	
SpawnPaintColor	*color*	

78.1.2 Property Descriptions

TargetVehicle Asset Pointer

Actual vehicle to use when attempting to load or spawn this asset.

LoadPaintColor color

If set, overrides the default random paint color when loading a vehicle from a save file. Used to preserve colors of vehicles in existing saves.

SpawnPaintColor color

If set, overrides the default random paint color when spawning a new vehicle. Optionally used to preserve colors of vehicles in spawn tables.

CHAPTER

SEVENTYNINE

OBJECT ASSETS

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

Type *EObjectType*: 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.

79.1 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".

Is_Clutter *bool*: Defaults to false. If true, object is not instantiated when player has the "Load Clutter" graphics option disabled.

> **Warning**
>
> Objects the player can collide with should never be clutter. Otherwise, the player will "rubberband" on these objects in multiplayer.

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`.

Load_Nav_On_Server *bool*: If true, Nav game object will be instantiated in singleplayer and on dedicated server. Useful for objects which need to affect navmesh baking without colliding with zombies during gameplay. Defaults to true for "medium" and "large" objects.

Load_Nav_In_Editor *bool*: If true, Nav game object will be instantiated in the level editor. Useful for objects which need collision with zombies during gameplay without affecting navmesh baking. Defaults to true for "medium" and "large" objects.

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.

Exclude_From_Satellite_Capture *bool*: If true, object will be hidden when rendering GPS/satellite view. Defaults to true if `Holiday_Restriction` is other than `None`.

79.1.1 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`.

79.1.2 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.

79.1.3 Interactables

Interactability *enum* (`None`, `Binary_State`, `Dropper`, `Note`, `Water`, `Fuel`, `Rubble`, `NPC`, `Quest`, `Dialogue`): 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.
- `Dialogue` objects open the dialogue screen - similar to NPCs - with a non-NPC appearance and custom interact text.

> **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_Animation_Component_Path *string*: (`Binary_State`-only) Transform path relative to object root with Animation component. Defaults to "Root".

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_Dialogue *GUID*: Dialogue asset to open for `Interactability Dialogue` mode. By default the object's name is used as the character name in dialog, but this can be overridden with the `Dialogue_Name` option in the localization file.

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 effect is played when a section of the object is destroyed.

Interactability_Emissive_Material_Mode *enum* (`Auto` or `None`): (`Binary_State`-only). Defaults to `Auto`, creating a material instance for child renderer of `Toggle` game object. The downside of `Auto` is exclusion from the level batching texture atlas. When set to `None`, no material instance is created.

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`): When using `Binary_State`, controls how on/off state affects `Nav` game object. Defaults to `None` which doesn't affect Nav. `On` activates Nav when object is in the on state and deactivates in the off state. (`Off` does the opposite.)

> **Note**
>
> When `Interactability_Nav` is combined with `Rubble_Nav_Mode`, the Nav game object is only active if **both** activate it. (i.e., AND)

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_RewardItem_Origin *EItemOrigin*: When using `Interactability Dropper`, overrides the dropped item's state. For example, setting the origin to `Admin` will cause items to spawn at full quality. Defaults to `Nature`.

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.

79.1.4 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_All_Sections_Destroyed_Alert_Radius *float*: If set, alert nearby enemies when all sections are destroyed.

Rubble_Blade_ID *byte*: Weapons are unable to damage this object unless they have a matching `BladeID_#` value. Defaults to 0.

Rubble_Can_Zombies_Damage *bool*: If true, zombies can attack this object if it's blocking them. Defaults to false.

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_Nav_Mode *enum* (`Unaffected`, `DeactivateIfAllDead`): Defaults to `Unaffected`. `DeactivateIfAllDead` deactivates the Nav game object when all rubble sections are destroyed, but keeps it active otherwise.

> **Note**
>
> When `Rubble_Nav_Mode` is combined with `Interactability_Nav`, the Nav game object is only active if **both** activate it. (i.e., AND)

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

Rubble_Reset *float*: Delay before a destroyed object respawns, in seconds.

Rubble_Respawn_All_Sections_Simultaneously *bool*: If true, all sections respawn at the same time. Defaults to false.

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.

Rubble_Section_Destroyed_Alert_Radius *float*: If set, alert nearby enemies when an individual section is destroyed. Not used when the final section is destroyed—in that case, **Rubble_All_Sections_Destroyed_Alert_Radius** applies instead.

Rubble_Zombie_Damage_Multiplier *float*: Multiplier for damage from zombies if RubbleCanZombiesDamage is true.

79.1.5 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_#`.

79.2 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`.

Dialogue_Name *string*: Overrides character name in dialogue. Defaults to the object name.

CHAPTER

EIGHTY

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.

80.1 Metadata

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

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

80.2 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

EIGHTYONE

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.

81.1 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.

**
**: New line.

<pause>: Pause dialogue for 0.5 seconds before continuing.

CHAPTER

EIGHTYTWO

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.

82.1 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.

82.1.1 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.

82.2 Appearance

While in the Appearance menu in-game, modders can press Page Down to copy the player's current appearance to clipboard.

Face *uint8*: Index of face image.

Hair *uint8*: Index of hair mesh.

Beard *uint8*: 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.

82.3 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.

82.4 Conditions

An NPC character can be made to only appear while certain *conditions* are met by the player.

82.5 Localization

Name *string*: Object name in level editors.

Character *string*: Character name displayed when interacted with.

CHAPTER

EIGHTYTHREE

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.

> **Tip**
>
> To quickly test a specific dialogue, you can use the `/dialogue guid` command. This opens the dialogue as if your character were an NPC.

83.1 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.

83.2 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.

83.3 Localization

Message_#_Page_# *Rich Text*: Text shown for the corresponding message page.

Response_# *Rich Text*: Text shown for the corresponding response option.

CHAPTER

EIGHTYFOUR

QUEST ASSETS

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

Type *enum* (`Quest`)

ID *uint16*: Must be a unique identifier.

84.1 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.

Quests have two types of rewards lists:

1. `Rewards` are granted when the quest is completed.
2. `AbandonmentRewards` are granted when the quest is removed without completing it. These are not granted when the player finishes the quest normally.

84.2 Localization

Name *string*: Quest name in user interfaces.

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

CHAPTER

EIGHTYFIVE

VENDOR ASSETS

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

Type *enum* (`Vendor`)

ID *uint16*: Must be a unique identifier.

85.1 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 *GUID* or *uint16*: GUID or legacy 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.

85.2 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 *GUID* or *uint16*: GUID or legacy 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 ID of a spawnpoint node as set in the level editor. For example, `Liberator_Jet`. If an ID is not provided, the vehicle will spawn above the NPC.

Selling_#_PaintColor *color*: If set, overrides color of purchased vehicle. Vehicle redirector asset's `SpawnPaintColor` and vehicle asset's `DefaultPaintColors` are bypassed.

Selling_#_Ammo *byte*: Override for the amount of ammuntion that should be loaded in the item sold.

Selling_#_Barrel *uint16*: Override for the barrel attachment that should be attached to the item sold.

Selling_#_Grip *uint16*: Override for the grip attachment that should be attached to the item sold.

Selling_#_Magazine *uint16*: Override for the magazine attachment that should be attached to the item sold.

Selling_#_Sight *uint16*: Override for the sight attachment that should be attached to the item sold.

Selling_#_Tactical *uint16*: Override for the tactical attachment that should be attached to the item sold.

85.3 Other Properties

Disable_Sorting *flag*: Disable vendor sorting.

Currency *GUID*: 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.

85.4 Localization

Name *string*: Vendor name in user interfaces.

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

Buying_#_Description *Rich Text*: If set, overrides item description shown in the vendor menu. I.e., how the vendor would describe the item rather than how the player would.

Selling_#_Description *Rich Text*: Same as **Buying_#_Description**.

CHAPTER

EIGHTYSIX

CONDITIONS

Conditions can be held by NPCs, interactable objects, and item blueprints. Each grouping of conditions is called a **conditions list** and starts with the `Conditions` property.

Properties in a conditions list are named in the format of `ConditionPrefix_#_PropertyName`. For most conditions lists the prefix is `Condition`, such as `Condition_#_Type`. This is not always the case, such as for *blueprints* which use `Blueprint_#_Conditions`.

Conditions *byte*: Total number of conditions. There should be a number of `Condition_#_Type` properties equal to this value.

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_Zombie`, `Player_Life_Food`, `Player_Life_Health`, `Player_Life_Stamina`, `Player_Life_Virus`, `Player_Life_Water`, `Quest`, `Reputation`, `Skillset`, `Holiday`, `Time_Of_Day`, `Volume_Overlap`, `Weather_Blend_Alpha`, `Weather_Status`): Specify the type of condition. Like other indexed properties, indicing starts at `0`.

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.

86.1 Flags

86.1.1 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.

86.1.2 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.

86.1.3 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.

86.1.4 Flag_Short

Short flag condition.

Condition_#_Type *enum* (`Flag_Short`)

Condition_#_ID *uint16*: ID of flag to check.

Condition_#_Value *int16*: 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.

86.2 Player

86.2.1 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.

86.2.2 Experience

Condition_#_Type *enum* (`Experience`)

Condition_#_Value *int*: Target value, in terms of experience.

86.2.3 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.

86.2.4 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.

86.2.5 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.

86.2.6 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.

86.2.7 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.

86.2.8 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.

86.2.9 Kills_Zombie

Condition_#_Type *enum* (`Kills_Zombie`)

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.

86.2.10 Player_Life_Food

Condition_#_Type *enum* (`Player_Life_Food`)

Condition_#_Value *int*: Target value, in terms of the player's current food.

86.2.11 Player_Life_Health

Condition_#_Type *enum* (`Player_Life_Health`)

Condition_#_Value *int*: Target value, in terms of the player's current health.

86.2.12 Player_Life_Stamina

Condition_#_Type *enum* (`Player_Life_Stamina`)

Condition_#_Value *int*: Target value, in terms of the player's current stamina/energy.

86.2.13 Player_Life_Virus

Condition_#_Type *enum* (`Player_Life_Virus`)

Condition_#_Value *int*: Target value, in terms of the player's current immunity.

86.2.14 Player_Life_Water

Condition_#_Type *enum* (`Player_Life_Water`)

Condition_#_Value *int*: Target value, in terms of the player's current water.

86.2.15 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.

86.2.16 Reputation

Condition_#_Type *enum* (`Reputation`)

Condition_#_Value *int*: Target value, in terms of reputation.

86.2.17 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.

86.3 World

86.3.1 Holiday

Condition_#_Type *enum* (`Holiday`)

Condition_#_Value *enum* (*ENPCHoliday*): Target value, as the holiday.

86.3.2 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.

86.3.3 Time_Of_Day

Condition_#_Type *enum* (`Time_Of_Day`)

Condition_#_Second *int*: Second of a 24-hour clock (military time) to compare against. For example: `0` is midnight (the start of a day), `43200` is noon (12 o'clock), and `86400` is midnight (the end of a day). This condition respects the map's configured "Bias" values, as well as the day/night cycle length of the world. As a visual reference, the Clock item can be used.

86.3.4 Volume_Overlap

Condition_#_Type *enum* (`Volume_Overlap`)

Condition_#_VolumeID *string*: ID of volume(s) placed in the level editor to test.

Condition_#_PlayerCount *int*: Target number of players in matching volumes to compare against. Volumes with identical IDs are grouped together.

86.3.5 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.

86.3.6 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.

86.4 Localization

Condition_#: Name of the condition as it appears in user interfaces.

CHAPTER

EIGHTYSEVEN

REWARDS

Rewards can be granted by NPCs, objects, and items. Each grouping of rewards is called a **rewards list** and starts with the `Rewards` property.

Properties in a rewards list are named in the format of `RewardPrefix_#_PropertyName`. For most rewards lists the prefix is `Reward`, such as `Reward_#_Type`. This is not always the case, such as with *quests* which have two separate rewards lists named `Rewards` and `AbandonmentRewards`, or consumables which use `Quest_Rewards`.

Rewards *byte*: Total number of rewards in the rewards list. There should be a number of `Reward_#_Type` properties equal to this value.

Reward_#_Type *enum* (`Airdrop`, `Flag_Bool`, `Flag_Math`, `Flag_Short`, `Flag_Short_Random`, `Achievement`, `Currency`, `Cutscene_Mode`, `Effect`, `Event`, `Experience`, `Item`, `Item_Random`, `Hint`, `Player_Life_Food`, `Player_Life_Health`, `Player_Life_Stamina`, `Player_Life_Virus`, `Player_Life_Water`, `Player_Spawnpoint`, `Quest`, `Reputation`, `Rewards_List_Asset`, `Teleport`, `Vehicle`, `Zombie`, `Remove_Zombies`): Specify the type of reward. Like other indexed properties, indicing starts at `0`.

Reward_#_GrantDelaySeconds *float*: If set, the reward will be queued for the specified number of seconds before being granted to the player. Defaults to -1.

Reward_#_GrantDelayApplyWhenInterrupted *bool*: If true, the reward will be granted when the player dies or disconnects. Defaults to false, in which case when the player dies any pending rewards are cancelled.

87.1 Flags

87.1.1 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".

87.1.2 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`, `Random_Inclusive`, `Random_Exclusive`): For example, using the Addition operation would set A to the value of A + B.

`Random_Inclusive` operation: Set flag A to random number between the value of A and B. For example if A is 1 and B is 3 the random number could be 1, 2, or 3.

`Random_Exclusive` operation: Set flag A to random number between the value of A and B, excluding B. For example if A is 1 and B is 3 the random number could be 1 or 2. If the value of A and B are the same then the exclusion rule is ignored.

87.1.3 Flag_Short

Reward_#_Type *enum* (`Flag_Short`)

Reward_#_ID *uint16*: ID of flag to modify.

Reward_#_Value *int16*: Modify flag's current value with this short value.

Reward_#_Modification *enum* (`Assign`, `Decrement`, `Increment`): Set value, subtract value, or add value.

87.1.4 Flag_Short_Random

Reward_#_Type *enum* (`Flag_Short_Random`)

Reward_#_ID *uint16*: ID of flag to modify.

Reward_#_Min_Value *int16*: Minimum short value to modify flag's current value by.

Reward_#_Max_Value *int16*: Maximum short value to modify flag's current value by.

Reward_#_Modification *enum* (`Assign`, `Decrement`, `Increment`): Set value, subtract value, or add value.

87.2 Non-flags

87.2.1 Achievement

Reward_#_Type *enum* (`Achievement`)

Reward_#_ID *string*: ID of achievement to grant. Only specific achievements can be granted as a reward.

87.2.2 Airdrop

Reward_#_Type *enum* (`Airdrop`)

Reward_#_Use_Random_Airdrop_Node *bool*: If true, call in airdrop at a random airdrop node placed in the level editor.

Reward_#_Cargo *GUID or uint16*: Optional spawn table ID overriding which items to drop.

Reward_#_Spawnpoint *string*: Location to call in the airdrop, using the spawnpoint name as set in the nodes level editor.

87.2.3 Currency

Refer to *Currency* documentation.

Reward_#_Type *enum* (`Currency`)

Reward_#_GUID *string*: GUID of currency asset.

Reward_#_Value *int*: Amount of currency to reward.

87.2.4 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.

87.2.5 Effect

Reward_#_Type *enum* (`Effect`)

Reward_#_GUID *Asset Pointer*: *Effect Asset* to spawn.

Reward_#_Spawnpoint *string*: Location to spawn the effect, using the spawnpoint name as set in the nodes level editor. For example, `Liberator_Jet`.

Reward_#_AtPlayerPosition *bool*: If true, spawn the effect at the target player's position.

Reward_#_IsReliable *bool*: If true, multiplayer will ensure the effect is replicated. If false, it won't be retransmitted if the packet is lost. Defaults to true.

Reward_#_RelevantDistance *float*: If set, overrides the default multiplayer relevant distance of 128 meters. Defaults to -1.

Reward_#_OnlyRelevantToInstigator *bool*: If true, only the player triggering this reward will see the effect. Defaults to false. Takes priority over `RelevantDistance` if set.

87.2.6 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.

Reward_#_Replicate *bool*: If true, event is triggered on clients as well as the server. Defaults to true. If false, event is only triggered on authority (singleplayer/server).

Reward_#_InstigatorOnly *bool*: If true, event is run only for the player triggering the reward. Defaults to false. Takes priority over `Replicate`. In multiplayer this means the event is only broadcast for the triggering client, not the server.

87.2.7 Experience

Reward_#_Type *enum* (`Experience`)

Reward_#_Value *int*: Amount of experience to reward.

87.2.8 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*: If true, the item should be automatically equipped by the player (if possible). Auto-equipping isn't possible if another item in the same slot is already equipped. Defaults to false.

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.

87.2.9 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 *bool*: If true, the item should be automatically equipped by the player (if possible). Auto-equipping isn't possible if another item in the same slot is already equipped. Defaults to false.

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`.

87.2.10 Hint

Reward_#_Type *enum* (`Hint`)

Reward_#_Text *Rich Text*: Debug text that is shown when the asset's *localization file* is empty. If a localization file has been included then the localized text is used instead.

Reward_#_Duration *float*: Duration of the hint, in seconds. Defaults to 2 seconds.

> **Note**
>
> For localized hints to work in multiplayer, please add `Keep_Localization_Loaded true` to the owning asset. Otherwise, the server's language is used.
>
> This is necessary because the server doesn't currently have a way to reference the reward itself (which the client has the text loaded for), instead the asset ID and localization key are replicated.

87.2.11 Player Life Food

Reward_#_Type *enum* (`Player_Life_Food`)

Reward_#_Value *int*: Amount of food to add. Can be negative to decrease food.

87.2.12 Player Life Health

Reward_#_Type *enum* (`Player_Life_Health`)

Reward_#_Value *int*: Amount of health to add. Can be negative to decrease health.

87.2.13 Player Life Stamina

Reward_#_Type *enum* (`Player_Life_Stamina`)

Reward_#_Value *int*: Amount of stamina/energy to add. Can be negative to decrease stamina level.

87.2.14 Player Life Virus

Reward_#_Type *enum* (`Player_Life_Virus`)

Reward_#_Value *int*: Amount of virus to add. Can be negative to decrease virus level.

87.2.15 Player Life Water

Reward_#_Type *enum* (`Player_Life_Water`)

Reward_#_Value *int*: Amount of water to add. Can be negative to decrease water.

87.2.16 Player Spawnpoint

Reward_#_Type *enum* (`Player_Spawnpoint`)

Reward_#_ID *string* Override the player's default spawn location, using the ID of a spawnpoint node or the name of a map location node, as set in the level editor. 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.

87.2.17 Quest

Reward_#_Type *enum* (`Quest`)

Reward_#_ID *uint16*: Quest ID to give as a reward.

87.2.18 Remove Zombies

Kills zombies that match a set of filters.

Reward_#_Type *enum* (`Remove_Zombies`)

Reward_#_Zombie *enum*: Which "type" of zombie to remove, the same list as zombie kills condition. To-do: move to its own page. Defaults to `None` in which case all zombie types match.

Reward_#_LevelTableOverride *int*: Unique ID of a zombie type shown in the level editor. If set, only zombies spawned from this table are removed. Defaults to `-1` in which case all zombie tables match.

Reward_#_Nav *byte*: Index of the navmesh that zombies should be removed from, seen as visible in the level editor. Defaults to `255` in which case all navmeshes match.

87.2.19 Reputation

Reward_#_Type *enum* (`Reputation`)

Reward_#_Value *int*: Amount of reputation to reward.

87.2.20 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.

87.2.21 Teleport

Reward_#_Type *enum* (`Teleport`)

Reward_#_Spawnpoint *string*: Location to teleport the player to as a reward, using the ID of a spawnpoint node as set in the level editor. For example, `Liberator_Jet`.

87.2.22 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 ID of a spawnpoint node as set in the level editor. For example, `Liberator_Jet`. If an ID is not provided, the vehicle will spawn above the NPC.

Reward_#_PaintColor *color*: If set, overrides color of spawned vehicle. Vehicle redirector asset's `SpawnPaintColor` and vehicle asset's `DefaultPaintColors` are bypassed.

87.2.23 Zombie

Respawns zombie(s) at named Spawnpoint nodes. If insufficient dead zombies are available to respawn, alive zombies are converted to the appropriate type and teleported.

> **Note**
>
> Spawnpoints must be within a navmesh.

Reward_#_Type *enum* (`Zombie`)

Reward_#_Zombie *enum*: Which "type" of zombie to spawn, the same list as zombie kills condition. To-do: move to its own page.

Reward_#_Spawnpoint *string*: Where to spawn the zombies, using the name as set in the nodes level editor. When multiple nodes share a name each zombie is spawned at a random node.

Reward_#_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.

Reward_#_SpawnQuantity *int*: Amount of zombies to spawn.

Reward_#_CooldownId *string*: If set, only spawn if a named global cooldown (shared between all players) has passed.

Reward_#_CooldownDuration *float*: Seconds since CooldownId last ran before this reward can spawn zombies again.

87.3 Localization

Rewards lists properties can be localized in the asset's localization file.

Reward_#: Localization for the name of the reward as it appears in user interfaces.

Tip

Localization properties follow the same rules as other parts of the rewards list – including needing to use the correct prefix. For example, the property name for a localized hint on an interactable object would be `Interactability_Reward_#`.

CHAPTER

EIGHTYEIGHT

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.

Tip: you can use the `/RunRewardList (guid)` admin command to test a reward list in-game.

CHAPTER

EIGHTYNINE

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.

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

89.1 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.ItemCurrencyAsset`

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.

89.2 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.

89.3 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

NINETY

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

NINETYONE

ANIMAL ASSETS

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

Type *enum* (`Animal`)

ID *uint16*: Must be a unique identifier.

91.1 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.

Should_Prevent_Move_During_Startle *bool*: If true, animal won't start moving until startle animation finishes. Defaults to false.

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.

91.2 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.

91.3 Localization

Name *string*: Animal name in user interfaces.

CHAPTER

NINETYTWO

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.

92.1 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

NINETYTHREE

CRAFTING ASSETS

This asset defines blueprints outside of an item asset. This may be useful for organizational purposes or blueprints without a clear "owner" item. For example, if a blueprint has multiple inputs and outputs it might not make sense to include in a specific item asset.

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

Type `SDG.Unturned.CraftingAsset`

93.1 Properties

The same *Blueprints* property as items. However, only the V2 blueprints format is supported here.

CHAPTER

NINETYFOUR

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 one of `BlueprintName` string or `Blueprint` index. For example, to prevent the Chef Hat from being salvaged:

```
Blueprints
[
        {
                Item a6099002318e4d58b8e59d431bcf1b8a
                BlueprintName Salvage
        }
]
```

Note

`BlueprintName` should be used instead of `Blueprint` (index) where possible because the index may change if the item's Blueprints list is reorganized. This requires specifying a `Name` in the blueprint, however.

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

CHAPTER

NINETYFIVE

EFFECT ASSETS

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

Type *enum* (`Effect`)

ID *uint16*: Must be a unique identifier.

95.1 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.

OneShotAudio *Master Bundle Pointer*: AudioClip or OneShotAudioDefinition to play alongside effect. Useful for audio-only effects, in which case Effect prefab is unnecessary and can be excluded.

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.

95.2 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.

95.3 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

NINETYSIX

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.

96.1 FoliageResourceInfoAsset Properties Reference

Type *string*: `SDG.Framework.Foliage.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.

Draw_Distance *int32*: Sets how far from the camera the foilage is drawn for instanced-mesh foilage (grass, pebbles, small details). Minor performance option with noticeable impact on graphical fidelity.

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.

Uniform_Scale *bool*: If true, max and min scale are floats rather than vector3. This enables an optimization for instanced mesh foliage like grass and pebbles, packing more instances per batch. This requires shaders that support uniform scaling, such as `Framework/Grass (Uniform Scaling)` and `Framework/Detail (Uniform Scaling)`.

96.2 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

NINETYSEVEN

ROAD ASSETS

This asset allows roads to be shared between levels, and exposes some previously-unavailable properties for configuration.

97.1 Properties

Property Name	Type	Default Value
Chart	*EObjectChart*	`None`
Depth	*float32*	`0.0`
OffsetAlongNormal	*float32*	`0.0`
PhysicsMaterial	*string*	See description
RepeatDistanceScale	*float32*	`1.0`
TexturePath	*string*	See description
Width	*float32*	`0.0`
VanillaPhysicsMaterial	*enum*	See description

97.1.1 Chart EObjectChart `None`

If not `None`, overrides how road appears in chart generation.

If `None` (default) legacy classification is used:

- Concrete roads wider than 16 meters are `Highway`.
- Other concrete roads are `Road`.
- Non-concrete roads are `Path`.

97.1.2 Depth float32 `0.0`

Total size along the up axis.

> **Note**
>
> If converting a legacy road configuration, please note that Depth shown in the legacy editor is actually *half* the total depth.

97.1.3 OffsetAlongNormal float32 `0.0`

Distance along the terrain surface normal to move each road vertex.

97.1.4 PhysicsMaterial Master Bundle Pointer to `PhysicMaterial`

Unity `PhysicMaterial` to apply to road collider. Not used if *VanillaPhysicsMaterial* is assigned.

97.1.5 RepeatDistanceScale float32 `1.0`

By default, Texture is uniformly scaled along the road according to its aspect ratio and the road's width. For example, if *Width* is 8 meters and Texture is 256x128 pixels, the texture will repeat every 4 meters.

This property multiplies the distance along the road before the texture repeats.

> **Note**
>
> If converting a legacy road configuration, the repeat distance was the texture's height in pixels divided by the Height value. For example: a 64x64 pixel texture with Height of 2 would repeat every 32 meters. To calculate an equivalent RepeatDistanceScale, divide the legacy repeat distance by the road asset's Width.

97.1.6 TexturePath Master Bundle Pointer to `Texture2D`

If not specified, the game looks for a texture named `Texture` in the accompanying asset bundle.

97.1.7 Width float32 `0.0`

Total horizontal size before the road begins tapering off into the terrain.

> **Note**
>
> If converting a legacy road configuration, please note that Width shown in the legacy editor is actually *half* the total width.

97.1.8 VanillaPhysicsMaterial string

Optional name of a built-in Unity `PhysicMaterial` to apply to road collider. For example, "Concrete_Static" or "Gravel_Static" for the legacy road physics materials.

CHAPTER

NINETYEIGHT

LANDSCAPE MATERIAL ASSET

Landscape materials represent textures that can be applied to map terrain. Each asset specifies the texture drawn on the terrain, the physics material used for footstep and impact audio, and optionally foilage to be baked on wherever the material is painted.

AprilFools_Redirect *GUID*: GUID of the landscape material to use instead during the April Fools event.

Auto_Max_Angle_Begin *float*: Slope angle at which the auto-painted weight begins to fade out. Defaults to `0.0`.

Auto_Max_Angle_End *float*: Steepest slope angle that is auto-painted. Defaults to `0.0`.

Auto_Min_Angle_Begin *float*: Shallowest slope angle that is auto-painted. Defaults to `0.0`.

Auto_Min_Angle_End *float*: Slope angle at which auto-painted weight is full strength. Defaults to `0.0`.

Auto_Ray_Length *float*: Length of the sphere cast downwards when utilizing auto foundation. This should be set when using auto foundation painting. Defaults to `0.0`.

Auto_Ray_Mask *ERayMask*: Value of the raymask layers the auto foundation sphere cast can hit. Defaults to `0`.

Auto_Ray_Radius *float*: Radius of the auto foundation sphere cast. Defaults to `0.0`.

Christmas_Redirect *GUID*: GUID of the landscape material to use instead during the Christmas event.

Foliage *GUID*: Points to a foilage collection to bake wherever this material is painted. Refer to *Foliage Assets*.

Halloween_Redirect *GUID*: GUID of the landscape material to use instead during the Halloween event.

Mask *Master Bundle Pointer*: Normal map for the terrain material. Despite its name, the terrain material's normal map is set to this texture. If not specified, a black texture is used.

Physics_Material *string*: Name of the physics material used. Determines footstep sounds, bullet impacts, and other effects. This is matched against the `UnityName` of *Physics Material Assets*.

Texture *Master Bundle Pointer*: Albedo texture drawn on the terrain. When unspecified, a black texture is used.

Use_Auto_Foundation *bool*: If true, the level editor's auto splatmap mode paints this material at full strength beneath objects.

Use_Auto_Slope *bool*: If true, the level editor's auto splatmap mode paints this material according to the terrain's slope angle.

CHAPTER

NINETYNINE

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.

Skillset_Loadouts *dictionary*: Overrides per-skillset starting items. Can be used to prevent skillset default items in singleplayer. Server "Loadout" command takes priority over this option. Please refer to Skillset Loadout properties below for more details.

TerrainColors *array*: Specifies which colors are too similar to terrain colors. Please refer to Terrain Color Properties below.

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.

Death_Music *Master Bundle Pointer*: Audio clip played after death.

UnderwaterFogDensity *float*: Overrides fog effect intensity while the camera is underwater. Defaults to `0.075`.

Allow_Building_In_Safezone_In_Singleplayer *bool*: If true, players can bypass safezone's no-buildables mode in singleplayer. Defaults to false.

Tags *list* of *Asset Pointer* to *Tag Assets*: Blueprints can test for these tags as an alternative to `Map` name check. May be extended in the future.

Supports_Fishing_Volumes *bool*: If true, this level has assigned fishing spawn tables to water volumes (or set the default table). Defaults to false.

Default_Fish_Spawn_Table *Asset Pointer*: Fishing rods using per-water-volume fishing spawn table fallback to this table.

ZombieDifficultyAssetPrioritization *enum* (`NavmeshOverridesTable`, `TableOverridesNavmesh`): Allows the level to override whether per-table or per-navmesh difficulty asset takes priority. By default, `NavmeshOverridesTable` is used. Using `TableOverridesNavmesh` instead allows for the level to override the difficulty of zombies based on the spawn table, even if the navmesh has a different difficulty asset assigned.

99.1 Cloud Override Properties

When clouds are disabled (`Has_Clouds false`), these properties can be used to control a custom particle system using the cloud color and intensity from the lighting settings.

> **Warning**
>
> Custom clouds are *not* recommended as they behave inconsistently at lower max draw distances.
>
> Early versions of 3.x had a very high max draw distance (~8 km). The sun, moon, clouds, and stars were 3D objects scaled almost to the edge of the far clip plane.
>
> Later versions combined these details into a skybox shader which enabled the max draw distance to be configured by players, but at the cost of reduced customization.
>
> As a workaround, mappers created 3D cloud particles attached to the camera. Distant particles were cut off at the far clip plane, but a shader with the `ZClip False` resolved this.
>
> Unfortunately, it seems after the engine update in 3.25.9.2 this workaround broke: the particle system can be culled at low draw distances. This is understandable from the engine's perspective, as using the particle system's origin for culling is suboptimal. As of March 2026 we're not aware of a workaround to force the particle system to render.

CloudOverride_Prefab *Master Bundle Pointer*: Prefab to instantiate and attach to lighting.

CloudOverride_ParticleSystems *list*: Dictionaries describing particle systems in `CloudOverride_Prefab` with the following additional properties:

Path *string*: Path to a Particle System component relative to `CloudOverride_Prefab`. This renderer should likely use the Unturned "Particles/Standard Surface (ZClip False)" shader to prevent a harsh transition with low max draw distance.

RateOverTimeScale *float*: Particle System's emission rate is multiplied by current time of day's cloud slider value (between zero and one) scaled by this value.

MaterialColorPropertyNames *list*: Particle System's material instance will have these color properties set to the current time of day's cloud color. Defaults to just _Color.

WarmupTime *float*: When restarting the particle system it is simulated this number of seconds. Alternative to enabling Particle System's `Prewarm` option. Defaults to zero.

For example:

```
1 CloudOverride_ParticleSystems
2 [
```

(continues on next page)

(continued from previous page)

```
        {
                Path System1
                RateOverTimeScale 1.5
                WarmupTime 10
                MaterialColorPropertyNames
                [
                        _Color
                        _EmissionColor
                ]
        }
        {
                Path System2
                RateOverTimeScale 3
                MaterialColorPropertyNames
                [
                        _Color
                        _EmissionColor
                ]
        }
]
```

99.2 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.

99.3 Skill Rule Properties

Id *string*: Name of skill, for example Sharpshooter.

Default_Level *int*: Skill level when player spawns. The `Spawn_With_Max_Skills` gameplay config option takes priority.

Max_Unlockable_Level *int*: Maximum skill level attainable through gameplay. Higher levels are hidden in the skills menu.

Base_Cost *int*: If set, overrides XP cost to purchase first level of the skill.

Per_Level_Cost_Increase *int*: If set, overrides XP cost increase with each level. Added to `Base_Cost` after the first level.

Cost_Multiplier *float*: Multiplier for total XP upgrade cost.

```
Skills
[
        {
```

(continues on next page)

(continued from previous page)

```
                Id Overkill
                Default_Level 0
                Max_Unlockable_Level 0
        }
        {
                Id Parkour
                Default_Level 2
                Max_Unlockable_Level 2
        }
        {
                Id Crafting
                Default_Level 1
                Max_Unlockable_Level 3
                Cost_Multiplier 5
        }
]
```

99.4 Skillset Loadout Properties

Can contain the following keys: `None`, `Fire`, `Police`, `Army`, `Farm`, `Fish`, `Camp`, `Work`, `Chef`, `Thief`, `Medic`

Each key is a list of items with the following properties:

Asset: *Asset Pointer*: Item or spawn table to grant an item from.

Amount *int*: Number of times to grant this item. Defaults to 1.

Origin *EItemOrigin*: Determines starting state of the item. Defaults to World.

Example:

```
Skillset_Loadouts
{
        Army
        [
                {
                        // Eaglefire at max quality with full ammo
                        Asset 4
                        Origin Admin
                }
                {
                        // Military magazine x2 with random ammo
                        Asset dbfb1d0d11ca438e9dffb95f76e61274
                        Amount 2
                }
        ]

        // Other skillsets will spawn with nothing
}
```

99.5 Terrain Color Properties

Color *color*: Actual base color/albedo of terrain material. Players will be kicked from multiplayer servers if their customized skin color is too similar to the value of this property.

HueThreshold *float32*: Values are clamped from 0 to 1. If difference between hues is greater than this threshold, the colors are not too similar.

SaturationThreshold *float32*: Values are clamped from 0 to 1. If difference between saturations is greater than this threshold, the colors are not too similar.

ValueThreshold *float32*: Values are clamped from 0 to 1. If difference between values is greater than this threshold, the colors are not too similar.

99.6 Music Properties

Loop *Master Bundle Pointer*: Looping audio clip played until loading finishes.

Outro *Master Bundle Pointer*: Audio clip played once loading finishes.

MYTHICAL EFFECT ASSETS

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

Type *enum* (`Mythic`)

ID *uint16*: Must be a unique identifier.

> **Note**
>
> At the time of writing (2025-02-04) mythical effects aren't moddable, sorry! This document is primarily for item creators working on curated updates, as well as preparation for eventual open sourcing.

When testing, we recommend attaching your mythicals to some items' Effect transform in the editor and resetting the local transform. This helps visualize how they will appear in-game.

100.1 Unity Prefabs

System_Area: Instantiated multiple times to cover a larger area. Mythical shirts and pants attach them to character bones, and mythical vehicle skins attach them to several points on the surface. As such, this shouldn't rely on any specific orientation. Layer should be set to Enemy.

System_Hook: Attached to a cosmetic item's "Effect" prefab. Layer should be set to Enemy.

For silly legacy reasons the orientation is rather unfortunate: +Z is the mythical's up direction and +Y is the mythical's forward direction.

System_Third: Attached to third-person weapon skin. Unlike for cosmetics, +Z is forward and +Y is up. It may be helpful to increase particle size to make them more visible over the shoulder. Emission typically uses a box of size 25x25x50 cm. Layer should be set to Item.

System_First: Attached to first-person weapon skin. Particles should be smaller than usual, and try to avoid the up direction to reduce blocking the player's aim. First-person particles are typically half the size, but use the same emission shape size as third-person. Layer should be set to Viewmodel.

100.2 Localization

Particle_Tag_Name *string*: When a mythical item is crafted with this effect, this is the display name shown in the inventory.

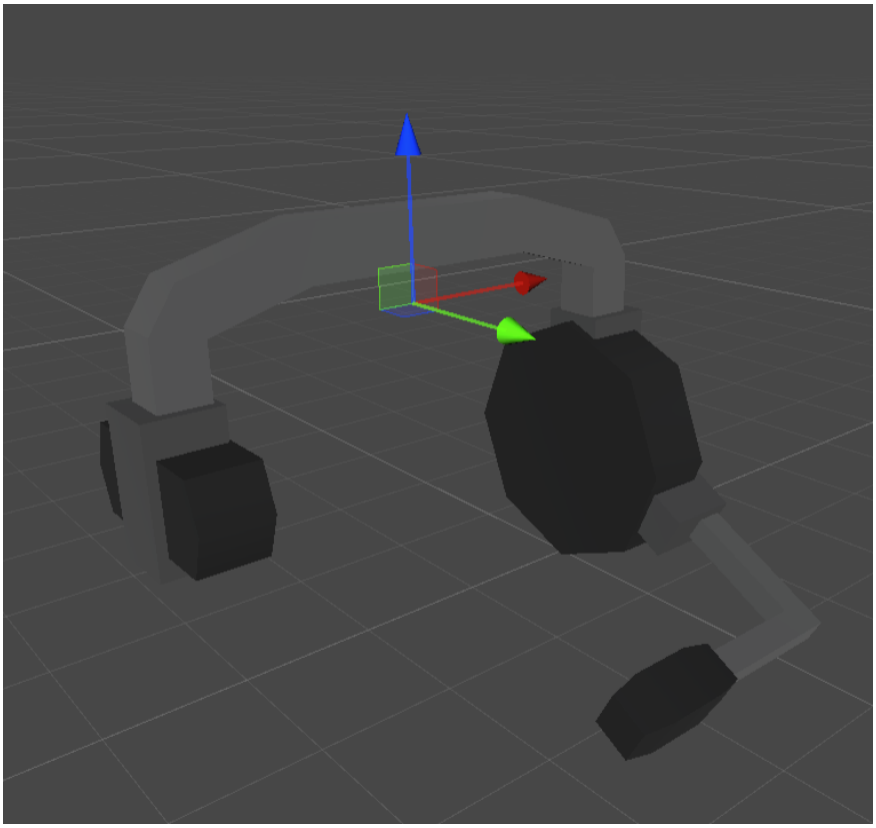


Fig. 1: Example “Effect” transform positioning and orientation.

CHAPTER

ONE

OUTFIT ASSETS

The OutfitAsset class allows for defining 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.

101.1 Metadata

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

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

101.2 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

TWO

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.

102.1 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 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.

- ZombieBipedLand: zombie equivalent of BipedLand. If unassigned, BipedLand is used with 85% pitch scaling.
- ZombieFootstepRun: zombie equivalent of FootstepRun. If unassigned, FootstepRun is used with 85% pitch scaling.
- ZombieFootstepWalk: zombie equivalent of FootstepWalk. If unassigned, FootstepWalk is used with 85% pitch scaling.
- MegaZombieFootstep: mega zombie footstep sound. If unassigned, FootstepWalk is used with 72% pitch scaling and +50% volume.
- MegaZombieLand: mega zombie landing sound. If unassigned, BipedLand is used with 72% pitch scaling and +50% volume.

TireMotionEffect *Asset Pointer* to *Effect Asset*: Effect to spawn while driving on this material. Its transform is set to the ground hit position with the Z axis aligned to the wheel's up vector and rotated according to forward/reverse speed. The vanilla effects use the rate over distance emission mode.

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 $2 m/s^2$.

`Character_Max_Speed_Multiplier` *float*: Allows speed to reach up to this multiplied by the target move speed.

CHAPTER

THREE

REDIRECTOR ASSETS

Redirector Assets are a special asset type only used when resolving asset references (GUIDs and legacy IDs). When an asset reference points to a redirector, the asset system instead returns the asset pointed to by the redirector.

> **Note**
>
> Most features do not save the resolved asset, instead they save the original asset reference. This means, for example, that redirecting some objects and re-saving a level will save the original object references, not the redirected objects.

103.1 Game Data File

Note that `TargetAsset` is required for this asset to function.

103.1.1 Properties

Property Name	Type	Default Value
AssetCategory	*enum*	`None`
TargetAsset	*GUID*	

103.1.2 Property Descriptions

AssetCategory EAssetType `None`

If set, an asset's legacy ID can be redirected as well. For example: a redirector with `AssetCategory Item` that is pointing to an asset with the legacy ID of `4`, would be found when using `/give 4`.

TargetAsset GUID

GUID of actual asset to use when an asset reference points to the redirector asset.

CHAPTER

FOUR

RESOURCE ASSETS

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

Type *enum* (`Resource`)

ID *uint16*: Must be a unique identifier.

104.1 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.

Debris_Vertical_Offset *float*: Distance along tree's local up axis to offset debris spawn position. Defaults to 1.0.

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`.

Ignore_Collision_Between_Stump_And_Debris *bool*: If true, prevent collisions between falling tree and the stump. (i.e., debris can fall through stump) Defaults to true.

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.

RandomAngleDeviation_Max *float*: Maximum angle in degrees away from the up direction. Defaults to 5.

RandomAngleDeviation_Min *float*: Minimum angle in degrees away from the up direction. For example, can be set to 0 to allow the tree to be perfectly upright, or a higher value to prevent the tree from ever being upright. Defaults to 5.

RandomUniformScale_Max *float*: Maximum scale. The same randomized value is used for all axes (uniform). Defaults to 1.1 for backwards compatibility.

RandomUniformScale_Min *float*: Minimum scale. The same randomized value is used for all axes (uniform). Defaults to 1.1 for backwards compatibility.

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 tree's in-game scale is a random number between 1.1 and the result of `1.1 + (Scale * 2)`.

Deprecated since version 3.24.7.0: Scale is replaced by the `RandomUniformScale_Min` and `RandomUniformScale_Max` properties.

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.

104.2 Localization

Name *string*: Resource name in user interfaces.

Interact *string*: Override the text shown for interactable resources using the `Forage` flag.

CHAPTER

FIVE

SERVER BROWSER CURATION ASSETS

Most of the information for this asset type is covered in *Server Browser Curation*.

Type: `ServerCuration`

Icon: Unlike curation lists hosted on the web, the asset can include a 32x32 Icon texture in an asset bundle.

Name: If a localization file is present, the display name is loaded from that instead of the asset file.

> **Note**
>
> The asset is not enabled by default when subscribed to. It needs to be enabled from the Curation menu in the Server Browser.

CHAPTER

SIX

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.

106.1 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.

106.2 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.

106.3 Exporting Legacy Spawn Tables

Legacy spawn tables can be created within the level editor. For the most, using the legacy spawn system is probably fine for most modded maps. But if you would like to take advantage of automatically keeping up-to-date with new official content—along with better supporting modded content on your map—then you should create spawn assets instead.

You might have already created some legacy spawn tables within the level editor. Fortunately, it is fairly straightforward to export those tables as spawn assets!

Open your map in the level editor. When you are ready to export, open the pause menu. Next to the button called "Legacy Spawns", enter a number above `1000` that should be used as the starting ID for your converted tables. Clicking the "Legacy Spawns" button will now generate spawn assets based on your legacy spawn tables.

This process will usually only take a couple seconds. As part of this process, all of the legacy spawn tables on your map will be automatically updated to point to your newly-created spawn assets instead. You should save the game before exiting.

These files will appear within a subfolder named "Exported_Legacy_Spawn_Tables", located within your map's folder. Root tables are prefixed with the map's name, while the converted tiers have been prefixed with the table's name. To start using your converted tables, create a folder named "Bundles" within your map folder, and move the contents of the "Exported_Legacy_Spawn_Tables" folder into it.

Tip

You can use the "Proxy Tables" button to generate empty spawn asset files instead! This is a quick way to get started with creating spawn assets on a newly-created map, where you do not have any legacy spawn tables that need to be converted.

Along with converting the map's legacy spawn tables, the game will generate a spreadsheet named "IDs.csv". This spreadsheet can be used to more easily keep track of the IDs of each of your spawn assets.

CHAPTER

SEVEN

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.

107.1 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

EIGHT

TAG ASSETS

Although **tags** have some display properties, their primary purpose is to act as a unique identifier shared across multiple assets. This makes them especially useful for cross-mod compatibility.

Currently, tags are used in two ways:

1. Workstation crafting capabilities.
2. Blueprint categorization.

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

Type `Tag`

108.1 Tag Properties

NameColor *color*: Optional override for color of name label in the user interface.

HasIcon *bool*: If true, the game will expect an Icon texture in the asset bundle. Defaults to true.

IconPath *Master Bundle Pointer*: Overrides where to load Icon texture from.

TintIcon *bool*: If true, tints the icon according to player's foreground color preference. Defaults to true. Tinted icons are typically white #ffffff to fully change color. If your icons are colorful you may want to set this to false.

108.2 Localization

Name *string*: Display name in user interfaces.

CHAPTER

NINE

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*.

109.1 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
```

109.2 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.

Ambient_Audio_Takes_Priority_Over_Ambiance_Volumes *bool*: If true, `Ambient_Audio_Clip` causes per-ambiance-volume audio to fade out. Defaults to false.

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.

Fish_Bite_Interval_Multiplier *float*: Multiplier for interval before a fish takes the bait. Defaults to 1.

109.3 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.

109.4 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.

109.5 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.

109.6 NPC Conditions

Global weather state and current weather intensity blend can be tested through NPC conditions. Refer to *Conditions* documentation for documentation.

CHAPTER

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`.

110.1 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.

Speciality_Health_Override_Mode `None`, `MultiplyEditorHealth`, `MultiplyDefaultHealth`, or `Replace`: defaults to `None`. If set, allows zombie health to overridden per-zombie-type.

- `None`: Do not override zombie health.
- `MultiplyEditorHealth`: Per-speciality value is a multiplier for health configured in the level editor.
- `MultiplyDefaultHealth`: Per-speciality value is a multiplier for vanilla health value.
- `Replace`: Per-speciality value replaces zombie's health.

Speciality_Health_Overrides *dictionary*: If using `Speciality_Health_Override_Mode`, this pairs zombie speciality (e.g., Crawler, Sprinter, Flanker, etc) to a number. For example:

```
Speciality_Health_Overrides
{
        Crawler 100
        Burner 200
}
```

110.2 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

ONE

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".

111.1 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.

111.2 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

TWO

CURATED MAPS

> **Warning**
>
> This program is still evolving. Some of this information could change, especially once we begin to accept the first maps under this new program!

Community-created maps that are officially linked from in-game are considered **Curated Maps**. Players can experience more high-quality content, and modders can earn money from their work while getting their creations highlighted.

We have introduced a new Curation Program for 2025 and beyond. Creating maps is intended to be a fun and rewarding hobby, and our new program should be more accessible to creators.

112.1 How does a map get accepted?

Our streamlined program starts with you simply publishing a map to the Steam Workshop. It looks something like this:

1. You publish a cool map on the Steam Workshop.
2. We reach out to authors of cool maps on the Steam Workshop.
3. Legal agreements (e.g. establishing revenue share between co-authors).
4. You make any changes necessary (if any).
5. We commission artists (or you create) the contents of store bundles released alongside the map.
6. Your curated map is added to the game.

While creating your map, we recommend gathering feedback from other players! Maps can be accepted at any time after being published, so authors can create and revise content without much worry.

112.2 Eligibility / Guidelines

Several factors may influence our decisions when accepting new maps. First and foremost – curated maps should meet our guidelines (detailed further below).

> **Tip**
>
> We encourage working on projects even if they would not ineligible for curation. Creating a variety of Workshop content creates a valuable portfolio for future curation attempts, and may still benefit from being featured.
>
> Although not every map should be curated, we often feature non-curated maps in blogposts. Some particularly high-quality maps have received revenue share from Stockpile bundles.

Following these guidelines will significantly improve your eligibility:

112.2.1 Include Custom Content

Maps should feel new and exciting to players. Although we may accept maps that primarily use vanilla assets, we strongly believe including some amount of custom content on new maps is important.

This helps make each map feel unique. Consider creating new items, having a unique architecture style for your towns, incorporating our modding features in innovative ways, and exploring game mechanics in different ways.

112.2.2 No Third-Party Content

All content on the map must be an official asset from the base game, or be a custom asset that was created by you. We cannot accept maps that use assets from other people's mods (including other curated maps). Our modding documentation can help you get started!

112.2.3 Art Style

Curated experiences should still look and feel like *Unturned*. Any custom content on the map should generally match the base game's art style (similar to preexisting curated maps).

Human-like drawings such as faces and skulls should be blocky, similar to the in-game characters. This is a common source of revision requests from us.

112.2.4 Quality Assurance

- **Asset Validation**: Running the game with the *-ValidateAssets* command-line flag should not produce any warnings or errors.
- **English Text**: Having at least one English-speaking member on the mod team is recommended. Clear grammar, spelling, and punctuation are important. While native English speakers can often understand poorly written text, clear and correct English is much more accessible for non-native readers and helps simplify translation into other languages.
- **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 "HawaiiMasterBundle" 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.

112.2.5 Content Appropriacy

Content should be what is typically considered "family-friendly". For example:

- Text should be devoid of harsh profanity (anything considered heavy or mild cursing, slurs, or strongly implied). Some alternatives to traditional profanity include nonsense words (such as *gosh*, *darn*, *dang*, *drats*, and *heck*), cut-off text, or redactions (e.g., *[REDACTED]* or *[UNINTELLIGIBLE]*).
- Explicit depictions of drugs, alcohol, and other substances is not allowed. Similar ideas with a looser association, such as berry mixes instead of alcohol; or things like vineyards, bottles, kegs, or distilleries; is allowed. Cigarettes, vaping, smoking, and tobacco are not allowed.
- Text should be gender-neutral. For example, "Firefighter" instead of "Fireman."
- Do not link to out-of-game content (for example, websites and phone numbers). We typically request phone numbers be changed to 555: https://en.wikipedia.org/wiki/555_(telephone_number)

112.3 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.

112.4 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 map author(s).

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.

Craftable Items: Ten to twenty items of uncommon rarity. Unlike mystery box contents, it is far more appropriate for craftable items to have matching sets and simple recolors.

112.5 FAQ

Q. Do curated maps receive updates?

We may suggest some fixes or other adjustments necessary for the map to meet our expectations. However, additional content is largely left to the discretion of the map author(s).

Q. What game modes can maps be accepted for?

Maps designed for officially-supported modes (Survival and Arena) are the most likely to be accepted. We may occasionally consider some maps designed for popular custom game modes.

Q. Can multiple maps be released together?

If/when considering maps to release together, we'd like to ensure it makes sense to do so. For example, an arena map and a PvE-focused survival map. This is something we're still exploring.

CHAPTER

THREE

EDITOR ASSET REDIRECTORS

Editor Asset Redirectors allow for quickly replacing objects, resources, materials, or foliage assets in bulk.

Redirects apply while loading a map in the level editor. Any changes are then kept when saving the map.

Create a file named "**EditorAssetRedirectors.txt**" in the Unturned folder.

- Empty lines are ignored.
- Lines starting with `//` or `#` are ignored.
- Each redirect should include two *GUIDs* separated by an arrow `->`.

For example:

```
// Replace Boulder_00 with Boulder_01
6125b4de591b44359237f6d7191dd919 -> ee402fc9debe4f03bffb31a49eb04fb7

// Replace Maple_0 with Maple_3
63cb368c94b14000aabc5325b048cfa3 -> 011d1369cd56497488827b44509b0b4b
```

CHAPTER

FOUR

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

FIVE

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)

115.1 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.

115.2 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.

The maximum included texture size defaults to 128x128. Including bigger textures risks exceeding the maximum texture size, but can be adjusted with this option in the level config:

```
"Batching_Max_Texture_Size": 256
```

115.3 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.

115.4 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.

115.5 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.

115.6 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`.

115.7 Previewing renderers using atlas

You can visualize which renderers have been included in the texture atlas by loading singleplayer with the `-PreviewLevelBatchingTextureAtlas` launch option:

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.

115.8 Previewing renderers using static batching

You can visualize which mesh renderers are included in static batching by loading singleplayer with the `-PreviewLevelBatchingUniqueMaterials` launch option.

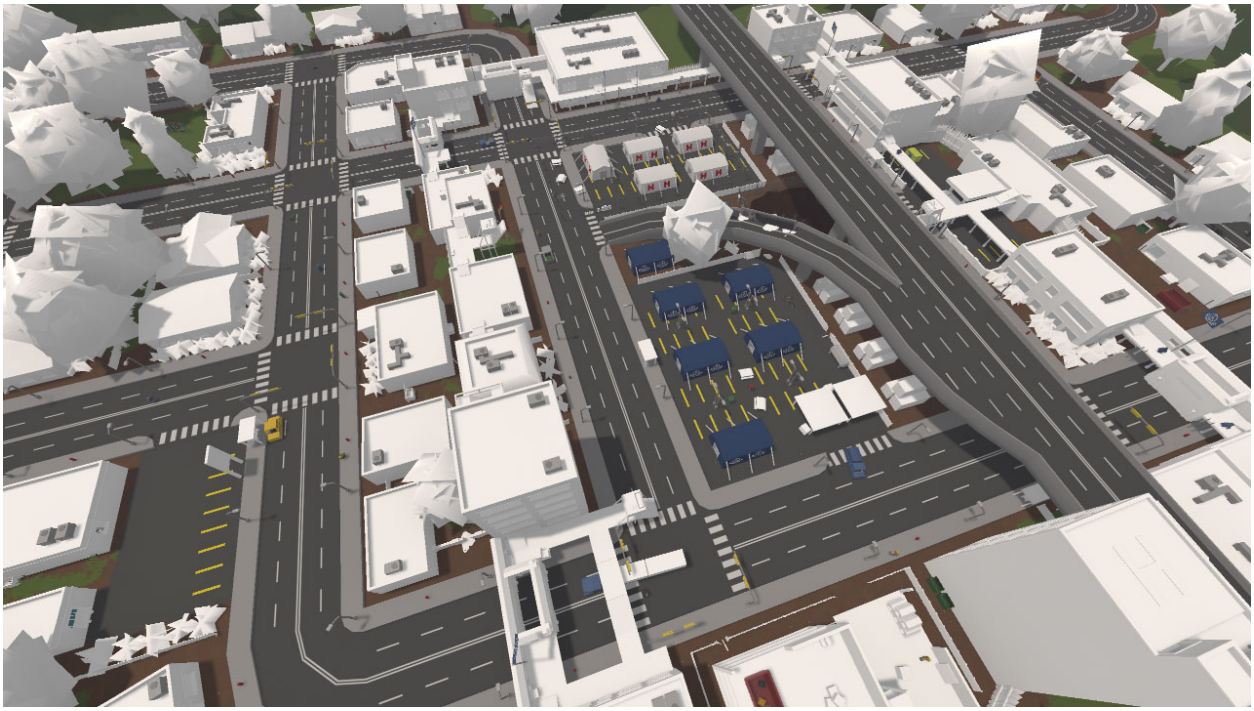


Fig. 1: Berlin with texture atlas preview enabled.

Each unique material is assigned a random hue. The brightness of the color decreases in order of uniqueness. i.e., The brightest color is used by the most renderers, and the darkest color is used by only a few.

CHAPTER

SIX

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.

116.1 Main Menus

Creators *string[]*: Names in credits.

Collaborators *string[]*: Names in credits.

Thanks *string[]*: Names in credits.

CustomCredits: Maps header internal title(s) to a list of names. The display title is formatted according to the level's localization file. For example:

```
"CustomCredits":
{
        "Music":
        [
                "musician67",
                "SoundDesigner (these names aren't localized)"
        ],
        "Art":
        [
                "MyFavouriteArtist"
        ]
}
```

> **Note**
>
> In that example, the keys `Music` and `Art` would be translated from the level's localization file.

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.

RequiredWorkshopFileIds *ulong[]*: Dependency workshop file IDs. If these file(s) are not loaded then singleplayer and editor menus display a "Missing Dependencies" message and prevent entering the map.

```
"RequiredWorkshopFileIds":
[
        123456789,
        123456789
]
```

116.2 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,
                "Amount": 1
        },
        {
                "Table_ID": 28008,
                "Amount": 1
        }
]
```

116.3 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
```

(continues on next page)

(continued from previous page)

```
        }
]
```

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
}
```

Additionally, overrides can be applied per-difficulty with the `EasyDifficulty_Config_Overrides`, `NormalDifficulty_Config_Overrides`, and `HardDifficulty_Config_Overrides` properties.

Allow_Underwater_Features *bool*: Should legacy details and navigation bounds be restricted underwater?

Enable_Static_Volumes *bool*: Enables performance optimizations of volume overlap checks. This setting is intended for levels with high numbers of volumes *not* added through Unity prefabs. It should not be enabled if volumes move, resize, or otherwise change at runtime. Defaults to false.

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 landscape tiles. Defaults to true.

Use_Legacy_Water *bool*: Should global water plane be enabled? Alternative is to use water volumes. 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_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.

Enable_Clutter_Option *bool*: If true, "Load Clutter" graphics option is supported on this map. Defaults to false. This is opt-in so that the map creator(s) can decide whether the removed details are an acceptable compromise.

Batching_Version *int*: Please refer to *Level Batching*.

Batching_Max_Texture_Size *int*: Overrides the maximum texture size included in the *Level Batching* atlas. Please keep in mind that including bigger textures risks exceeding the maximum texture size.

116.4 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*

116.5 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_Atmosphere *bool*: This property disabled stars in the skybox, but was deprecated due to changes to the skybox implementation.

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

SEVEN

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.

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.

117.1 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.

117.2 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.

117.3 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.

117.4 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

EIGHT

LEVEL VOLUMES

Volumes are box or sphere-shaped areas placed within the level editor. Each changes gameplay, visuals, audio, or state when a player is within its bounds.

Shape *enum* (`Box`, `Sphere`): Shape of the volume. This setting only appears on volumes that support both shapes.

Falloff *float*: Distance inward from the edge before the volume reaches full intensity. Both ambiance and oxygen volumes support this setting.

118.1 Gameplay

118.1.1 Deadzone

Gives players radiation via lowering their virus / infection stats. Can be counteracted when they wear protective clothing.

Deadzone Type *enum* (`Default Radiation`, `Full Suit Radiation`): Protection level that players need. Default radition will require a gasmask with filters. Full suit radiation requires a gasmask an full body suit.

Damage per Second (Unprotected) *float*: Damage dealt each second to players without protection. Defaults to `0`.

Damage per Second (Protected) *float*: Damage dealt each second to players who have protection. Defaults to `0`.

Radiation per Second *float*: Radiation (virus/infection) damage dealt each second to players without protection.

Mask Filter Degradation per Second *float*: How fast an equipped gasm ask filter loses quality. Defaults to `0.4`.

118.1.2 Horde Purchase

Area where players can buy an item in Horde mode. This level mode has been deprecated.

Item ID *uint16*: Legacy ID of the item to sell.

Cost *uint32*: Price of the item.

118.1.3 Kill

Instantly kills entities that enter the volume.

Kill Players *bool*: Kill players. Defaults to `true`.

Kill Zombies *bool*: Kill zombies. Defaults to `true`.

Kill Animals *bool*: Kill animals. Defaults to `true`.

Kill Vehicles *bool*: Destroy vehicles. Defaults to `false`.

Death Cause *enum*: Death cause reported for killed players, for example in the death message. Defaults to `BURNING`.

118.1.4 No Structures

Stops structures from being placed inside the volume.

118.1.5 Oxygen

Overrides whether players can breathe inside the volume.

Is Breathable *bool*: If true, players can breathe within the volume. If false, oxygen is depleted. Defaults to `true`.

118.1.6 Safezone

Protected zone where hostile actions are restricted.

No Weapons *bool*: Players inside cannot use weapon items. Defaults to `true`.

No Buildables *bool*: Players cannot place barricades or structures inside. Defaults to `true`.

No Incoming Damage *bool*: Players inside cannot take damage, unless the source bypasses safezones. Defaults to `true`.

118.1.7 Teleporter Entrance

Teleports players who enter the volume to a Teleporter Exit with a matching ID.

Pair ID *string*: ID of the Teleporter Exit to send players to.

118.1.8 Teleporter Exit

Exit for Teleporter Entrance volumes.

ID *string*: ID that Teleporter Entrances use to find this exit.

118.1.9 Underground Whitelist

Designates areas where players are allowed to be when below the terrain surface. This is only applied when enabled in the *Level Config*. When outside of a whitelist volume, the player will be teleported to the surface.

118.1.10 Arena Mode Compactor

Sphere only. Sets the starting circle in Arena mode. When the map has more than one of these, one is picked at random each round.

118.2 NPCs

118.2.1 NPC Overlap

Area that triggers the `Volume_Overlap` *NPC condition*.

ID *string*: ID that `Condition_#_VolumeID` refers to.

118.2.2 NPC Reward

Grants a *Rewards List Asset* to players who enter the volume, if the asset's conditions are met.

Asset GUID *GUID*: GUID of the rewards list asset.

Affect Vehicle Passengers *bool*: If true, pasengers of vehicles are granted the rewards list. Defaults to `false`.

118.3 Environment

118.3.1 Ambiance

Overrides ambient audio, fog, lighting, and weather within the volume.

Effect ID *GUID* or *uint16*: *Effect Asset* played as ambient audio while inside.

No Water *bool*: Hides the water surface while inside.

No Lighting *bool*: Turns off level lighting while inside. Useful for dark interiors.

Weather Mask *Bitmask*: Weather only blends in while inside if its `Volume_Mask` shares a bit with this mask.

Override Fog *enum* (`None`, `Constant`, `PerTimeOfDay`): Whether the volume overrides fog.

Fog Time *enum*: Picks which time of day the fog color and intensity fields edit when using `PerTimeOfDay`.

Fog Color *color*: Color of the overridden fog.

Fog Intensity `Fog_Intensity` *float*: Strength of the overridden fog.

Override Atmospheric Fog *bool*: Also override atmospheric (sky) fog.

Use Falloff *bool*: If true, blends by distance using **Falloff**. If false, blends over time using the fade durations below.

Priority *int*: Volumes with higher priority override volumes with lower priority where they overlap.

Audio Fade-In / Audio Fade-Out *float*: Seconds to fade audio in and out when **Use Falloff** is off. Both default to `2`.

Fog Fade-In / Fog Fade-Out *float*: Seconds to fade fog in and out when **Use Falloff** is off. Default to `20` and `8`, respectively.

Lighting Fade-In / Lighting Fade-Out *float*: Seconds to fade lighting in and out when **Use Falloff** is off. Both default to `4`.

118.3.2 Effect

Box only. Spawns an *Effect Asset* scaled to fill the volume.

Effect ID *GUID* or *uint16*: Effect to spawn. GUID usage is recommended.

Emission Rate *float*: Multiplier for the particle system's max particles and emission rate. Defaults to `1`.

Audio Range *float*: Sets the max distance of the effect's audio source. Defaults to `1`.

118.3.3 Foliage

Controls where foliage can be placed / baked.

Mode *enum* (`Additive`, `Subtractive`): Additive volumes allow foliage to be baked inside. Subtractive volumes stop foliage from baking inside.

Instanced Meshes *bool*: Affects instanced mesh foliage such as grass. Defaults to `true`.

Resources *bool*: Affects resource foliage such as trees and rocks. Defaults to `true`.

Objects *bool*: Affects object foilage. Defaults to `true`.

118.3.4 Water

A body of water that players can interact with.

Surface Visible *bool*: Shows the water surface. Defaults to `true`.

Reflection Visible *bool*: Enables planar reflections on the surface. Defaults to `false`.

Sea Level *bool*: Marks this as the level's sea level volume.

Refill Type *enum* (`Clean`, `Salty`, `Dirty`): Type of water given when refilling canteens from this volume. Defaults to `Salty`.

Fish *GUID*: Spawn table for fish caught in this volume. Requires `Supports_Fishing_Volumes` in the *Level Asset*.

Fishing Minimum Depth Override *float*: If set, the water at the bobber must be deeper than this to catch fish.

118.4 Miscellaneous

118.4.1 Cartography (GPS/Chart)

Box only. Sets the area captured for the GPS and Chart map images.

118.4.2 Manual Object Culling

Hides the objects inside it when the player is farther away than the cull distance. See *Manual Object Culling*.

Cull Distance *float*: Distance from the volume's center which its objects are hidden. Defaults to `64`.

Include Large Objects *bool*: Also culls objects of type `Large`, such as shipping containers. Defaults to `false`.

118.4.3 Navmesh Clip

Solid volume that blocks zombie navigation.

118.4.4 Player Clip

Invisible solid that blocks player movement.

Block Player *bool*: If false, the volume no longer blocks players but will prevent buildables from being placed within it. Defaults to `true`.

CHAPTER

NINE

SETTING UP A SERVER

Players can host their own multiplayer servers through the **Unturned Dedicated Server** tool. (This can be abbreviated as "**U3DS**".)

This tool is provided alongside Unturned but must be installed separately. It can be installed and ran from your Steam Library, or instead through *SteamCMD* for a much more advanced setup.

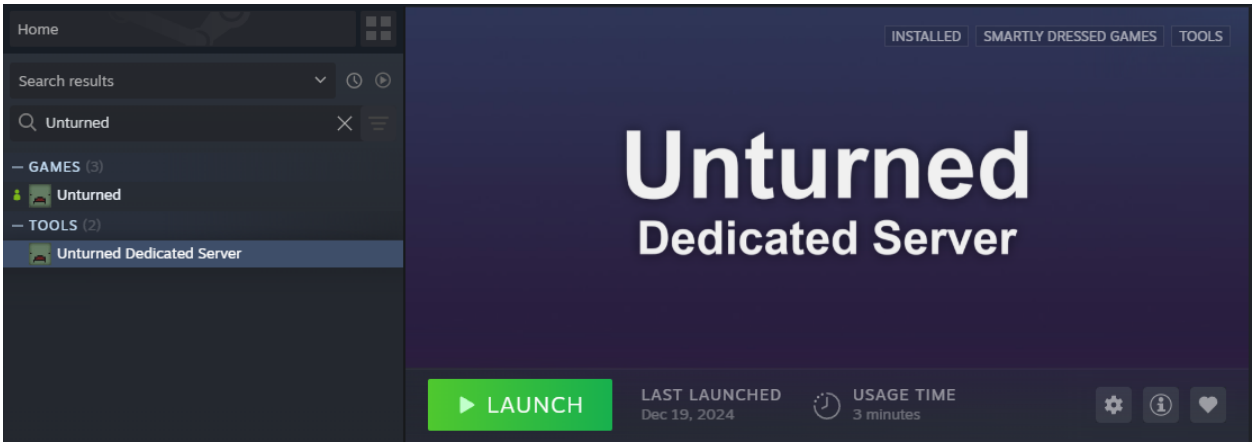


Fig. 1: Find the **Unturned Dedicated Server** app in your Steam Library.

Servers can be hosted on Windows and Linux operating systems. There is no support for hosting on macOS devices.

119.1 Simple Setup

This section provides a quick guide to creating your own server—perfect for hosting a *private* multiplayer world for just you and a few friends.

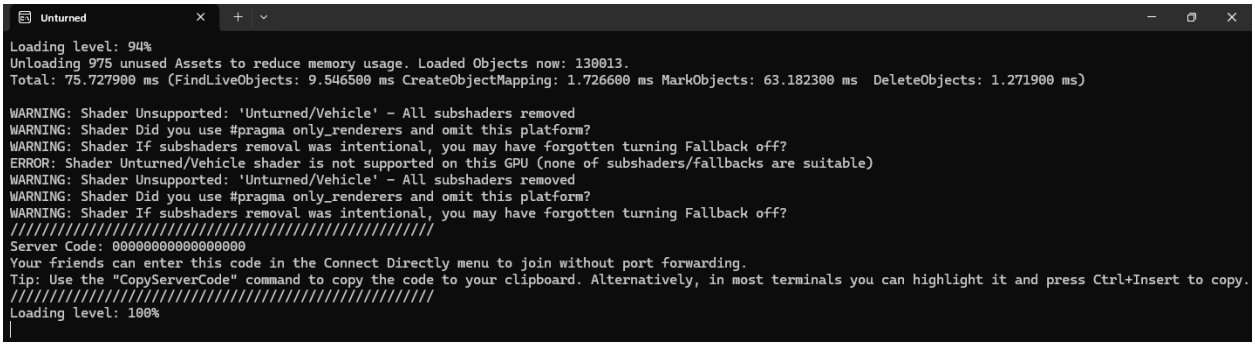
Port forwarding and other optional – *but useful* – features are not covered in this section. Consider reading more advanced setup instructions once your server is functional.

119.1.1 1. Launch the Unturned Dedicated Server app

Launch the **Unturned Dedicated Server** app from your Steam Library.

The Server Console will open in a new window. It may take a few minutes for your server to boot up while it generates any necessary files and loads your world data.

You should see "Loading level: 100%" when it finishes booting, along with a *Server Code*.



119.1.2 2. Invite your Friends

You can invite players to your server through your Steam Friends List.

Players can also join your server by using the 17-digit **Server Code** created when you opened your server. This randomly-generated code will change each time you launch the server. It can found be clicking the "Copy Server Code" button while in your server, or by copying it from your Server Console.

> **Hint**
>
> Players—including yourself—can join your server by inputting the server's **Server Code** in the **Connect Directly** menu. For friends, it's often quickest to just invite them through Steam!

By default, your server is not visible to others from the Multiplayer menu unless they are on the same LAN connection. Without following more advanced setup instructions, this *functionally* means that players can only join via your Steam Friends List or if you share the Server Code with them.

119.2 Save and Shutdown

There is no built-in autosave feature. You must use the `Save` command to manually save your server while its still running, or the `Shutdown` command to safely save and close your server.

Savedata and configuration files can be found in the `...\U3DS\Servers\` directory. You will see a folder for each server you have created.

119.3 Configure Server Settings

Configuration files can be found inside your server's server folder. Most settings cannot be configured while the server is open.

> **Hint**
>
> If you created a server using the *Simple Setup* guide, you can find your server's settings from the `...\U3DS\Servers\Default\` directory.

Commonly-used configuration files are:

- **\Server\Commands.dat** – Configure basic server settings, such as: Map, Password, and Max Players.
- **\Config.txt** – Adjust difficulty settings and advanced server settings, such as: increasing Item Spawn Chance or assigning a Game Server Login Token. (see *Server Configuration*)

- **\WorkshopDownloadConfig.json** – Download mods from the Steam Workshop.

119.3.1 Commands

Commands—sometimes referred to as console commands or cheat commands—range from tweaking server settings to spawning items. Most commands are set in the **Commands.dat** file, with each command separated by a newline.

Learn more about available commands on the Unturned Wiki. Below is an example setup that includes many commonly-used commands:

```
// [[ SERVER CONFIGURATION ]]
// Lines starting with "//" are comments - which aren't read when launching your server.
// Read more about commands on the game's wiki: https://unturned.wiki.gg/Commands

// Name of your server in the server list.
Name My Unturned Server

// Specify which map to load, by name.
// Official maps include PEI, Washington, Yukon, Russia, and Germany.
// Use the WorkshopDownloadConfig.json file to download Workshop maps.
Map PEI

// Maximum number of players that can be on the server.
MaxPlayers 24

// Assign a player's steamID64 as the "Owner" to give them administrator privileges, such␣
↪as invoking cheat commands while in-game.
// Owner YourSteamID

// Enable "cheat commands" for administrators, such as spawning items or vehicles.
Cheats

// Require a password to join the server.
// Password ExamplePassword1234

// Disable player-versus-player combat.
// PvE

// Limit players to a certain camera perspective, such as only first-person. By default,␣
↪players can use both first-person and third-person perspectives.
// Perspective First
// Perspective Third
Perspective Both
// Perspective Vehicle

// Which port should be used when port forwarding.
// Port 27015
```

Runtime

Some commands only work during runtime. For example, the `Save` command only works while the server is running.

You can use runtime commands in two ways:

a. Type them into the **Server Console**.

b. From the **in-game chat** (only if you are an administrator) while prefixing them with `@` or `/`.

> **Tip**
>
> Add your Steam ID as an administrator (a) via the `Owner` command, (b) by using the `Admin` command during runtime, or (c) by including it in the **Adminlist.dat** file.

Use the `Help` command to either view a list of *all* commands, or describe a specific command.

Some runtime commands require `Cheats` to be enabled. For example, `Give` is a cheat command used to spawn items.

Learn about more commands on the Unturned Wiki.

119.3.2 Difficulty Settings

Configure your server's game rules, public server listing, and many other options from the *Server Configuration* file.

By default, there are separate config files for each difficulty preset – Easy, Normal, and Hard. By default, servers use the Normal difficulty file.

> **Hint**
>
> You can use the `Difficulty` command to use the settings configured in either the Easy or Hard files.

119.3.3 Steam Workshop Mods

Use the **WorkshopDownloadConfig.json** file to install mods from the Steam Workshop for use on your server. Players attempting to join your server will automatically download any mods configured in this file!

1. Open the **WorkshopDownloadConfig.json** file.
2. Find a mod on the Steam Workshop and copy the **File ID** in the URL. For example, the Hawaii map.

```
URL: https://steamcommunity.com/sharedfiles/filedetails/?id=1753134636
File ID: 1753134636
```

3. Add the File ID to the **File_IDs** list. Example:

```
"File_IDs":
[
        1753134636
],
```

> **Tip**
>
> You can add multiple mods by separating each File ID with a comma. Example:

```
"File_IDs":
[
        1753134636,
        1702240229
],
```

4. During server startup, any specified mods (and their dependencies) will be installed and updated. Players will automatically begin downloading any mods while connecting to the server.

Curated maps—and a few official arena maps—are only accessible via the Steam Workshop. Like other mods, they need to be downloaded by the server in order to be used:

Workshop Map	File ID
A6 Polaris	2898548949
Athens Arena	1454125991
Arid	2683620106
Belgium	1727125581
Buak	3000549606
Bunker Arena	1257784170
California	1905768396
California2	3707778928
Canyon Arena	1850209768
Carpat	1497352180
Cyprus Arena	1647991167
Cyprus Survival	1647986053
Dango	1850228333
Easter Island	1983200271
Escalation	3251926587
Elver	2136497468
France	1975500516
Greece	1702240229
Hawaii	1753134636
Hawaii Assets	1753131903
Ireland	1411633953
Kuwait	2483365750
PEI Arena	2396897717
Rio de Janeiro	3416057692
Rio de Janeiro (Original)	1821848824
Washington Arena	2404652624

119.4 Switching to an Internet Server

Servers are hosted as a **LAN server** by default. This means players can only join when invited through your Steam friends list, or by using your randomly-generated Server Code. LAN servers are *not* visible from the Internet server list. There are a couple options for switching from a LAN server to an **Internet server**.

Attention

Internet servers should follow our *Server Hosting Rules*.

Get started by adding a *Game Server Login Token* (GSLT). This login token is necessary when hosting an Internet server. Configuring it also means your Server Code will no longer change each time you launch your server.

For an Internet server, you must configure either *Fake IP* or *Port Forwarding*. Configuring both is **not necessary**. Fake IP is easier to enable, and it should be adequate for most servers.

119.5 Related Topics

Some servers may benefit from a more advanced setup than the one provided in our *Simple Setup* guide. Chiefly:

- *Use SteamCMD* to host multiple servers at once.
- Install plugins through frameworks like *Rocket (LDM)* or *OpenMod*.

USING STEAMCMD (ADVANCED SETUP)

Some servers may benefit from a more advanced setup than the one provided in our *Simple Setup* guide. This page assumes you have read the full "*Setting up a Server*" documentation.

When hosting multiple servers at the same time, we recommend using SteamCMD to install the **Unturned Dedicated Server** tool rather than installing it from your Steam Library. SteamCMD is a command-line version of the Steam Client.

It is not possible to run multiple servers at once—from the same computer—*without* using SteamCMD.

Servers can be hosted on Windows and Linux operating systems. There is no support for hosting on macOS devices.

120.1 Download SteamCMD on Windows

Installation on Windows operating systems (such as Windows 10 or Windows 11) is very straightforward.

1. Download SteamCMD for Windows.
2. Extract the contents of the zip to somewhere you can find it again.
3. Run `steamcmd.exe`.

120.2 Download SteamCMD on Linux

Installation on Linux varies by distribution and your admin preferences. We recommend referring to Valve's Linux Documentation. Once downloaded, run the `steamcmd.sh` script.

120.3 Install the Unturned Dedicated Server with SteamCMD

1. Login to Steam anonymously.

```
login anonymous
```

2. Download the Unturned Dedicated Server application.

```
app_update 1110390
```

Tip

The same command can be used to update an existing server installation. To verify the integrity of the server files and repair any missing or corrupted files, append the `validate` flag:

```
app_update 1110390 validate
```

3. Close SteamCMD once the download finishes.

```
quit
```

4. In your SteamCMD folder, the server files can now be found within the default app install directory. For the Unturned Dedicated Server, this is `.../steamapps/common/U3DS/`.

If you need additional information, consider reading the SteamCMD wiki page's "Downloading an App" section.

120.4 Launch a Server on Windows

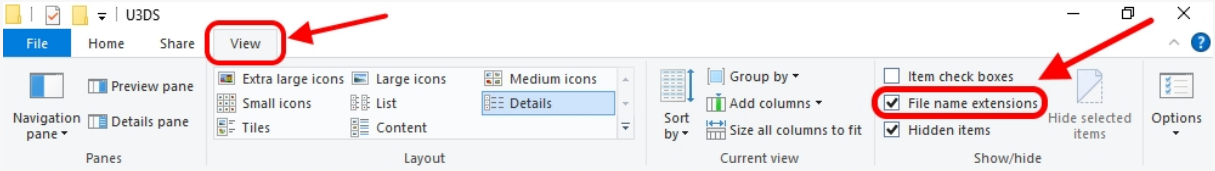
Although old, this video guide remains fairly accurate and can supplement the reading below: https://www.youtube.com/watch?v=8axVrnSLlx4.

1. Navigate to your server files. When using the default app install directory, these should be found at `...\steamcmd\steamapps\common\U3DS`.
2. Create a new text file within the U3DS directory. This file can be named anything – but for the purposes of this guide, we will call it "**MyServer**".
3. Change the file type from a text file (.txt) to a batch script file (.bat). To do this, simply edit the file's name and replace the `.txt` file extension with `.bat`. For example, "**MyServer.bat**".

Warning

If the file name does not display the `.txt` file extension – e.g., you see "MyServer" rather than "MyServer.txt" – you need to enable the viewing of "File name extensions" in Windows.

Click the "View" button in File Explorer and enable the "File name extensions" option. Once enabled, you can see and edit file extensions as part of file names.



4. Open **MyServer.bat** in any text editor, such as Notepad.
5. Copy-and-paste one of the following scripts into the file, depending on the type of server you are hosting:

   For an **Internet** server:

   ```
   start "" "%~dp0ServerHelper.bat" +InternetServer/MyServer
   ```

   For a **LAN** server:

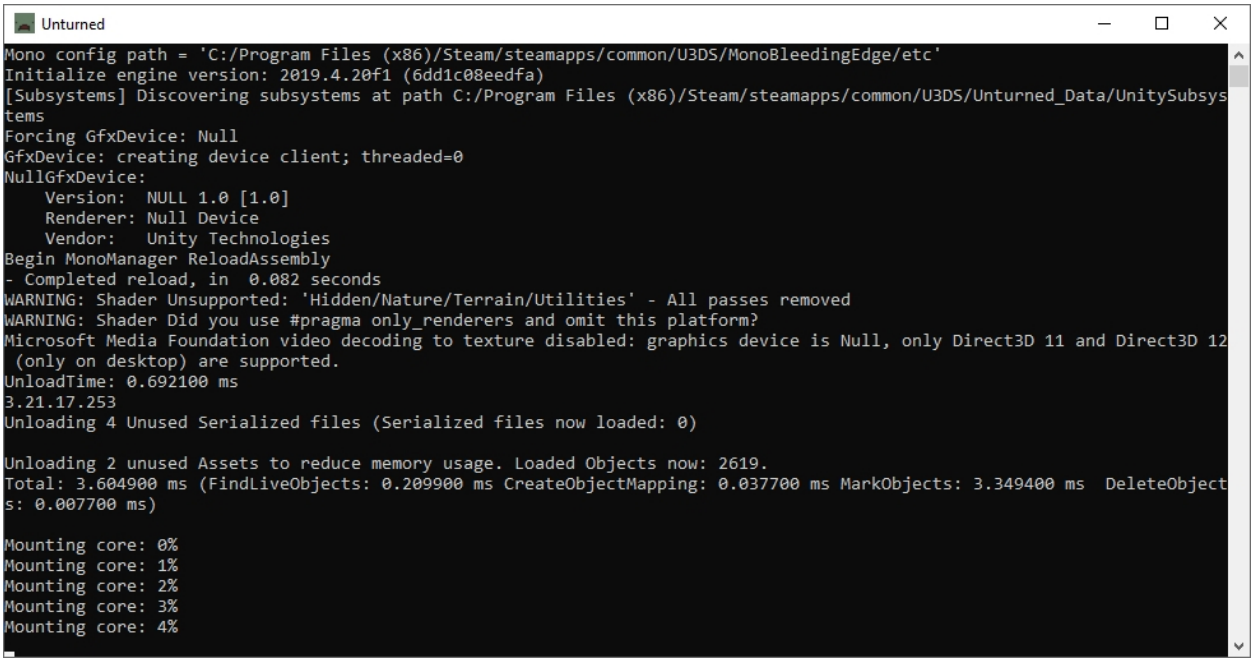
   ```
   start "" "%~dp0ServerHelper.bat" +LanServer/MyServer
   ```

Tip

In the provided scripts, "MyServer" is being used as the ServerID. Your server's savedata and configuration files will be found in a folder named after your chosen ServerID.

ExampleServer.bat is an example of a LAN server. Opening it in a text editor will provide additional information that may be helpful should you need it.

6. Save your changes to the file and close it.
7. Double-click **MyServer.bat** to launch your server. A command-line interface should appear. Since this is the first time you have launched your server, it will need to generate your server's configuration files.



8. You should see "Loading level: 100%" when the server has finished loading (and generated all necessary files). Use the `Shutdown` command in the command-line interface to safely save and close your server for now.
9. Your server's configuration files can be found in a newly-created folder (named after its ServerID) within the `...\U3DS\Servers` directory. This is where your savedata and configuration files are stored.

> **Tip**
>
> Refer to *Setting up a Server – Configure Server Settings* for guidance on customizing your server.

10. When hosting an Internet server, refer to *Setting up a Server – Switching to an Internet Server* for guidance on adding a *Game Server Login Token* and using either *Fake IP* or *Port Forwarding*.

Once configured, you can open your server by running the **MyServer.bat** script!

120.5 Launch a Server on Linux

1. Navigate to your server files. When using the default app install directory, these should be found at `.../steamcmd/steamapps/common/U3DS`.
2. Copy-and-paste one of the following commands into your Linux terminal, depending on the type of server you are hosting:

   For an **Internet** server:

   ```
   ./ServerHelper.sh +InternetServer/MyServer
   ```

   For a **LAN** server:

```
./ServerHelper.sh +LanServer/MyServer
```

> **Tip**
>
> In the provided scripts, "MyServer" is being used as the ServerID. Your server's savedata and configuration files will be found in a folder named after your chosen ServerID.
>
> **ExampleServer.sh** is an example of a LAN server. Opening it in a text editor will provide additional information that may be helpful should you need it.

3. You should see "Loading level: 100%" when the server has finished loading (and generated all necessary files). Use the `Shutdown` command in the command-line interface to safely save and close your server for now.
4. Your server's configuration files can be found in a newly-created folder (named after its ServerID) within the `...\U3DS\Servers` directory. This is where your savedata and configuration files are stored.

> **Tip**
>
> Refer to *Setting up a Server – Configure Server Settings* for guidance on customizing your server.

5. When hosting an Internet server, refer to *Setting up a Server – Switching to an Internet Server* for guidance on adding a *Game Server Login Token* and using either *Fake IP* or *Port Forwarding*.

Once configured, you can open your server by running the **MyServer.sh** script!

120.6 Related Topics

Some additional topics are covered elsewhere. Of particular interest:

- Install plugins through frameworks like *Rocket (LDM)* or *OpenMod*.
- Automate server maintenance with *Server Auto Restart* features.
- Configure a *Bookmark Host* so players can "Bookmark" your server.
- *Stay notified* when the Unturned Dedicated Server app has an update.

CHAPTER

ONE

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

121.1 Recent changes

2025-06-12: Clarified what counts as a "vanilla premium content" and what servers are allowed to do instead. (Previously, this was only detailed in the "Recent changes" section.)

2024-06-03: Scaled back the degree to which the server list is moderated. The previous level was untenable especially when considering anyone can freely create an unlimited number of servers, and the moderation system is duct-taped on top of the otherwise unmoderated Steam server list. For more information, please refer to our support article about moderation and server curation as well as some behind-the-scenes details.

121.2 Monetization Types

Servers are not allowed to sell or otherwise offer access to **vanilla premium content**. This includes the Gold Upgrade benefits in addition to any vanilla cosmetics and/or skins. For example, any item purchasable from the Stockpile or the Steam Community Market for Unturned is considered a "vanilla premium content".

When offering cosmetic perks to players as a microtransaction, the server should own (or have licensed) the rights to that content. For example, you could create your own custom "cosmetics" and offer those to players instead.

121.3 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.

121.3.1 Unspecified

The "Monetization" field in each *Server Configuration* file defaults to `Unspecified`. If you are unsure what to configure your server's monetization type as, then you can leave it unspecified.

121.3.2 None

Servers that are entirely unmonetized, or only offer a donation option, can use the `None` value.

121.3.3 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.

121.3.4 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.

121.4 Online Conduct

Repeat violations of Steam's Online Conduct rules by a multiplayer server may result in moderation. For example, servers that exist to promote the sale of cheats are not allowed.

If you encounter inappropriate behavior by *other players* while playing on a server, we encourage you to report it to the server's owner first so they are aware of the issue.

121.5 Roleplaying Current Events

Simulating gameplay of current, real-world tragedies is **not** allowed and may result in reduced visibility or other moderation on the server. For example, roleplaying ongoing conflicts (such as the the Russo-Ukrainian War, or Israeli-Palestinian conflict) or natural disasters.

121.6 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.

121.7 Workshop File Copyright Infringement

Mod authors can submit a notice of copyright infringement here: https://steamcommunity.com/dmca/create/. These are reviewed directly by Steam's copyright agent.

121.8 FAQ (Frequently Asked Questions)

121.8.1 Can I report servers for admin abuse or pay-to-win features?

This is not reportable. Unfortunately, the best option is to play on a different server, or to *host your own multiplayer server* for you and your friends. Moderating individual community servers is not tenable at our current small scale with current tools, but may be revisited if/when it becomes feasible.

121.8.2 How can my server mitigate Denial of Service (DoS) attacks?

If your server is receiving DoS attacks, the *Fake IP* feature is a great way to protect against this behavior. It routes traffic through Steam's relay network, shielding your server's IP, and potentially reducing network latency for players.

121.8.3 How can I prevent people from re-uploading my Workshop files?

Update your mod to include an *Asset Bundle Custom Data* and configure it to use the original upload's Workshop file ID. This adds a verification check to when a server attempts to download the mod – preventing it from being downloadable if the mod's Workshop file ID doesn't match the configured value. As a last resort, if your content has been reuploaded to Steam by another user without your permission, you may consider submitting a notice of copyright infringement. Notices of copyright infringement are reviewed by Valve's copyright agent, and should only be submitted if you understand the legal information on their submission page.

CHAPTER

TWO

BOOKMARK HOST

Configuring the **Bookmark Host** property along with a *Login Token* enables the bookmark button, allowing players to find your server even after an IP or port change.

By default the game supports Steam's built-in Favorites and History server lists. These lists remember your server by the combination of IPv4 address and port, so if your address or port change the link is broken. Steam mitigates this if you have a *GSLT* assigned by automatically updating the saved address, but this doesn't happen in realtime. These lists also aren't compatible with the newer *Fake IP* feature.

Considering the trend toward Fake IP, a replacement was necessary, not to mention a popular request from server hosts. Thankfully, with GSLTs in widespread use now and providing a stable persistent ID, the client can save custom per-server bookmark data. The intention is to make bookmarks a better alternative to the legacy favorites/history lists, including proper support for IPv6 at some point. At the time of writing (2024-05-24) the client doesn't save history using the bookmark host property yet, but it should be added once more hosts opt-in.

To enable bookmarking, set `BookmarkHost` in your *Server Configuration* to one of these formats:

1. A DNS entry with an `A` record pointing to your server's public IP. For example, if you own the "example.com" domain you could add an A record "myunturnedserver" pointing at your game server IP and set `BookmarkHost` to "myunturnedserver.example.com". In this case the client will save your server's current port number, so the IP address can change but the port can't. If your server doesn't have a static public IP you could use dynamic DNS to update the DNS record periodically. Note that this option is inapplicable for servers using Fake IP because both the IP and port are randomly assigned after each restart.
2. The URL of a custom web API to return the address, and optionally the port. Clients perform a GET request if `BookmarkHost` begins with "http://" or "https://". The response should be plain text. Examples of valid responses include:
   - 127.0.0.1
   - 127.0.0.1:27015
   - myunturnedserver.example.com
   - myunturnedserver.example.com:27015

   The hosts of Pandahut have generously shared an example implementation consisting of a server plugin which updates the latest server address on a custom backend, and a web request handler to serve the latest address to clients. Note that we don't necessarily recommend using this program as-is, it's just an example: BookmarkHostPlugin by PandahutMushy

CHAPTER

THREE

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

FOUR

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

FIVE

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

SIX

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, set `Use_FakeIP` to `true` in your *Server Configuration*.

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.

126.1 Technical Details

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.

126.2 Bookmarking

Unfortunately, Fake IP is not compatible with Steam's built-in "Favorites" and "History" server lists. As an optional workaround, these servers can utilize our *Bookmarking* feature instead.

CHAPTER

SEVEN

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.)

127.1 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.

127.2 Unturned Configuration

The GSLT can be set in one of two places depending on your preference:

- With the `Login_Token` property in each *Server Configuration* 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.

127.3 Automating GSLTs

Valve provides an `IGameServersService` web API for managing GSLTs. Consult their documentation here: https://partner.steamgames.com/doc/webapi/IGameServersService

CHAPTER

EIGHT

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.

128.1 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.

128.2 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.

128.3 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).

128.4 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

NINE

MODULES

Modules are assembly files that the game loads at startup. Each module has a `.module` configuration file, and the game invokes methods in any class in its assemblies that implement the `SDG.Framework.Modules.IModuleNexus` interface. Plugin frameworks such as *OpenMod* and *Rocket* utilize this system. While traditionally used only by the server, modules can also be installed on player clients.

Warning

Players and server hosts should maintain caution when installing modules from untrusted sources. Distributions on the workshop are never officially verified.

129.1 Installing a Module

Place the module's folder inside of the `Modules` directory of your Unturned installation, then restart the game or server. The game searches this folder and its subfolder for `.module` files. You may keep each module within its own folder.

```
Unturned/
└── Modules/
    └── MyModule/
        ├── MyModule.module
        └── MyModule.dll
```

The `-ModulesPath` *launch option* can be used to define the search to another directory instead.

129.2 Configuration File

`.module` files are formatted in JSON. Field names are case-sensitive.

```
{
        "IsEnabled": true,
        "Name": "ExampleModule",
        "Version": "1.0.0.0",
        "Dependencies":
        [
                {
                        "Name": "OtherModule",
                        "Version": "1.0.0.0"
                }
        ],
```

(continues on next page)

(continued from previous page)

```
        "Assemblies":
        [
                {
                        "Path": "/ExampleModule.dll",
                        "Role": "Both_Optional"
                }
        ]
}
```

IsEnabled *bool*: If false, the module will not be loaded. Defaults to `true`.

Name *string*: Name of module.

Version *string*: Version of the module.

Dependencies *array of objects*: Modules that must be loaded before this module. If the dependency is missing or its version is lower than specified, the module is not loaded.

> **Name** *string*: Name of the required module.
>
> **Version** *string*: Minimum version of the required modules.

Assemblies *array of objects*: Assemblies that belong to the module.

> **Path** *string*: Path to the `.dll` file. This is relative to the folder containing the `.module` file.
>
> **Role** *EModuleRole*: Specifies if the module should be loaded on the server, client, or both. Options include: `Client`, `Server`, `Both_Optional`, and `Both_Required`.
>
> **Load_As_Byte_Array** *bool*: If true, the entire assembly is read into memory before being loaded rather than loaded directly from the file. Prevents the file from being locked during runtime. Defaults to `false`.

CHAPTER

NETMETHODS AND NETIDS

130.1 NetMethods

NetMethods are remote procedure (RPCs) used to communicate between the server and clients. A NetMethod can come in the form of either **static**, calling a static method, or **instance**, calling a method on specific object identified by a **NetId**. These have various use cases in *modules* ran on both the server and client.

In most typical use cases, a NetMethod is a method with the `SteamCall` attribute. These are then paired with a static handle used to invoke it remotely. The vanilla naming convention on these is prefixing this method with `Receive` and its handle with `Send`.

The following are the four handle types, each have different call directions and designation of either static or instance:

Handle	Direction	Receive Method
`ClientStaticMethod`	Server to client	Static method. No NetId is sent.
`ClientInstanceMethod`	Server to client	Instance method on the object paired with the given NetId.
`ServerStaticMethod`	Client to server	Static method. No NetId is sent.
`ServerInstanceMethod`	Client to server	Instance method on the object paired with the given NetId.

These handles also have generic versions for methods that have parameters. For example, `ClientStaticMethod<string>` invokes a static client method with one `string` parameter passed in.

There also exists read and write methods for the networking data of each NetMethod. These are *Generated Methods* and will have to be generated or manually created.

> **Note**
>
> The server and client must have identical method lists. Creating these in an environment of various modules / modifications will require synchronization. Therefore, serverside plugins aren't capable of adding new RPCs.

130.1.1 Static Methods

Static NetMethods are used for anything that isn't tied to an instance object, such as chat messages. When received, the game will call the static receiving method directly when read.

The following snippet from `EffectManager` is a good example of static methods being used for both directions of communication:

```
// Server to client. Modifies the text of an element within a UI effect.
private static readonly ClientStaticMethod<short, string, string> SendUIEffectText =␣
↪ClientStaticMethod<short, string, string>.Get(ReceiveUIEffectText);
```

(continues on next page)

(continued from previous page)

```
[SteamCall(ESteamCallValidation.ONLY_FROM_SERVER, legacyName = nameof(tellUIEffectText))]
public static void ReceiveUIEffectText(short key, string childNameOrPath, string text)
{
        // ...
}

// Client to server. Notifies the server that a button in a UI effect was clicked.
private static readonly ServerStaticMethod<string> SendEffectClicked = ServerStaticMethod
↪<string>.Get(ReceiveEffectClicked);

[SteamCall(ESteamCallValidation.SERVERSIDE, ratelimitHz = 20, legacyName =␣
↪nameof(tellEffectClicked))]
public static void ReceiveEffectClicked(in ServerInvocationContext context, string␣
↪buttonName)
{
        Player player = context.GetPlayer();
        if (player == null)
                return;

// Invoke the event on the server to alert any listening plugins.
        onEffectButtonClicked?.Invoke(player, buttonName);
}
```

130.1.2 Instance Methods

Instance NetMethods call a method on one specific object. When received, the generated `_Read` method reads the NetId, finds the object paired with it, and called the receive method on that object. The call is ignored if no pairing is found, or if the object type is unexpected.

Because the receive method is not static, the handle is created from the type and method name instead. The following snippet from `InteractableFarm` is a good example of instance methods being used for both directions of communication:

```
    // Server to client. Updates when this crop was planted.
    internal static readonly ClientInstanceMethod<uint> SendPlanted =␣
↪ClientInstanceMethod<uint>.Get(typeof(InteractableFarm), nameof(ReceivePlanted));

    [SteamCall(ESteamCallValidation.ONLY_FROM_SERVER, deferMode =␣
↪ENetInvocationDeferMode.Queue)]
    public void ReceivePlanted(uint newPlanted)
    {
            updatePlanted(newPlanted);
    }

    // Client to server. Requests to harvest this crop.
    private static readonly ServerInstanceMethod SendHarvestRequest =␣
↪ServerInstanceMethod.Get(typeof(InteractableFarm), nameof(ReceiveHarvestRequest));

// Invokes the static handle to trigger the harvest request.
    public void ClientHarvest()
    {
```

(continues on next page)

(continued from previous page)

```
            SendHarvestRequest.Invoke(GetNetId(), ENetReliability.Unreliable);
    }

    [SteamCall(ESteamCallValidation.SERVERSIDE, ratelimitHz = 10)]
    public void ReceiveHarvestRequest(in ServerInvocationContext context)
    {
            // ...
    }
```

130.2 Reading and Writing Parameters

How a NetMethod's data is read and written depends on the receive method's parameters.

130.2.1 Generated Parameters

If the receive method has basic parameters, like `ReceiveUIEffectText(short key, string childNameOrPath, string text)`, the NetMethod is read and written by generated code. The handle's type arguments have to match the parameters, and the parameter values are passed in directly on invocation.

```
SendUIEffectText.Invoke(reliability, transportConnection, key, childNameOrPath, text);
```

The receive method can also take an invocation context (`in ClientInvocationContext` or `in ServerInvocationContext`) as the first parameter. The generated `_Read` method passes it along. Server methods commonly use this to find the calling player, such as `ReceiveEffectClicked` above.

130.2.2 Manual Parameters

If the receive method *only* has a parameter that is the invocation context, no code will be generated for its parameters. Instead, the receive method is made responsible for reading its own data from `context.reader`, and the sender is resposible for writing it. This is typically used for data that the generator can't handle, such as large batches of objects (e.g. barricade or structure regions).

For instance methods, a `_Read` method will still be generated. This method still reads the NetId and find the object it points to. It then calls the receive method, which reads the rest of the data.

```
private static readonly ClientInstanceMethod SendScreenshotDestination =␣
↪ClientInstanceMethod.Get(typeof(Player), nameof(ReceiveScreenshotDestination));

// Writes the data using a callback.
SendScreenshotDestination.Invoke(GetNetId(), ENetReliability.Reliable,␣
↪transportConnection, SendScreenshotDestination_Write, length, data);

private void SendScreenshotDestination_Write(NetPakWriter writer, ushort length, byte[]␣
↪data)
{
        writer.WriteUInt16(length);
        writer.WriteBytes(data);
}

// Reads the data in the same order.
[SteamCall(ESteamCallValidation.ONLY_FROM_SERVER)]
public void ReceiveScreenshotDestination(in ClientInvocationContext context)
```

(continues on next page)

(continued from previous page)

```
{
        NetPakReader reader = context.reader;
        ushort length;
        reader.ReadUInt16(out length);
        byte[] data = new byte[length];
        reader.ReadBytes(data);
        HandleScreenshotData(data);
}
```

Note

As with most networking data, data must be read in the same order and format that it was written in. Mismatches may cause silent errors.

130.3 NetIds

A `NetId` is a struct that wraps a 32-bit unsigned integer (`uint`). The `NetIdRegistry` class pairs each NetId with an object (e.g. `InteractableFarm` or `Player`) or specifically a `Transform`. This is then used to allow the server and client to agree on which object an instance NetMethod should call its receive method on. A NetId of `0` represents an invalid identifier.

For example, there could be many farm crops around the map. When a client attempts to harvest one, it sends that crop's NetId over so the server knows which specific `InteractableFarm` it was for.

NetIds are assigned by either the server or calculated from the level. For the server, it claims a block of NetIds for each new barricade, structure, vehicle, or player. For the level, trees and objects calculate their NetIds based on their level data position, so they don't need to be sent over the network.

The registry is cleared before a level is loaded. NetIds are considered volatile and should not be saved or reused between sessions / restarts.

Tip

Most objects that have an assigned NetId such as `Player`, `SteamPlayer`, `BarricadeDrop`, `StructureDrop`, and `Interactable` have a `GetNetId()` method. You can use `NetIdRegistry.Get<T>(netId)` to find the object paired with a NetId.

130.4 SteamCall Attribute

Required for receive methods of any NetMethod.

validation *ESteamCallValidation* (`ONLY_FROM_SERVER`, `SERVERSIDE`, `ONLY_FROM_OWNER`): Determines who is allowed to invoke the method.

ratelimitHz *int32*: Maximum number of calls per second for each player. Players that repeatedly exceed the limit may be kicked.

deferMode *ENetInvocationDeferMode* (`Queue`, `Discard`): Determines the method behaviour when a client receives an instance method for a NetId that is not registered yet. This may be the case for objects that are pending instantiation. `Queue` holds the call until the object exists. `Discard` will ignore it.

legacyName *string*: Older name for the method. This has been kept for backwards compatibility with plugins that invoked RPCs by name. Usage is deprecated.

130.5 Generated Methods

NetMethods require methods for reading and writing networking data. These are automatically generated for the game and are contained within `<Type>_NetMethods` classes containing `_Read` and `_Write` methods. You can generate this code in Unity using the `Window > Unturned > Net Gen` menu.

CHAPTER

ONE

OPENMOD

OpenMod is a spiritual successor to *Rocket* as 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.

131.1 Installation

131.1.1 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.

131.1.2 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.

131.2 Plugins

You can find open-source OpenMod plugins here.

131.2.1 Resources

- GitHub Repository
- Documentation

131.2.2 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

TWO

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.

132.1 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.

132.2 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`.

132.3 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

THREE

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.

133.1 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.

133.2 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.

The RocketMod organization on GitHub hosts several related archived projects: RocketMod (Abandoned)

133.3 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

FOUR

SERVER AUTO RESTART

Server maintenance can be somewhat automated without plugins, by utilizing a couple features from your server's Config.json file.

134.1 Scheduled Maintenance

We recommend restarting your server once every 24 hours, as much of the game's older code still uses single-precision floating point time.

There are three settings for scheduled shutdowns:

1. **Enable_Scheduled_Shutdown**: When `true`, the server will shutdown at a specified time.
2. **Scheduled_Shutdown_Time**: Local time for when the server will shutdown.
3. **Scheduled_Shutdown_Warnings**: Players will be notified via a chat broadcast at these times before the scheduled shutdown. For example, 30:00 will broadcast in chat 30 minutes before the shutdown.

134.2 Checking for Updates

Your server can monitor for updates, and shutdown when one is detected. Three settings are important when checking for updates:

1. **Enable_Update_Shutdown**: When `true`, the server will monitor for updates.
2. **Update_Steam_Beta_Name**: Defaults to `public`, but can be set to `preview` for servers running on the preview branch.
3. **Update_Shutdown_Warnings**: After an update is detected, the server will wait for the longest of these durations to notify players before shutdown. For example, if the longest time is 2:30, the server will broadcast in chat 2 minutes and 30 seconds before the shutdown.

134.3 Practical Application

These options are most useful in conjunction with a script that updates and restarts the server in a loop. For example, this Windows .bat script can be placed in the steamcmd folder to infinitely update and restart the server:

```
1 @echo off
2 rem @ suppresses echo command from being echoed, and then disables echoing in this␣
  ↪script.
3
4 rem This is a label for use with "goto". The script will return to this label to update␣
  ↪and restart the server.
```

(continues on next page)

(continued from previous page)

```
:loop

rem %~dp0 expands to the path to this script's directory, allowing it to be called from a different working directory.
rem The "/wait" option pauses our script until steamcmd is finished.
rem Start steamcmd, download latest version of Unturned Dedicated Server, then close cleanly.
echo Updating...
start "" /wait "%~dp0steamcmd.exe" +login anonymous +app_update 1110390 +quit

echo Finished update! Launching server...
start "" /wait "%~dp0steamapps\common\U3DS\Unturned.exe" -batchmode -nographics +InternetServer/MyServer

echo Server has exited. Restarting after timeout...
echo:
echo Press CTRL+C and then Y during this timeout to cancel restart.
timeout 10

rem Return to the "loop" label to update and restart the server.
goto loop
```

Here's a similar example for Linux, tested on Ubuntu. Note that the U3DS install path may be different depending how steamcmd was installed and whether `+force_install_dir` was specified, but typically one of these locations:

1. `~/.steam/steam/steamapps/common/U3DS`
2. `~/Steam/steamapps/common/U3DS`

As such, it may be helpful to create a symbolic link to the U3DS directory:

`ln -s ~/.steam/steam/steamapps/common/U3DS ~/U3DS`

OR

`ln -s ~/Steam/steamapps/common/U3DS ~/U3DS`

This example script assumes the symbolic link exists at `~/U3DS`.

1. Save script as `MyServer.sh` in your home directory (e.g., `nano ~/MyServer.sh`):

```
#! /usr/bin/bash
while true; do
        echo Updating...
        steamcmd +login anonymous +app_update 1110390 -validate +quit

        echo Finished update! Launching server...
        cd ~/U3DS
        source ServerHelper.sh +InternetServer/MyServer

        echo Server has exited. Restarting after timeout...
        echo Press Ctrl+C during this timeout to cancel restart.
        read -t 10
done
```

2. Open a screen to run the server in with `screen -S MyServer`.

3. Run script with `bash MyServer.sh`.
4. If everything's working OK, press `Ctrl + A` then `D` to detach, leaving the script running.
5. You can re-attach to the screen with `screen -r MyServer`.

CHAPTER

FIVE

SERVER BROWSER CURATION

This feature allows anyone to create and share lists of "rules" that filter or label servers in the server browser. Lists can be shared through the Steam Workshop as *Server Browser Curation Assets*, or automatically downloaded from a URL on the Internet.

If you're a server host, suppose you want to prevent bad actors from copying your server details. Your first rule would `Allow` your genuine servers, for example, matching by `ServerID` ("server code"). Your second rule could then `Deny` servers with a regex that matches your server network's branding.

- *Examples*
- *FAQ (Frequently Asked Questions)*

135.1 Properties

135.1.1 Name string

Your display name in the user interface. For example, a server network might display as "MyNetwork Verified Servers", or someone creating a list of high-quality well-moderated hosts might choose "MyName's Recommendations".

135.1.2 IconURL string

Optional URL of a 32x32 image to display in the user interface. For example, a server network might use the same icon they use in the server browser.

135.1.3 Labels list of dictionaries

Labels applied by rules are defined here. Each has the following properties:

Name *string*: Name used to refer to this label from a rule.

Text *string*: Rich text displayed in the server browser.

For example, defining two labels:

```
Labels
[
        {
                Name Verified
                Text <color=green>MyNetwork Verified</color>
        }
        {
                Name Fake
                Text <color=red>MyNetwork Imposter</color>
```

(continues on next page)

(continued from previous page)

```
        }
]
```

135.1.4 Rules list of dictionaries

Rules are processed from top to bottom. Each has the following properties:

Action *enum* (`Label`, `Allow`, `Deny`): `Label` applies a label and continues processing rules, whereas `Allow` and `Deny` both stop rule processing. `Deny` blocks the server, either hiding it or moving it to the bottom depending on the player's settings.

> **Note**
>
> `Allow` and `Deny` **can** apply labels as well. The only difference is that `Label` action doesn't affect whether server is allowed or denied.

Inverted *bool*: If true, negate whether this rule matches. i.e., binary NOT.

Description *string*: Text shown in the rules list user interface and in the tooltip for servers moved to the bottom of the list. Please use this to document *why* your rule exists!

Label *string*: Name of a label to apply.

Type *enum* (`Name`, `IPv4`, `ServerID`): Determines which server data this rule matches against.

- **Name**: Match server's name using one or more regular expression (regex). Refer to **Regex** or **Regexes** property.
- **IPv4**: Match server's public IP using one or more CIDR addresses. Refer to **Filter** or **Filters** property.
- **ServerID**: Match server's ID (also referred to as "server code" or Steam ID). Refer to **Value** or **Values** property.

Regex *string* or **Regexes** *list of string*: The rule matches if any of the regexes match the server name. (binary OR)

One quick way to test a regex is to prefix it with "regex:" in the server browser name filter to search by regex. There are also a variety of helpful, free regex creation tools online.

For example, matching any server with "MyNetwork" in the name: `Regex (?i)(MyNetwork)` (`(?i)` enables case-insensitive matching)

Filter or **Filters**: The rule matches if any of the rules match the server IPv4 address. (binary OR)

IPv4 address with optional subnet mask and optional port number or range. Here's an example list with explanations:

```
Filters
[
        // Matches any port on this IP address.
        10.8.0.1

        // Matches port 27015.
        10.8.0.1:27015

        // Matches ports 27015 through 27030 (inclusive).
        10.8.0.1:27015-27030

        // Matches any port on IP addresses in the range 192.168.1.0 through 192.168.1.
↪255.
```

(continues on next page)

(continued from previous page)

```
        192.168.1.0/24
]
```

Value *uint64* or **Values** *list of uint64*: The rule matches if any of the Steam IDs match the server's Steam ID. (binary OR)

From the server console you can copy the server ID with "CopyServerCode". From the in-game server lobby screen you can copy it to the clipboard by pressing PageDown.

135.2 Examples

Here's a hypothetical verification list for the "NelsonNet" network:

```
Name NelsonNet Verification Example
IconURL https://cdn.smartlydressedgames.com/ShareX/2024/12/ExampleIcon.png

Labels
[
        {
                Name Verified
                Text <color=#708fbd>NelsonNet Official</color>
        }
]

Rules
[
        {
                Action Allow
                Description Verify NelsonNet's Steam IDs
                Label Verified
                Type ServerID
                Values
                [
                        85568392932910946
                ]
        }
        {
                Action Deny
                Description Hide fake NelsonNet servers (case-insensitive check for
↪"NelsonNet" in the name)
                Type Name
                Regex (?i)(NelsonNet)
        }
]
```

For an example of adding a curator by web URL, here's the link for that list: `https://cdn.smartlydressedgames.com/ShareX/2024/12/ExampleCurationList.txt`

135.3 FAQ (Frequently Asked Questions)

135.3.1 How can I find the details for someone else's server?

Pressing the **"Clipboard Debug"** hotkey (default **PageDown**) on the server lobby screen copies a variety of public information about the server to your clipboard. For example:

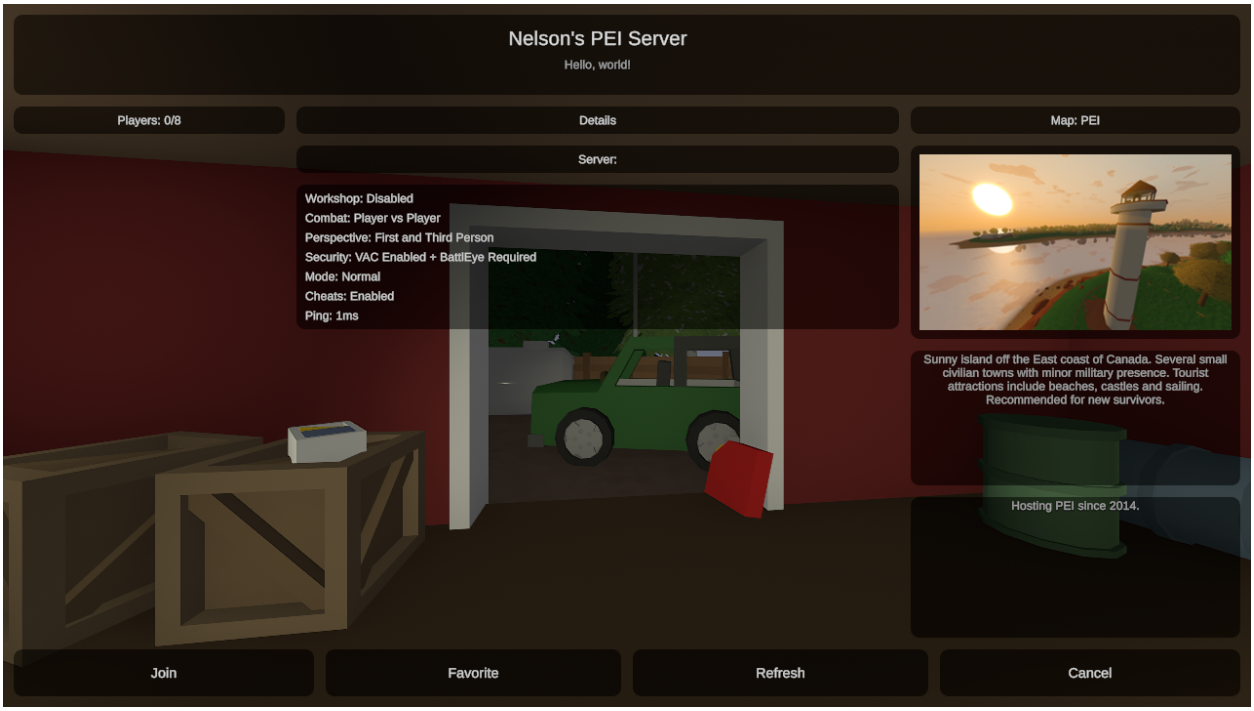


Fig. 1: An example server.

Pressing the key on this server copies the following information to the clipboard:

```
1 Name: Nelson's PEI Server
2 Description:
3 Thumbnail:
4 Address: 192.168.48.73
5 Connection Port: 27016
6 Query Port: 27015
7 SteamId: 85568392932910946 (k_EAccountTypeGameServer)
8 Ping: 1ms
9 0 workshop file(s):
```

Relevant details being:

- Name (line 1) can be used in a **Name** rule.
- Address (line 4) can be used in an **IPv4** rule.
- SteamId (line 7) can be used in a **ServerID** rule.

135.3.2 How do I create a regular expression (regex)?

As a starting point, `(?i)(your text here)` matches "your text here" (without quotes) case-insensitively. It would also match "Your Text Here" or "yOuR tExT hErE", but not "yourtexthere".

For more advanced uses, we'd recommend using an online tool. At the time of writing (2025-02-07) these are top search results for "regex tool":

1. https://regex101.com/
2. https://regexr.com/
3. https://regex-generator.olafneumann.org/
4. https://www.regextester.com/

CHAPTER

SIX

SERVER CODES

A **Server Code** is a randomly generated, 17-digit numeric code assigned to your server. Your friends can join your multiplayer server by inputting its **Server Code** in the Connect Directly menu without you needing to *port forward*.

Your server code can be found by clicking the "Copy Server Code" button while in your server. Alternatively, it appears in the Server Console immediately after your server finishes loading, and can be easily copied to your clipboard with the `CopyServerCode` command.

If you would like to limit player connections to Server Codes, you can utilize the `-SNS_DisableIPSocket` launch option. Likewise you can use `-SNS_DisableP2PSocket` for disabling Server Code (P2P) connections.

136.1 Limitations

The primary downside of Server Codes is their incompatibility with the pre-joining server info screen (which displays things like the server name, installed mods, current players, and other details). The info screen uses Steam's A2S protocol, which can only be queried by IP, so joining by Server Code enters the server immediately instead. You can enable *Fake IP* to work around this limitation.

Your server's Server Code will change each time your server restarts. You can assign a *Game Server Login Token* to keep your Server Code linked between sessions.

136.2 How does it work?

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.*

CHAPTER

SEVEN

SERVER CONFIGURATION

Most server options are found in your server's `Config.txt` file (e.g., `Servers/Default/Config.txt`). If your server was created before **3.25.8.0**, please refer to the **How is Config.json converted to Config.txt?** and **Legacy Json File** sections.

To check which options are overridden by your config you can add the `-LogGameplayConfig` command-line flag to print overrides to the console.

Values left empty will use the default. They can be overridden by levels unless the `-NoLevelConfigOverrides` command-line flag is used.

137.1 How is Config.json converted to Config.txt?

Servers created before **3.25.8.0** have a `Config.json` file rather than `Config.txt`. The `.json` file stored options for each difficuly (easy, normal, hard), so during startup a per-difficulty `.txt` file is converted. For example, if the server ID is "ExampleServer" and difficulty is Normal (default), the file will be created at: `Servers/ExampleServer/Config_NormalDifficulty.txt`

Only options changed from the default value are converted. This is because empty values in `Config.txt` indicate the default should be used.

137.2 How can I change which config file is loaded?

You can override the config file path with the `-GameplayConfigFile` command-line option. For example, if your server ID is "ExampleServer", here's how to use this option:

`-GameplayConfigFile="SpecialEventConfig.txt"`

will create

`Servers/ExampleServer/SpecialEventConfig.txt`.

Alternatively, you can provide an absolute file path like:

`-GameplayConfigFile="C:/SpecialEventConfig.txt"`

137.3 How can I remove generated comments?

If you don't want explanations for each option in `Config.txt` you can add the `-GameplayConfigNoGeneratedComments` command-line flag. This removes all comments prefixed with >.

This is useful for a more concise config file.

137.4 How can I remove unused options?

If you don't want all available options listed in `Config.txt` you can add the `-GameplayConfigNoEmptyValues` command-line flag. This removes all non-overridden options.

This is useful for a more concise config file.

137.5 How can I prevent levels from overriding my config?

If you assign a value in `Config.txt` it will take priority over level overrides, whereas empty values (default) can be overridden by levels.

To entirely prevent levels from overriding any config, you can add the `-NoLevelConfigOverrides` command-line flag.

137.6 Legacy Json File

Older server folders have a `Config.json` file. During startup, the server will automatically convert it to the newer format according to the difficulty (easy/normal/hard).

Downsides of the older format:

1. Level overrides take precedent over default values. The server compares level config overrides to the values in the json file, and only overrides if it matches the default value. This means if you prefer the default (e.g., Hitmarkers ON) but the level turns them off it's not possible to prevent it without the `-NoLevelConfigOverrides` command-line flag.
2. It contains easy, normal, and hard difficulty values in the same file. It's confusing to change a value but not have it apply because it's for a different difficulty level.
3. Common to forget commas after values which breaks loading the file.
4. No support for comments.

Benefits of the newer format:

1. All settings have comments explaining their use, options, and default value. Your own comments are preserved alongside the autogenerated ones.
2. Separate files per difficulty, as well as the option for additional configs with the `-GameplayConfigFile` command-line option.
3. Easier for new hosts because notepad can open it by default, and commas are not required.

If you wish to continue using the `Config.json` file instead of the newer file, you can add this command-line flag: `-UseLegacyJsonGameplayConfig`

CHAPTER

EIGHT

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:

138.1 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.

138.2 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

NINE

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

ASSET POINTER

When an asset refers to another it does so using an **Asset Pointer**. These identify the target asset by its *GUID*.

140.1 In *.dat files

Note that the GUID here is not wrapped in quotes because Unturned *.dat files are treated as pairs of strings.

```
MyAssetPtr ################################
```

140.2 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" "################################" }
```

140.3 In JSON files

```
"MyAssetPtr": { "GUID": "################################" }
```

CHAPTER

ONE

BITMASK

Recommended Wikipedia Article

141.1 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

TWO

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
}
```

142.1 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

THREE

ENUMERATED TYPES

143.1 EAssetType

The EAssetType enumerated type is used as a scope for legacy IDs. Each legacy ID is unique within an EAssetType (allowing multiple assets to share the same legacy ID when they are different types). This is only used in older code or for maintaining backwards compatibility.

143.1.1 Enumerators

Named Value	Index	Description
`None`	0	Asset type is not applicable.
`Item`	1	Asset is an item.
`Effect`	2	Asset is an effect.
`Object`	3	Asset is an object (this includes NPC characters).
`Resource`	4	Asset is a resource.
`Vehicle`	5	Asset is a vehicle.
`Animal`	6	Asset is an animal.
`Mythic`	7	Asset is a mythical effect.
`Skin`	8	Asset is a skin.
`Spawn`	9	Asset is a spawn table.
`NPC`	10	Asset is related to NPCs (such as quests, vendors, or dialogues).

143.2 EBatteryMode

The EBatteryMode enumerated type is used to determine how a vehicle battery should behave. This is only used by *vehicle assets*.

143.2.1 Enumerators

Named Value	Description
`None`	Battery charge remains unchanged.
`Burn`	Battery charge will deplete over time.
`Charge`	Battery charge will replenish over time.

143.3 EItemOrigin

The EItemOrigin enumerated type is used when spawning items.

143.3.1 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.

143.4 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.

143.4.1 Enumerators

Named Value	Index	Description
`Common`	0	Corresponds to the "Common" (white-colored) rarity.
`Uncommon`	1	Corresponds to the "Uncommon" (green-colored) rarity.
`Rare`	2	Corresponds to the "Rare" (blue-colored) rarity.
`Epic`	3	Corresponds to the "Epic" (purple-colored) rarity.
`Legendary`	4	Corresponds to the "Legendary" (pink-colored) rarity.
`Mythical`	5	Corresponds to the "Mythical" (red-colored) rarity.

143.5 EItemType

The EItemType enumerated type consists of all the possible item types.

143.5.1 Enumerators

Named Value	Index	Description
`Hat`	0	Corresponds to the "Hat" item type.
`Pants`	1	Corresponds to the "Pants" item type.
`Shirt`	2	Corresponds to the "Shirt" item type.
`Mask`	3	Corresponds to the "Mask" item type.
`Backpack`	4	Corresponds to the "Backpack" item type.
`Vest`	5	Corresponds to the "Vest" item type.
`Glasses`	6	Corresponds to the "Glasses" item type.
`Gun`	7	Corresponds to the "Gun" item type.
`Sight`	8	Corresponds to the "Sight" item type.
`Tactical`	9	Corresponds to the "Tactical" item type.
`Grip`	10	Corresponds to the "Grip" item type.
`Barrel`	11	Corresponds to the "Barrel" item type.
`Magazine`	12	Corresponds to the "Magazine" item type.

continues on next page

Table 1 – continued from previous page

Named Value	Index	Description
Food	13	Corresponds to the "Food" item type.
Water	14	Corresponds to the "Water" item type.
Medical	15	Corresponds to the "Medical" item type.
Melee	16	Corresponds to the "Melee" item type.
Fuel	17	Corresponds to the "Fuel" item type.
Tool	18	Corresponds to the "Tool" item type.
Barricade	19	Corresponds to the "Barricade" item type.
Storage	20	Corresponds to the "Storage" item type.
Beacon	21	Corresponds to the "Beacon" item type.
Farm	22	Corresponds to the "Farm" item type.
Trap	23	Corresponds to the "Trap" item type.
Structure	24	Corresponds to the "Structure" item type.
Supply	25	Corresponds to the "Supply" item type.
Throwable	26	Corresponds to the "Throwable" item type.
Grower	27	Corresponds to the "Grower" item type.
Optic	28	Corresponds to the "Optic" item type.
Refill	29	Corresponds to the "Refill" item type.
Fisher	30	Corresponds to the "Fisher" item type.
Cloud	31	Corresponds to the "Cloud" item type.
Map	32	Corresponds to the "Map" item type.
Key	33	Corresponds to the "Key" item type.
Box	34	Corresponds to the "Box" item type.
Arrest_Start	35	Corresponds to the "Arrest_Start" item type.
Arrest_End	36	Corresponds to the "Arrest_End" item type.
Tank	37	Corresponds to the "Tank" item type.
Generator	38	Corresponds to the "Generator" item type.
Detonator	39	Corresponds to the "Detonator" item type.
Charge	40	Corresponds to the "Charge" item type.
Library	41	Corresponds to the "Library" item type.
Filter	42	Corresponds to the "Filter" item type.
Sentry	43	Corresponds to the "Sentry" item type.
Vehicle_Repair_Tc	44	Corresponds to the "Vehicle_Repair_Tool" item type.
Tire	45	Corresponds to the "Tire" item type.
Compass	46	Corresponds to the "Compass" item type.
Oil_Pump	47	Corresponds to the "Oil_Pump" item type.
Vehicle_Paint_Toc	48	Corresponds to the "Vehicle_Paint_Tool" item type.

143.6 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.

143.6.1 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.

143.7 ENPCHoliday

The ENPCHoliday enumerated type consists of all of the game's recognized holidays or seasonal events. You can find the duration of scheduled seasonal events in the `Client.log` file generated after launching the game, or in the `HolidayUtil.cs` source file.

The start and end times for all holidays are relative to the player's local time, meaning they are affected by timezones. For Lunar New Year, the start and end dates are automatically calculated and the holiday's duration is set in the game's `Status.json` file.

> **Note**
>
> Some assets only support a few of the game's recognized holidays. Notably: Landscape Material Assets only support Halloween, Christmas, and April Fools' Day.

143.7.1 Enumerators

Named Value	Description	Duration
`None`	Represents no holiday or seasonal event.	
`Halloweer`	Corresponds to the Halloween holiday.	October 20, 2025 (00:00) – November 1, 2025 (12:00)
`Christma`	Corresponds to the Christmas holiday, and other holidays within the festive season.	December 7, 2025 (00:00) – January 2, 2026 (12:00)
`April_Fo`	Corresponds to the April Fools' Day holiday.	April 1, 2025 (00:00) – April 1, 2025 (23:59:59)
`Valentin`	Corresponds to the Valentine's Day holiday.	February 14, 2025 (00:00) – February 14, 2025 (23:59:59)
`Pride_Mo`	Corresponds to Pride Month, a month-long observance in June.	June 1, 2025 (00:00) – June 30, 2025 (23:59:59)
`Lunar_Ne`	Corresponds to the Lunar New Year holiday.	Varies based on the Chinese calendar, being calculated as the day *before* Lunar New Year to 15 days *after* Lunar New Year. For example: January 28, 2025 (00:00) – February 13, 2025 (23:59:59).
`Unturned`	Corresponds to the game's anniversary on Steam.	July 7, 2025 (00:00) – July 7, 2025 (23:59:59)
`Max`	Only used/implemented in the game's source code – no practical use for game assets.	

143.8 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*.

143.8.1 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.

143.9 EObjectType

The EObjectType enumerated type consists of all the possible object types.

143.9.1 Enumerators

Named Value	Index	Description
`Large`	0	Corresponds to the "Large" object type.
`Medium`	1	Corresponds to the "Medium" object type.
`Small`	2	Corresponds to the "Small" object type.
`NPC`	3	Corresponds to the "NPC" object type.
`Decal`	4	Corresponds to the "Decal" object type.

143.10 ESlotType

The ESlotType enumerated type is used for item placement in displays or on characters, and determines which item slot(s) an equippable item can be placed in. Some assets may only support using specific enumerators.

143.10.1 Enumerators

Named Value	Description
`None`	Does not correspond to any slot. Equippables can be hotkeyed.
`Primary`	Corresponds to the "Primary" slot. Equippables can be used from the primary slot.
`Secondary`	Corresponds to the "Secondary" slot. Equippables can be used from the primary or secondary slot.
`Tertiary`	Corresponds to the "Tertiary" slot. This is only used by NPCs.
`Any`	Corresponds to any/all item slots. Equippables can be used from any slot, or hotkeyed.

CHAPTER

FOUR

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

FIVE

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.

CHAPTER

SIX

MASTER BUNDLE POINTER

Identifies a Unity asset like a prefab, material or audio clip within a master bundle.

146.1 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

SEVEN

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.

147.1 Extended Tags

These tags are specific to *Unturned*.

**
**: 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.

147.2 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

147.3 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

EIGHT

STRUCTS

148.1 PlayerSpotLightConfig

The PlayerSpotLightConfig struct contains properties for configuring player spot lights. Certain item assets are able to utilize these properties.

148.1.1 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`

148.1.2 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

NINE

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
}
```

149.1 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
```

FAQ

Unturned's project files are available in a GitHub repository.

150.1 General

Q: Why make the game's source code available?

A: Unturned's greatest strength is our passionate community. Releasing the source files allows you and other players to further expand upon the game as you see fit, building a lasting legacy for the game regardless of the changes we make or how many years pass from now.

Q: How will this affect cheat development? (Will cheating be easier?)

A: The source code doesn't add much new information for cheat developers. High quality decompilations of Unturned already exist (reverse engineering the code), and we've previously shared the code's documentation. BattlEye's Unturned-specific anti-cheat code is still private (even we don't have access to it), and they will continue monitoring for any new cheats.

Q: Can I publish my derivate works on Steam?

A: We intend to support this if possible. For more information about mods on Steam, please refer to: https://store.steampowered.com/about/communitymods/

Q: Will the Steam version continue receiving updates?

A: Yes, we plan to continue maintaining the official Steam version.

Q: Will the project files be kept up-to-date with the version used by the official game?

A: Yes, we plan to keep the source code in sync with the latest official releases.

Q: Is the game "Open Source" now?

A: *Open Source* refers to the Open Source Initiative's definition, which the current non-commercial license doesn't meet. Ideally, once we finish another game and are hopefully able to re-acquire certain rights, we'd like to re-license the U3 SDK under the MIT License.

Q: Do I need a paid Unity license?

A: At the time of writing (2026-07-02), individuals and hobbyists earning less than $200,000 USD/year from their use of Unity are elligible for a free Unity Personal license. That being said, you should review Unity's terms.

Q: Will one-off custom forks cause fragmentation?

A: Our hope is to see like-minded devs join together around a shared fork, collaborating on enhancements that benefit all of their projects. (*The whole is greater than the sum of its parts!*)

150.2 Technical

Q: Why is the SDK ignoring my settings?

A: The SDK stores most save data (including certain settings) separately from the base game's save data. After you apply the same settings within the SDK they will be persisted.

Q: How do I export a development build?

A: The "development" build has additional logging, visualizations, and debug tools (e.g., the profiler) enabled. Within Unity open **Window > Unturned > Build Tool**. Select **Build Test** and once it completes run `Builds/Test/Unturned.exe`.

Q: How do I export release builds?

A: The Windows, macOS, and Linux builds are intended for distribution through Steam as a standalone mod. Within Unity open **Window > Unturned > Build Tool** and select **Build Standalone Platforms**.

Q: Can I host servers?

A: Yes: servers for your mod are filtered according to the `Name` configured in `Builds/Shared/ModInfo.json`. Development builds can run as either client or server, and the Unity editor can run a game server by enabling **Playing in Unity > Dedicated Server In Editor**.

Q: Will my workshop mods work?

A: By default, any regular Unturned workshop file should be compatible and automatically load. Note that upgrading your mod's Unity engine version may affect compatibility.

Q: Will my plugins work?

A: By default, most plugins for Unturned servers should be compatible. Custom code changes to your mod may affect compatibility. For example, renaming or removing code that plugins depend on will break compatibility with those plugins.

150.3 Contributing

Q: How can I contribute?

A: Two of the best ways you can get involved are helping out with a community fork (or starting one!), or participating in Discussions (especially answering questions!).

Q: How can I report an issue?

A: If it's related to the U3-SDK (as opposed to the vanilla game or a mod), please create an issue on the GitHub Issue Tracker!

Q: How can I report a security vulnerability?

A: If you've found an exploit affecting the vanilla game, please don't create a publicly visible issue. Instead, send an email to info@smartlydressedgames.com.

Q: Where can I get discuss development and ask questions?

A: Many developers frequent the GitHub Discussions. It's a great place to meet fellow modders, and if you're polite someone might be kind enough to help you out!

Q: Will you accept pull requests?

A: Our current goal is simply allowing players to create their own spin on the game. From minor balance changes and item additions, to total conversions with new gameplay elements. If there is significant community interest behind a mod we may reach out to collaborate in some form or another.

150.4 Dependencies

Q: Has any code been removed from this release?

A: Yes, certain code we can't redistribute has been removed. For example, third-party pathfinding and anti-cheat libraries. If you have an appropriate license to use those libraries you may use the integration from the base game in your project if you wish.

Q: Why do zombies walk into walls?

A: Smarter zombie navigation depends on the A* Pathfinding Project. If you have a license for this library, please feel free to use the integration from the base game.

Q: Why do interaction and selection tint the entire model?

A: Edge highlighting depends on a Highlighting System Plugin. If you have a license for this plugin, please feel free to use the integration from the base game.

Q: Why does the water look so simple?

A: Better water shaders and planar reflections depend on Unity 4 Pro Standard Assets. If you have a license for these assets, please feel free to use the integration from the base game.

Q: Why are missing component warnings logged in some levels?

A: Certain prefabs include NavmeshCut Components which can't be loaded without the pathfinding library.

150.5 Assets

Q: Can I modify the core assets?

A: We're comfortable with our *own* in-house assets being modified, however the core asset bundle has been intentionally excluded from the SDK. It contains some third-party licensed content (examples: community-submitted skins, audio files). Unfortunately, we don't have an easy and definitive list we can share of which assets are third-party versus created in-house. Thus we're unable to provide blanket permission for redistributing the entire set of core assets, or to advise on which specific assets could be relicensed or replaced.

Q: How can large assets be included in source control?

A: Internally, we use Git LFS. The SDK's `.gitattributes` file is prepared for Git LFS, with a comment on the first line describing the setup.

Q: Can I use Git LFS without enabling it on my public fork?

A: One option is two use two repositories: a public repository containing your fork's source code, and a private downstream repository containing your asset files.

150.6 Restrictions

Q: Can I commercialize/monetize my mod?

A: No, sorry, it must be strictly non-commercial. Please refer to the full license text for more details.

Q: What is considered non-commercial?

A: Players must be able to play and access all features and content of your mod for free. Taking payments (whether inside or outside the game) in return for in-game content or services is considered commercial use.

Q: Can my mod accept donations?

A: Accepting voluntary donations is fine. Providing in-game items, upgrades, benefits, etc for "donations" is considered commercial use.

Q: Can I offer out-of-game benefits to Patrons?

A: Yes, for example, special Discord roles for your donors is fine.

Q: Can I enable ads and sponsorships on YouTube videos about my mod?

A: Yes, this is not considered commercial use of the SDK.

Q: Are there any restrictions on mod naming?

A: Your mod must not imply it was created or endorsed by SDG. For example, you cannot title your mod "Unturned 2". Please refer to the full license text for more details.

Q: What platforms can I develop my mod for?

A: The SDK can target the same PC platforms as the Steam version of Unturned: Windows, macOS, and Linux (including Steam Deck). Please refer to the full license text for more details.

Q: Do I need to publish the source code for my mod?

A: No, you can keep your mod's code private if you wish.

Q: How are custom forks' servers affected by moderation?

Our server hosting guidelines only apply to servers that are accessible through Unturned's public server list. By default, servers hosted on forked projects are not visible in our public server list, although it is technically possible to make them appear there. Regardless of which version of the game a server is running, if it is listed in our public server list, it should comply with our server hosting guidelines.

That said, maintainers of forked projects are free to establish and enforce their own server hosting guidelines. If you host a server for a forked project, you would need to follow any rules they set, or you may be subject to moderation by that project's maintainers.

CHAPTER

ONE

UNITY PROJECT OVERVIEW

151.1 Downloading

Unturned's project files are stored using the Git version control system (VCS).

If you have the Git CLI installed, you can clone the files to your computer with this command:

```
git clone https://github.com/SmartlyDressedGames/U3-SDK.git
```

151.2 Getting Started

You'll need the same version of the Unity editor as described in *Installing Unity*. You can double-check the editor version in `ProjectSettings/ProjectVersion.txt`.

Steam needs to be running, and Unturned must be installed. (Workshop mods and large binary files are loaded from the latest official release of the game.)

To run the game in the editor, open the `GameStartup.unity` scene and click Play.

> **Tip**
>
> We recommend closing Unity's **Hierarchy** window in-game except when you need it. Unturned's scenes contain mostly top-level game objects for optimization purposes with the drawback of slowing down the Hierarchy window. For more information, Scenes Structure > Hierarchy depth and count.

151.3 Editor Preferences

Unfortunately, Unturned doesn't support hot-reloading. If modifying code while Unity is open, we recommend changing **Script changes while playing** to **Recompile after playing**.

151.4 Play Mode Settings

An editor window is available from Window > Unturned > Editor Settings. Primarily it sets *options* that are otherwise specified on the command-line.

Auto Load Level and Auto Load Mode: Set to a level's folder name to bypass the menu, going from the loading screen directly into singleplayer or the level editor.

Glazier: Overrides default *Glazier*.

151.5 Troubleshooting

Check Unity's log files. On Windows there's a shortcut in the project folder to the most recent log file `Unity Editor.log` as well as the containing folder `UnityEditor Logs Folder`.

151.6 File Organization

`Assets/Game/Sources` contains all of the source (e.g., `.blend`) and imported (e.g., `.fbx`) files for Unity assets exported in the asset bundle.

`Assets/Resources` are Unity assets loaded by the `Resources` class. Introducing new files to this folder should be avoided if possible.

`Assets/Runtime` contains all of the player code. Certain newer features have their own folders per-assembly-definition, but most game code is in the `Assembly-CSharp` folder. It would be nice to rename it, but as far as I'm aware we can't do this without breaking script references in asset bundles (as of 2024-10-18).

`Assets/Runtime/Assembly-CSharp/NetGen` is all generated networking code. It's included in Git to make the first-run process smoother.

The `Builds` folder contains exported Unity players.

151.7 Net Code

Most gameplay requires remote procedure calls (RPCs) to function properly. Even singleplayer is essentially a one-player server. The RPC code is automatically generated, but this is still a manual step in the editor:

1. Open **Window > Unturned > Net Gen**
2. Click **Generate**
3. Tab out and back in to ensure the scripts are imported

151.8 Continuous Integration

For each commit, we have set up Jenkins to build the project and runs tests, optionally uploading to a Steam branch.

At the time of writing (2024-10-18) the Jenkins server is locally hosted and not accessible over the Internet.

It **mostly** works with Pipelines using a script at `Build_Scripts/Jenkinsfile.txt`.

Launching the correct version of Unity relies on `Build_Scripts/JenkinsBootstrapper.exe` built from `JenkinsBootstrapper` in the project root. It expects Unity to be installed in one of these paths:

- `C:\UnityEditors`
- `C:\Unity Editors`
- `C:\Program Files\Unity\Hub\Editor`

(Yeah, sadly the development and build processes are very Windows-centric.)

CHAPTER

TWO

LEGACY ID AVAILABILITY

Ideally, new content should use GUIDs when possible.

Certain asset types—such as items—still rely heavily on legacy IDs. That said, they can be referenced by GUID in many cases. For example, NPC clothing and item spawn tables can reference items by GUID.

To find available legacy IDs:

1. Press `F1` in the main menu **Workshop** sub-menu.
2. Click **Export Asset IDs**.
3. Open `Extras/AssetIDs/All Assets/Grouped by Legacy Category`.
4. Each legacy asset category (e.g., Items) has a corresponding `Legacy ID Availability.csv` file listing IDs and whether they are reserved for core content.

Welcome to the official documentation for Unturned's modding and server hosting features! To navigate, use the table of contents in the sidebar or the search function in the top-left corner.

CHAPTER

THREE

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 may pop up take priority over the roadmap, so it may go a while between progress updates. Several high-priority ideas that don't yet have a solid plan, like a crafting revamp, are not listed on the board.

CHAPTER

FOUR

LEGACY TUTORIALS

Our earliest modding documentation was available through Steam Guides. Although most of these guides have been superseded by our documentation here, some still contain useful information about Unity setup that we have yet to bring over. The useful guides are:

- Items: Most of the sub-sections about Unity setup are still relevant.
- Vehicles: Information about Unity setup.
- Objects: Information about Unity setup.
- Misc. Assets: Unity setup for animals, effects, and resources.
- Maps: Explaining various features.
- Localization
- Creating 3rd-party Modules: Not to be confused with *mods* or *plugins*.
- Uploading Mods to the Steam Workshop
- Upgrading from Unity 5 LTS to Unity 2017 LTS: Only relevant for *very* old mods, but lists the important changes between those two versions.

There are several official video tutorials available, but many of these have gradually become outdated. They do not accurately represent the current modding features available, or may contain incorrect information, but are listed here for completeness:

- Hosting a Dedicated Server on Windows
- 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

If you refer to a video tutorial (official or otherwise), we recommend double-checking the information with our written documentation when possible.

CHAPTER

FIVE

OFFLINE DOWNLOADS

PDF and Epub versions of the documentation can be downloaded for offline use.

CHAPTER

SIX

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.