



nullsave

# INVENTORY COG

Version 1.12 – Released October 2020

In-depth Tutorial Series:

[https://www.youtube.com/watch?v=P6x02ldUI94&list=PLjKOAiXvjF-Vji1h\\_y12gAWRbBE4hYjuo](https://www.youtube.com/watch?v=P6x02ldUI94&list=PLjKOAiXvjF-Vji1h_y12gAWRbBE4hYjuo)

## Contents

Introduction .....	5
Equip Points.....	5
Skinned Equip Points.....	6
Skinned Bone Remapper .....	6
Store Points .....	7
Attach Points.....	7
DLC Support.....	8
Creating an Asset Bundle .....	8
Retrieving your DLC .....	8
Additional Notes .....	9
Inventory DB .....	10
Item Tags.....	10
Item Categories .....	11
Inventory Items.....	12
Creating a Loot/Drop Item .....	17
Loot Item .....	17
Loot Item With UI .....	18
Creating Equip Objects .....	18
Creating Item Previews .....	19
Inventory Cog™.....	19
Inventory Containers.....	22
Inventory Merchants.....	23
Crafting.....	24
Crafting Category .....	24
Crafting Recipes.....	24
UI Components.....	26
Menu Components .....	26
Container Menu UI.....	26
Inventory Menu UI.....	27
Item Menu UI.....	27
Item Menu Option .....	27
Item Menu Client.....	28

Merchant Menu UI.....	29
Pickup Menu UI.....	29
Count Select UI .....	30
List Components.....	30
Category List.....	30
Crafting Category List .....	31
Inventory Component List.....	32
Inventory Item List .....	32
Inventory Item Grid .....	32
Inventory Item Scroll List.....	34
Recipe List.....	35
Recipe Item Grid .....	36
Recipe Scroll List.....	37
Inventory Message List Client.....	39
Message List.....	39
Item Elements .....	40
Category UI.....	40
Component UI .....	40
Crafting Category UI .....	41
Equip Point UI.....	41
Item Detail UI .....	42
Item Preview UI.....	43
Item Tag UI.....	43
Item UI.....	44
Page Indicator UI .....	45
Rarity Color Indicator .....	45
Slot Item UI .....	46
Stat Modifier UI Editor.....	47
Recipe Component UI .....	47
Recipe UI .....	48
Quick API Reference.....	49
Change Log.....	51



## Introduction

Inventory Cog™ is a collection of components designed to make creating a full functioning inventory system with Unity quick and easy. This system provides support for equipping/storing, looting/dropping, repairs, breakdowns, upgrades, crafting, storing, and trading/merchants. There is also a host of UI and support elements to help you provide a robust experience to your users. *All categories and items are instantiated per character and support multiple characters/players.*

When combined with Stats Cog™ items can also directly affect your character's stats when equipped or consumed.

## Equip Points



In order to place physical copies of weapons/armor on to your character(s) you will need to Equip Points to locations where you would like them to appear.

These points are visualized by Gizmos as spheres.

If the character is using an armature, make sure to create these equip points on the appropriate bone transform so it will move with the character during animations.

Equip Points are designed for non-skinned mesh objects.

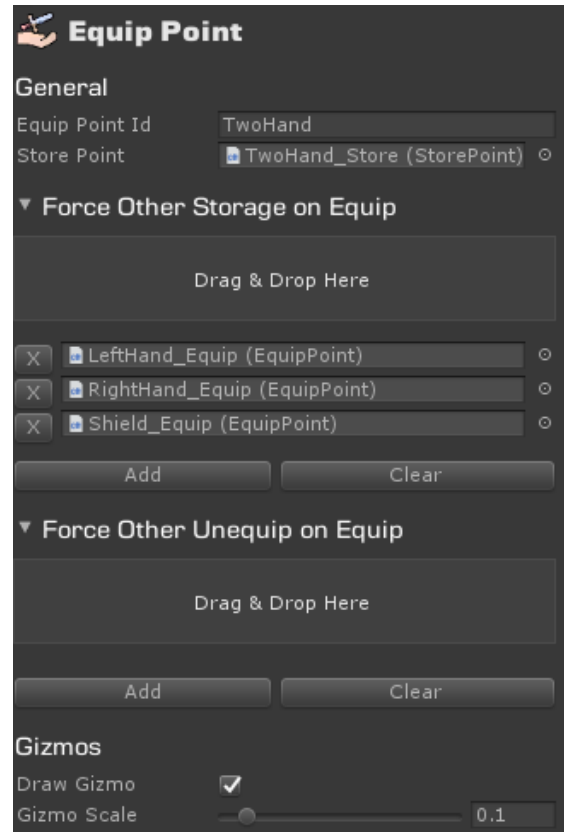
**Equip Point Id** – String name for the equip point. Strings are used to allow items to be easily swapped between characters

**Store Point** – Object reference to a store point on the same character, if supplied items will moved to this location when stored

**Force Other Storage on Equip** – List of equip points that should automatically store any equipped items when an item is equipped to this point

**Force Other Unequip on Equip** – List of equip points that should automatically unequip any items they have when an item is equipped to this point.

Note on **Force Other** options: This exists for cases where you might have a 2-handed item or dual-wielding. Example a shield in the right hand needs to be stored or unequipped to put a 2-handed weapon in the left hand.



## Skinned Equip Points

**Equip Point (Skinned)**

**General**

Equip Point Id: HairSkinned

Store Point: None (Store Point)

**Skinning**

Bone Source: HairType1 (Skinned Mesh Re)

Default Skin: HairType1 (Skinned Mesh Re)

▼ **Force Other Storage on Equip**

Drag & Drop Here

X Head\_Equip (EquipPoint)

Add Clear

▼ **Force Other Unequip on Equip**

Drag & Drop Here

Add Clear

Skinned Equip Points are designed to handle replacing skinned mesh components on your character for things such as armor, hair, etc. Usually this will reside at the root of your armature and can be seen in the demo scenes.

**Equip Point Id** - String name for the equip point. Strings are used to allow items to be easily swapped between characters

**Store Point** - Object reference to a store point on the same character, if supplied items will moved to this location when stored

**Bone Source** - Provides a bone source for mapping the replacement skinned mesh

**Default Skin** - Original skinned mesh to deactivate while this item is equipped

**Force Other Storage on Equip** - List of equip points that should automatically

store any equipped items when an item is equipped to this point

**Force Other Unequip on Equip** - List of equip points that should automatically unequip any items they have when an item is equipped to this point.

## Skinned Bone Remapper

**Skinned Bone Remapper**

**Behaviour**

Root Bone:

Bone Names

List is Empty

+ -

Automap

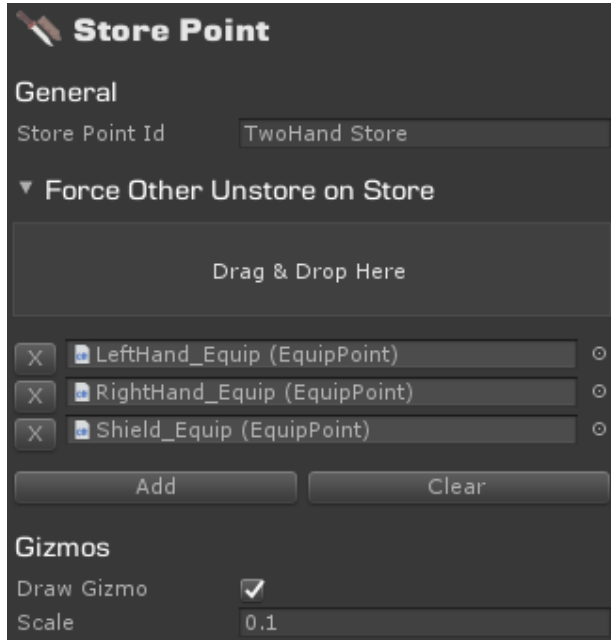
Clear

If your new item does not share every bone named the same and in the same order as on your armature you can add a Skinned Bone Remapper to the equippable object. Pressing “Automap” will grab the bones and populate them for you. You may do this manually as well if you desire. If you need to change names you can also do so here after mapping.

Note on skinned meshes: If you delete the armature (which is acceptable) on your item it will not render at design time, it will still show up at runtime when mapped.

## Store Points

Store points represent physical locations on your character to store items that were equipped but are not currently active. These are represented by Gizmos as cubes (see image in Equip Points for reference).

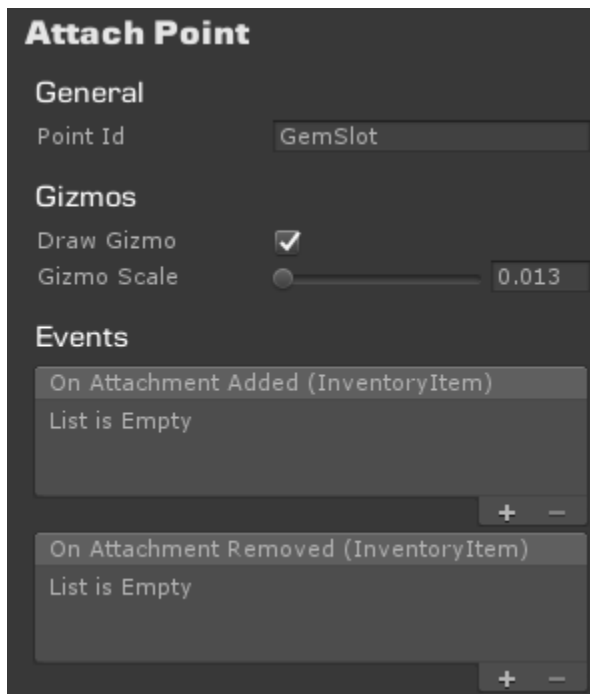


**Store Point Id** – String value used to reference this store point

**Force Other Unstore on Store** – List of Equip Points that should remove any items from their Store Point and restore them to an Equipped state.

Note on **Force Other** option: As with the force options in Equip Points this is designed to handle things such as two-handed weapons. In this example let's say there is a bow equipped with a sword and shield stored. When the bow is stored the sword and shield will automatically become equipped again.

## Attach Points



Attach points serve to allow you to physically add one or more attachments to an object. Examples include a gemstone, scope, silencer, etc.

**Point Id** – String value used to reference this store point

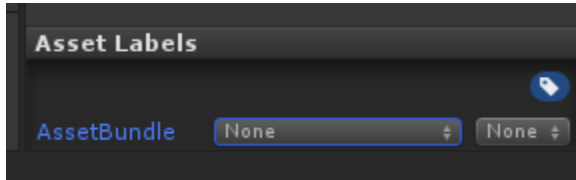
**onAttachmentAdded** – Raised when an attachment is added to the slot

**onAttachmentRemoved** – Raised when an attachment is removed from the slot.

## DLC Support

Starting with version 1.10 Inventory Cog has full support for DLC (downloadable content). In order to setup DLC packages and use them in your game there are only a couple of steps you need to follow.

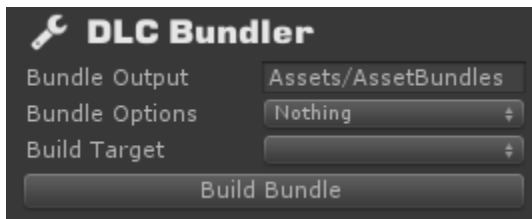
### Creating an Asset Bundle



Items used specifically for Inventory Cog should be in a bundle separate from other DLC. You can assign any folder or item to a bundle by selecting the “Asset Labels” tab in the Inspector and creating or assigning a

bundle to it.

To actually compile your Asset Bundle a tool has been included for your use. Simply create a new Game Object and add the DLC Bundler component.



**Bundler Output** – Path where the bundle will be created.

**Bundle Options** – Options used when creating the bundle (covered in Unity API)

**Build Target** – Target to use when building the bundle.

### Retrieving your DLC

Once your bundle(s) have been created and uploaded you will need to download and consume them. Another simple component has been created for this. Somewhere prior to your first playable level create a new Game Object and add the Inventory DLC Updater component.



**Inventory DLC**

**Behaviour**

Target Platform: Desktop

Host:

Bundle Files

Add Clear

Include Manifest File ☒

Buffer Kb: 100

**UI**

Progress: None (Slider)

**Events**

On DLC Update (Int32, Int32, Int64, Int64)  
List is Empty

On DLC Complete ()  
List is Empty

**Target Platform** – This is used to determine which platform the DLC is valid for. You can have multiple instances of the Inventory DLC Updater (one for each platform you support) and only the one set for the active platform will update.

**Host** – Top-level URL for accessing the bundle files

**Bundle Files** – List of files to retrieve from the host

**Include Manifest File** – If checked for each file specified the updater will also try and retrieve a “.manifest” file from the same location

**Buffer Kb** – Size of the download buffer in Kb

**Progress** – Slider to use display update progress (optional)

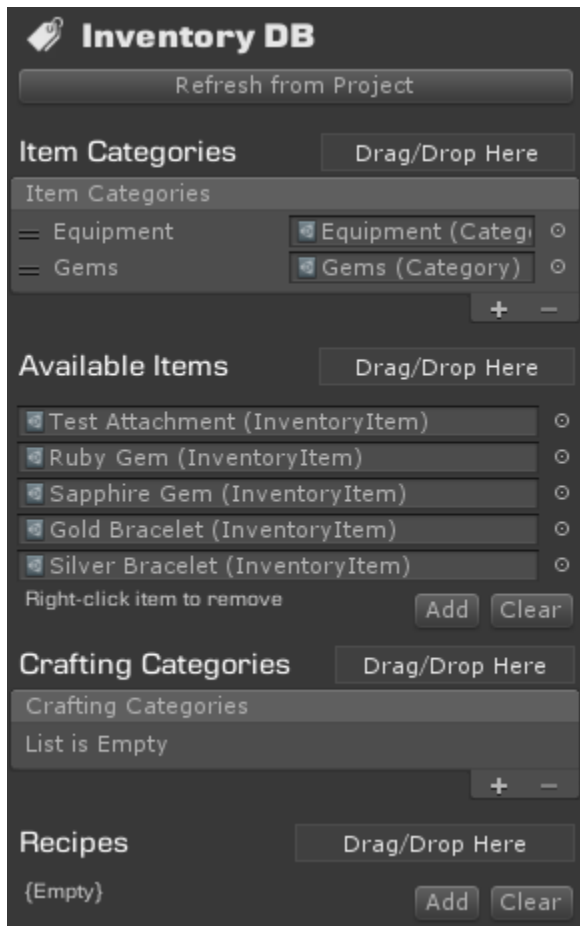
**onDLCUpdate** – Fired each time data is read

**onDLCComplete** – Fired when all updates are complete

### Additional Notes

For best results working with DLC it is recommended that **all** Inventory Cog related assets (GameObjects, Recipes, Categories, Items, etc) are placed in a **Resources** folder. It is also recommended that you remove any copies of the Inventory DB and allow it to load dynamically.

## Inventory DB



The Inventory DB is a top-level MonoBehaviour that keeps track of globally available items such as Categories (Items and Crafting), Recipes, and a global list of all items in your game.

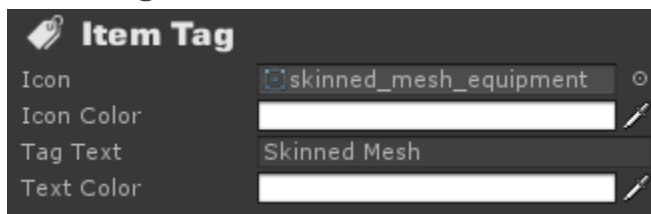
To begin, create a new Empty GameObject in your scene and add the “Inventory DB” component. This component will mark your object as “DontDestoryOnLoad” and will follow from scene to scene. Feel free to put this object in multiple scenes for development as they will simply override each other.

As you go through creating your items, categories and recipes do not forget to add them to your Inventory DB. While items can be passed without placing them in the Inventory DB they will not appear globally and risks errors.

The items available for placing here are covered in detail in later sections of the documentation.

**Refresh from Project** – This button will load all items, categories and recipes found in your project (including from the demos if you have not deleted them).

## Item Tags

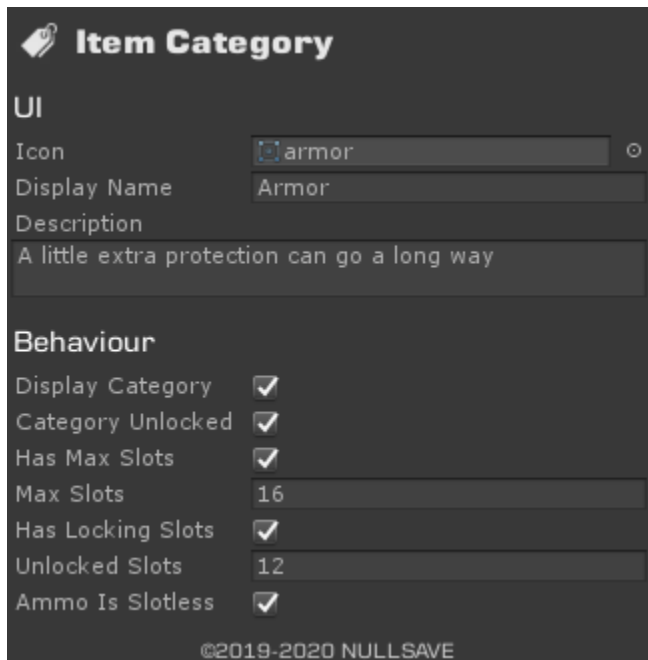


Item tags provide visual tags for Inventory Items and can be created by right-clicking in your “Project” pane and selecting Create/TOCK/Inventory/Item Tag. These tags can provide Icons and Text for an inventory item with unique

colors being applied to each.

## Item Categories

Item Categories are required by the Inventory Cog™ to sort your items appropriately. In some cases, you may not need to separate items into groupings, in this event it is still necessary to create a single Item Category to contain all your items. Item Categories are created by right-clicking in your “Project” pane and selecting Create/TOCK/Inventory/Item Category.



**Icon** – Sprite asset to associate with your category

**Display Name** – Name to display for this category

**Description** – Descriptive details for category

**Display Category** – Determines whether the category should be displayed in various UI elements (checked by default)

**Category Unlocked** – Determines if the category is unlocked. Items associated with this category will not be lootable unless this option is checked (checked by default)

**Has Max Slots** – When checked this option limits the number of slots or spaces that can be used in this

category

**Max Slots** – Sets the maximum number of items in this category allowed at one time in inventory

**Has Locking Slots** – Checking this box will allow you to have slots appear as “locked” and unusable (useful for upgrade/unlock behaviors)

**Unlocked Slots** – Sets how many slots are unlocked.

**Ammos Is Slotless** – If checked this option will not count Inventory Items marked as “Ammo” against the slot count

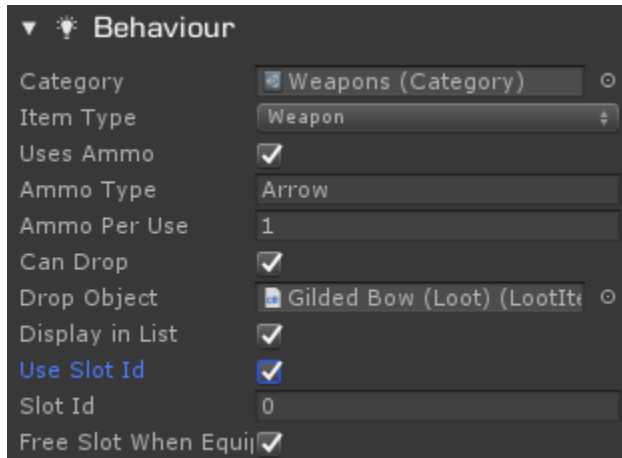
Item Categories are instantiated to each instance of Inventory Cog™ and do support having different values on different characters. These values are automatically saved by Inventory Cog™ when you invoke the `InventoryStateSave` method.

When Stats Cog™ is installed additional options will be available to set certain items as either “Static” values (normal) or based on Stat Values / Stat Expressions. These will operate in the same way as Stat Values & Expressions defined in Stats Cog™ documentation.

## Inventory Items

The heart of Inventory Cog™ is Inventory Items. These items can be equipped, stored, dropped, looted, bought and sold. If you have Stats Cog™ installed as well, they can even modify your character's stats.

These items are extremely versatile, and the editor is broken down into collapsible sections. We will cover those sections individually below.



**Category** – Scriptable Object reference which determines what category this item belongs to

**Item Type** – Specifies the type of item this is options include: Consumable, Weapon, Armor, Shield, Ammo, Ingredient, Quest Item, Journal, Component, Attachment and Container. (note: Weapon, Consumable, Ammo, Attachment and Container are special types that affect how the item is used. All other times are for

your use and do not affect behavior)

**Uses Ammo** – Sets if the item requires ammo (only available for Weapon item type)

**Ammo Type** – String value denoting type of ammo to use (for Weapon item type) or type of ammo to be (for Ammo item type)

**Ammo Per Use** – Specifies how much ammo is required for each use

**Can Drop** – Determines if the item can be dropped once looted (useful for things like quest items)

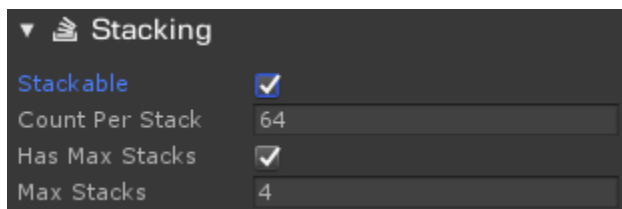
**Drop Object** – Prefab to instantiate when the item is dropped

**Display in List** – Determines if the item should be displayed in the inventory UI

**Use Slot Id** – Not specifically used by Inventory Cog™ but available for your use and consumed by Slot Item UI

**Slot Id** – Specifies which Slot Id to use

**Free Slot When Equipped** – If checked the item will not display in the inventory or count against maximum slots when equipped to the player



\* Not available for Container item types

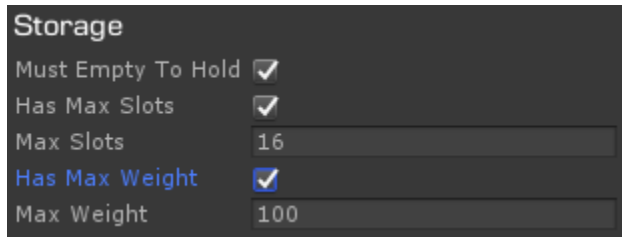
**Stackable** – Determines if the item can be stacked into a single slot

**Count Per Stack** – Sets the maximum number that can be in a single stack

**Has Max Stacks** – If checked only a

certain number of stacks will be allowed at any one time

**Max Stacks** – Sets the maximum number of stacks allowed



**Storage**

Must Empty To Hold ☒

Has Max Slots ☒

Max Slots 16

Has Max Weight ☒

Max Weight 100

\*Only available for Container item types

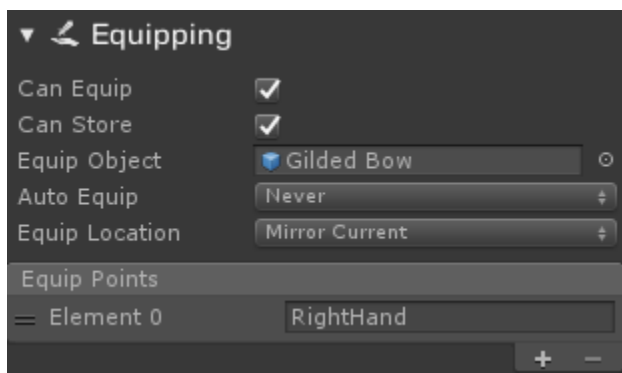
**Must Empty To Hold** – If checked the item must not be storing any other items to be added to the inventory

**Has Max Slots** – If checked the number of slots available in the container will be limited

**Max Slots** – Sets the maximum number of slots for the container

**Has Max Weight** – If checked the item will only be able to contain up to a certain weight

**Max Weight** – The maximum weight allowed in the container



**Equipping**

Can Equip ☒

Can Store ☒

Equip Object Gilded Bow

Auto Equip Never

Equip Location Mirror Current

Equip Points

Element 0 RightHand

\*Not available for Attachment items

**Can Equip** – If checked the item can be physically equipped to the character

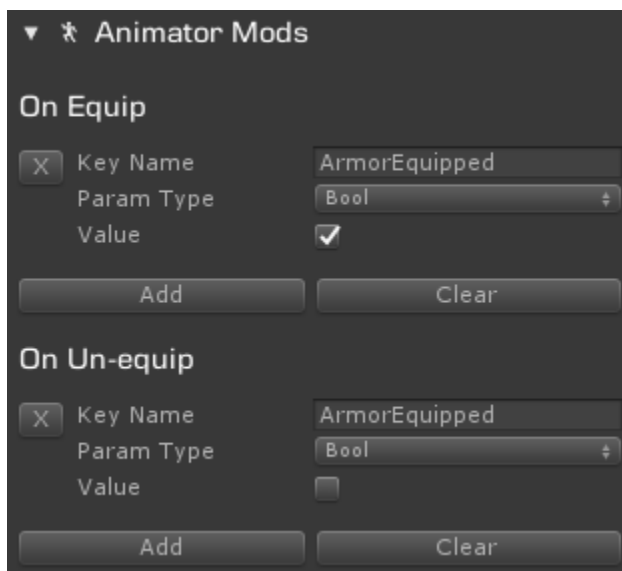
**Can Store** – Determines if, once equipped, the item can be stored

**Equip Object** – Prefab of the item to equip to the character

**Auto Equip** – Sets how to automatically equip the item when looted (Never, If Slot Free, Always)

**Equip Location** – Sets where to auto equip looted item (Mirror Current, Always Equip, Always Store)

**Equip Points** – List of strings corresponding to equip points on the character the item can be equipped. If there are more than one Inventory Cog™ will use the first free slot (if any) or the first slot listed (if none are free)



**Animator Mods**

**On Equip**

Key Name	Param Type	Value
ArmorEquipped	Bool	<input checked="" type="checkbox"/>

Add Clear

**On Un-equip**

Key Name	Param Type	Value
ArmorEquipped	Bool	<input type="checkbox"/>

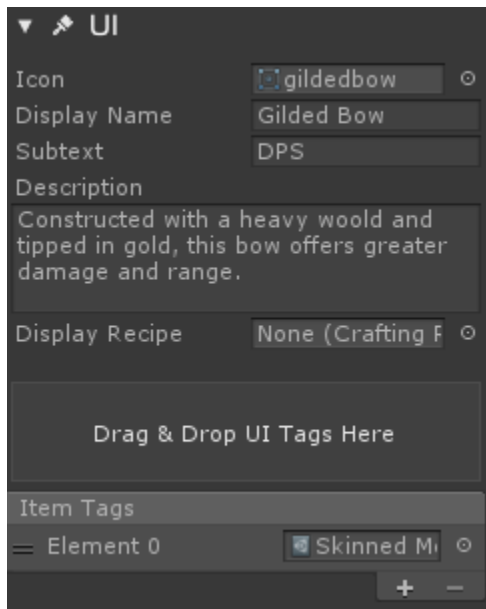
Add Clear

\*Not available for Attachment items

Sometimes you want to change the way your character moves, or acts based on what items they have equipped, Animator Mods helps do that.

**On Equip** – List of animator mods to set when the item is equipped to the character

**On Un-equip** – List of animator mods to set when the item is no longer equipped to the character



**Icon** – Icon to display for the item in the UI

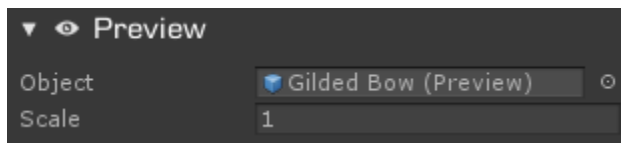
**Display Name** – Name to display in the UI for the item

**Subtext** – UI subtext for item

**Description** – Description of the item

**Display Recipe** – Recipe (if any) to display on UI in association with this item

**Item Tags** – Array of Item Tags for the item. These tags are reorderable and will be displayed in the order they appear here.

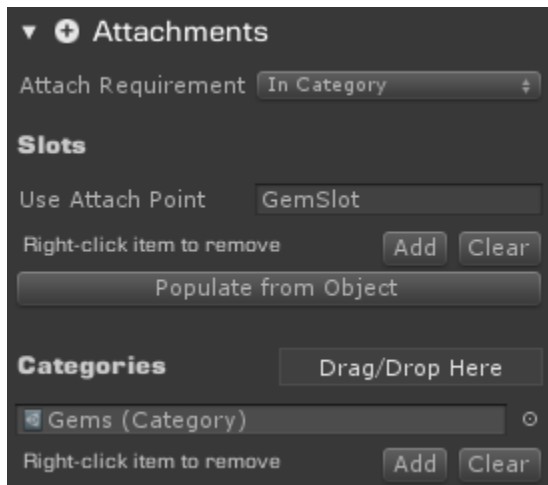


In some inventory UIs you may wish to display a preview model to the player, this section is created for that purpose.

**Object** – Prefab to instance when

previewing the item

**Scale** – Scale to set the object to in preview



\*Not available for Attachment items

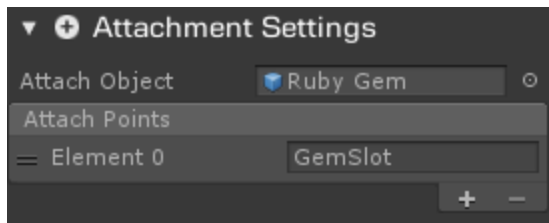
**Attach Requirements** – Requirements to use to allow an attachment to be added to this item (None Allowed, In Category, In Item List)

**Slots** – List of attachment slots available for item. The “Use Attach Point” is per slot and tells the slot where to physically equip the item (if anywhere)

**Populate from Object** – Automatically creates a slot for each Attach Point in the Equip Object

**Categories** – List of categories an attachment item can be in

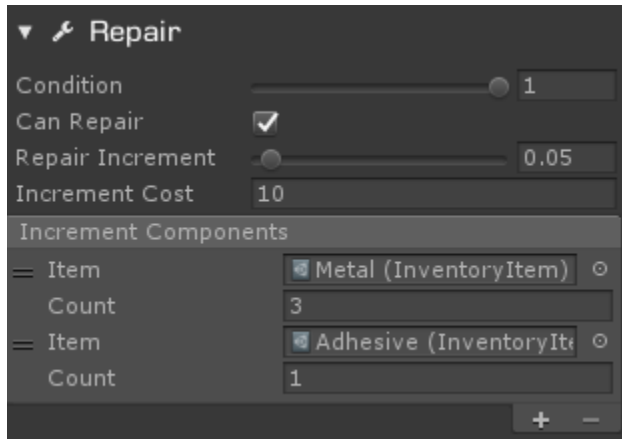
**Items** – List of specific attachments that are allowed on this item



\* Only available for Attachment items

**Attach Object** – Physical Game Object to attach to the item (if any)

**Attach Points** – A list of points the attachment can be added to



**Condition** – The default condition level for the item

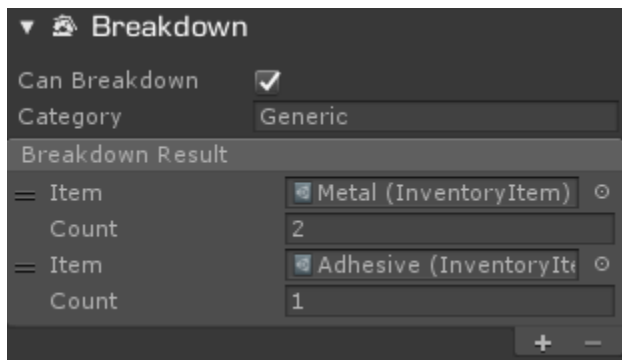
**Can Repair** – If checked, the item can be repaired when the condition is less than 1

**Repair Increment** – Amount of condition to restore with each usage of Cost/Components

**Increment Cost** – Cost associated with repairing the item 1 increment (not currently used but available for you)

**Increment Components** – A list of other Inventory Items (and their count) needed

to repair this item 1 increment. (Inventory Cog™ will automatically repair an item as much as possible based on the player's current inventory)



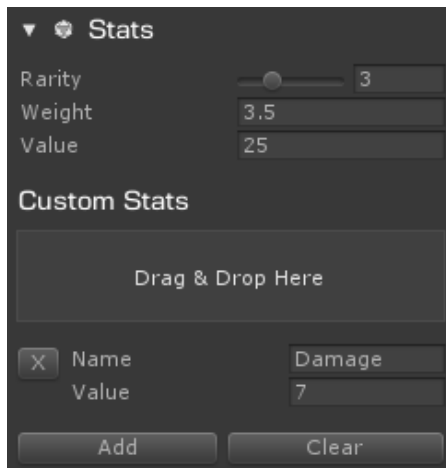
The Breakdown section allows you to provide a way for the player to break an item down into other components.

**Can Breakdown** – Determines if the item can be broken down

**Category** – String representing the category of breakdown (useful for filtering weapon/armor/etc)

**Breakdown Result** – A list of other Inventory Items (and their count) that

will be added to the player's inventory when this item is broken down



▼ **Stats**

Rarity

Weight

Value

**Custom Stats**

Drag & Drop Here

X Name

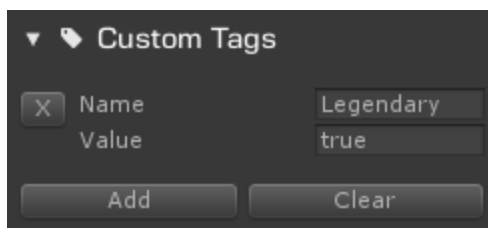
Value

**Rarity** – Base rarity for the item

**Weight** – Base weight for the item

**Value** – Base value for the item

**Custom Stats** are name/value pairs that allow you to store a collection of float values with unique names for your own usage.

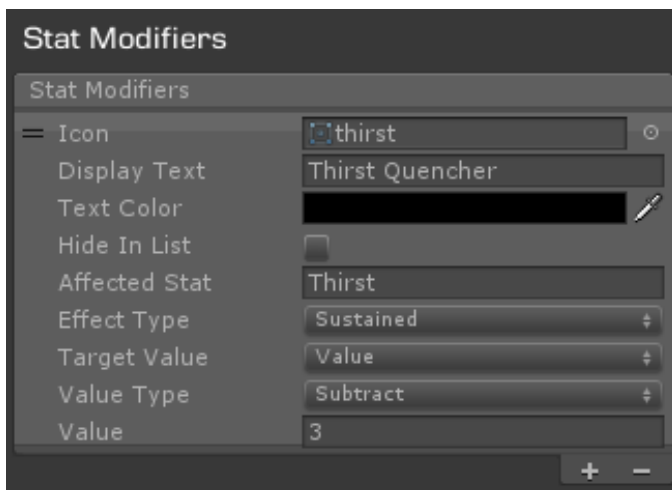


▼ **Custom Tags**

X Name

Value

Introduced with version 1.4 **Custom Tags** can be used to filter items in your lists and for any custom features you wish to add. These tags are saved/loaded with the rest of the inventory.



**Stat Modifiers**

Stat Modifiers

= Icon

Display Text

Text Color

Hide In List ☐

Affected Stat

Effect Type

Target Value

Value Type

Value

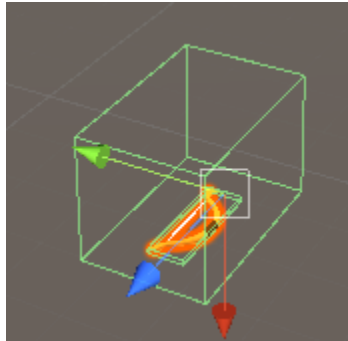
If you do not have the **Stats Cog** installed this section will not appear for you. With the **Stats Cog** installed you can apply sustained stat changes for equipped items (such as attack, or defense increases) or instant stat effects for items you use/consume such as increasing health when eating food.

Stat Modifiers are covered in detail in the **Stats Cog** documentation.



## Creating a Loot/Drop Item

Most items will need the ability to be picked up from the ground or dropped to the ground by the player from their inventory. To do this we will create a prefab which we can place in the “Drop Object” of an Inventory Item covered in the previous section.



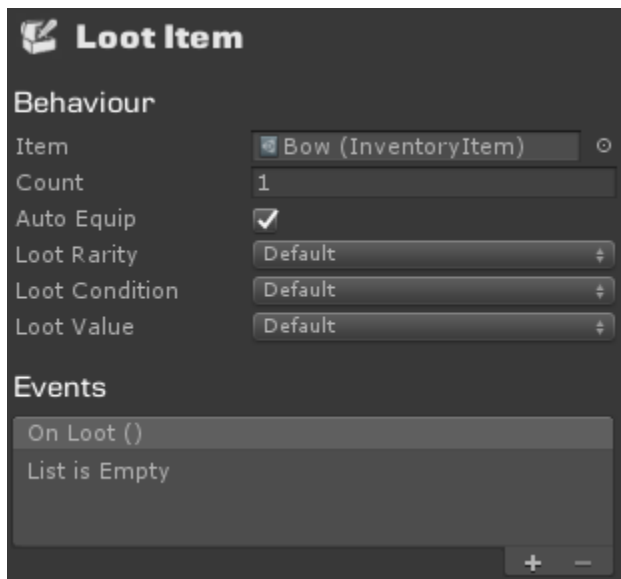
To begin, drag your model into the scene. Add any Rigidbody and colliders you wish.

If you're using a collider to pick up your object add a trigger collider as well. (Seen left as the larger green bounding box)

Next add either a Loot Item or a Loot Item With UI component to your model.

Once setup you can drag the object into a prefab and assign to the Inventory Item “Drop Object” if desired.

## Loot Item



**Item** – Reference to Inventory Item to add when looted

**Count** – Number of the item to add to inventory

**Auto Equip** – If **not** checked the item will not be auto equipped even if the Inventory Item's Auto Equip is set to If Slot Free or Always

**Loot Rarity** – Determines what rarity to assign when looted (Default, Constant, Random Between Constants)

**Loot Condition** – Determines what condition to assign when looted (Default, Constant, Random Between Constants)

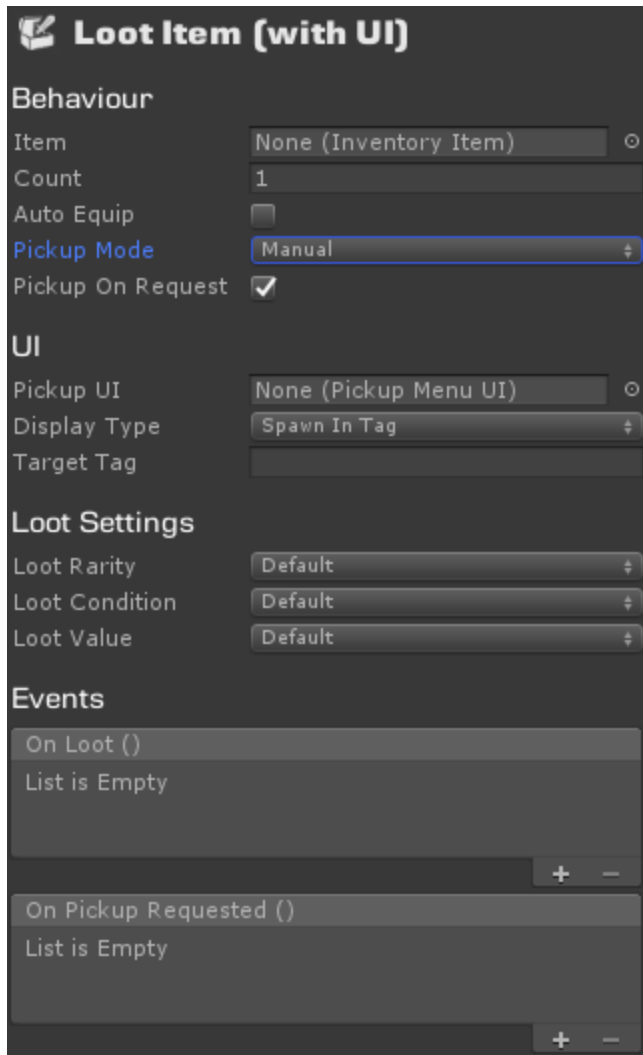
**Loot Value** – Determines what value to

assign when looted (Default, Constant, Random Between Constants)

**OnLoot** – Event fired when item is looted

## Loot Item With UI

Extended from the Loot UI component, this component allows you to specify custom UI options for your items.



**Item** – Reference to Inventory Item to add when looted

**Count** – Number of the item to add to inventory

**Auto Equip** – If **not** checked the item will not be auto equipped even if the Inventory Item's Auto Equip is set to If Slot Free or Always

**Pickup Mode** – Sets how to pick up the object (Manual, By Button, By Key)

**Pickup Button** – Button to monitor for pickup (By Button only)

**Pickup Key** – Button to monitor for pickup (By Key only)

**Pickup On Request** – Adds to inventory immediately on request if checked, otherwise added is done manually

**Pickup UI** – Prefab or GameObject reference of a Pickup Menu UI to show for this item.

**Display Type** – Sets how to display the prompt (Activate Game Object, Spawn On First Canvas, Spawn In Transform, Spawn In Tag)

**Parent Transform** – Transform to place prefab in (Spawn In Transform only)

**Target Tag** – Name of tag to find and place prefab in (Spawn In Tag only)

**On Loot** – Event fired when item is looted

**On Pickup Requested** – Event fired when pick up of item is requested

## Creating Equip Objects

In order to physically equip an object to your character you will need a prefab to use when equipping. No special components are needed for this but there are a few tricks that may come in handy for getting your item positioned just right.

To begin with create an Empty Game Object in your scene and name it as desired. For this example, it will be referred to as NewEquipObject. Next add your model as a child, which we will refer to as BaseModel.

With your character and the object in the scene press play. Once your scene has started, press the pause button and return to the scene view.

Now drag NewEquipObject into whichever Equip Point you wish to use. Unity will automatically adjust the scaling. Most likely the item will not be positioned exactly how you want it.

Adjust the BaseModel (not NewEquipObject) until it is positioned and rotated as desired. Once complete you can drag the NewEquipObject to create a prefab.

The object is now setup and can be assigned to the Inventory Item “Equip Object”.

### Creating Item Previews

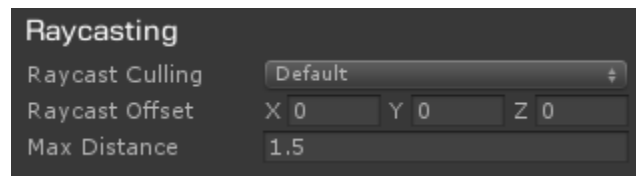
Item previews are used in the UI only and don’t require any special components added to them. Setup is quite like Equip Objects, placing the model inside of an Empty Game Object parent.

To adjust Item Previews, open the preview at run time and adjust the child offset as needed to center in your view.

## Inventory Cog™

The Inventory Cog™ component is the heart of the Inventory Cog™ system and should be added to every player and non-player character that needs to track inventory. As mentioned in the introduction all objects (even the Scriptable Objects) are instantiated per use and cleanly support multiple characters without any additional setup. These inventories can all be saved and loaded individually.

Like Inventory Items, the Inventory Cog™ is broken down into sections for easy display and use.

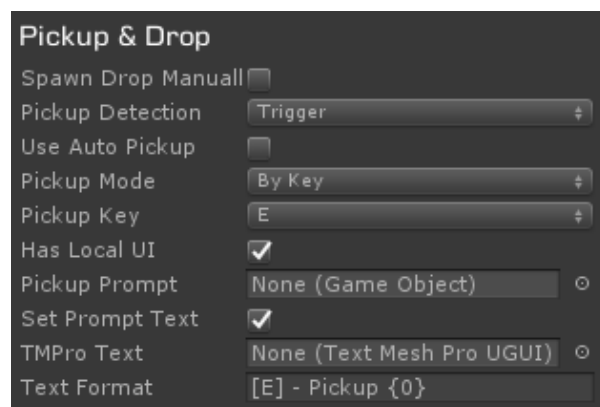


**Raycast Culling** – Sets which layer(s) should be used when raycasting

**Raycast Offset** – Offset from raycast source to use

**Max Distance** – Maximum distance from

character (not source)



**Spawn Drop Manually** – When checked Inventory Cog™ will **not** spawn a drop object, drop will need to be spawned manually. Otherwise the drop is spawned 2 units forward and 1.5 units up from the characters transform position

**Pickup Detect** – Determines how to detect items for pickup/loot (Trigger or Main Cam Raycast)

**Pickup Mode** – Sets how to trigger a pickup (Manual, By Button, By Key)

**Pickup Button** – Button to monitor for pickup (By Button only)

**Pickup Key** – Key to monitor for pickup (By Key only)

**Has Local UI** – Determines if there is an in-scene GameObject to display pick up prompts

**Pickup Prompt** – Reference to GameObject that displays the prompt

**Set Prompt Text** – If checked Text will be updated as defined

**TMPro Text** – Text object to update

**Text Format** – Format to use when populating text {0} is replaced with the item's Display Name



**Menu Mode** – Sets how to open the inventory menu (Manually, By Button, By Key)

**Menu Button** – Button to monitor to open menu (By Button only)

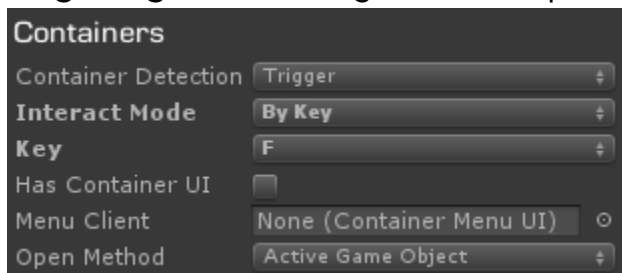
**Menu Key** – Key to monitor to open menu (By Key only)

**Menu Client** – GameObject to activate or Prefab to instance when opening menu

**Open Method** – Sets how to open the menu (Activate Game Object, Spawn On First Canvas, Spawn In Transform, Spawn In Tag)

**Parent Transform** – Transform to place prefab in (Spawn In Transform only)

**Target Tag** – Name of tag to find and place prefab in (Spawn In Tag only)



**Container Detection** – Determines how to detect items for pickup/loot (Trigger or Main Cam Raycast)

**Interact Mode** – Sets how to trigger a container (Manual, By Button, By Key)

**Button** – Button to monitor for opening container (By Button only)

**Key** – Key to monitor for opening container (By Key only)

**Has Local UI** – Determines if there is an in-scene GameObject to display container prompts

**Open Prompt** – Reference to GameObject that displays the prompt

**Set Container Text** – If checked Text will be updated as defined

**TMPro Text** – Text object to update

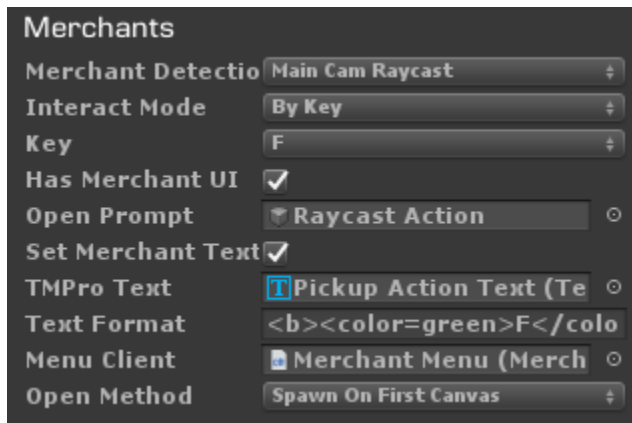
**Text Format** – Format to use when populating text {0} is replaced with the container's Display Name

**Menu Client** – GameObject to activate or Prefab to instance when opening menu

**Open Method** – Sets how to open the menu (Activate Game Object, Spawn On First Canvas, Spawn In Transform, Spawn In Tag)

**Parent Transform** – Transform to place prefab in (Spawn In Transform only)

**Target Tag** – Name of tag to find and place prefab in (Spawn In Tag only)



**Merchant Detection** – Determines how to detect items for pickup/loot (Trigger or Main Cam Raycast)

**Interact Mode** – Sets how to trigger a merchant (Manual, By Button, By Key)

**Button** – Button to monitor for opening merchant (By Button only)

**Key** – Key to monitor for opening merchant (By Key only)

**Has Local UI** – Determines if there is an in-scene GameObject to display

container prompts

**Open Prompt** – Reference to GameObject that displays the prompt

**Set Merchant Text** – If checked Text will be updated as defined

**TMPro Text** – Text object to update

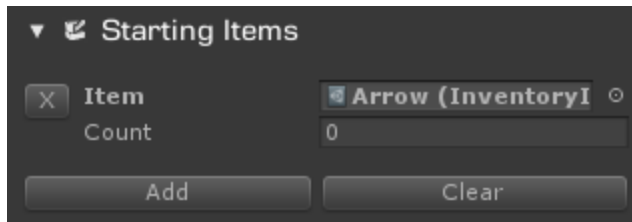
**Text Format** – Format to use when populating text {0} is replaced with the merchant's Display Name

**Menu Client** – GameObject to activate or Prefab to instance when opening menu

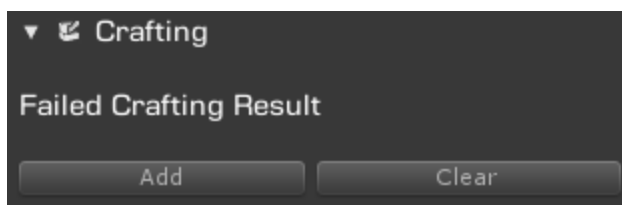
**Open Method** – Sets how to open the menu (Activate Game Object, Spawn On First Canvas, Spawn In Transform, Spawn In Tag)

**Parent Transform** – Transform to place prefab in (Spawn In Transform only)

**Target Tag** – Name of tag to find and place prefab in (Spawn In Tag only)



**Starting Items** – Contains a list of items (and their counts) that a player starts the game with



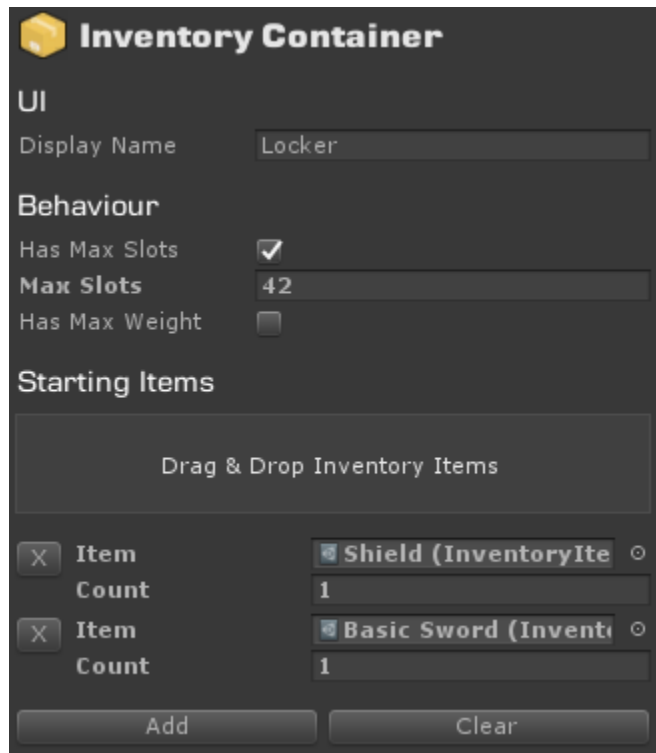
**Failed Crafting Result** – Contains a list of items (and their counts) to add to a player's inventory when they attempt to craft and there is no recipe that corresponds to the attempt. This is separate from when a player attempts to

craft a known recipe and fails.

Note: While playing a Debug view becomes available showing you all items and counts currently in inventory.

## Inventory Containers

Inventory Containers allow players to store or retrieve items from a container. Like Inventory Cog™ instances containers states can be saved and loaded.



The screenshot shows a configuration panel for an 'Inventory Container'. At the top is a yellow cube icon and the title 'Inventory Container'. Below this are three sections: 'UI', 'Behaviour', and 'Starting Items'. The 'UI' section has a 'Display Name' field with the value 'Locker'. The 'Behaviour' section has 'Has Max Slots' checked, 'Max Slots' set to 42, and 'Has Max Weight' unchecked. The 'Starting Items' section has a 'Drag & Drop Inventory Items' area and a list of items. The list contains two items: 'Shield (InventoryItem)' with a count of 1, and 'Basic Sword (InventoryItem)' with a count of 1. At the bottom are 'Add' and 'Clear' buttons.

Section	Property	Value
UI	Display Name	Locker
	Has Max Slots	✓
Behaviour	Max Slots	42
	Has Max Weight	✗
Starting Items	Item	Shield (InventoryItem)
	Count	1
Starting Items	Item	Basic Sword (InventoryItem)
	Count	1

**Display Name** – Name to display in the UI

**Has Max Slots** – When checked the container will have a maximum number of storage slots

**Max Slots** – Sets the maximum number of slots for a container

**Has Max Weight** – If checked the item will only be able to contain up to a certain weight

**Max Weight** – The maximum weight allowed in the container

**Starting Items** – Lists all items (and their counts) the container has at the beginning of the game

## Inventory Merchants

Inventory Merchants provide your player with a way to buy and sell goods in game.

**Inventory Merchant**

**Behaviour**

Display Name: General Merchant

Buy Modifier: 1

Sell Modifier: 1

Limit Vendor Currency: ☒

Currency: 1000

Stock Replenishes: ☒

Replenish Time: 3600

**Count Selection**

Use Count Selection: ☒

Min To Show Count: 3

Count Select UI: Buy Sell Count Selection

Count Container: Main Canvas (Rect Transf)

**Available Stock**

Drag Items Here

X Item	Adhesive (InventoryI	10
Count		
X Item	Apple (InventoryI	10
Count		
X Item	Arrow (InventoryI	32
Count		
X Item	Arrow (Fire) (Inve	32
Count		
X Item	Arrow (Ice) (Inver	32
Count		
X Item	Paper (Inventory	
Count		

Add Clear

**Display Name** – Name to display in UI

**Buy Modifier** – Amount to multiply item value by when buying from player

**Sell Modifier** – Amount to multiply item value by when selling to player

**Limit Vendor Currency** – When checked the merchant will start with a limited amount of currency

**Currency** – Amount of currency merchant starts with

**Stock Replenishes** – When checked stock will automatically replenish over time

**Replenish Time** – Second to wait before replenishing stock

**Use Count Selection** – When checked system will prompt user how many items to buy/sell

**Min To Show Count** – The minimum number of an item a user must be able to buy/sell in order to show count prompt (otherwise 1 item will be bought/sold each time)

**Count Select UI** – Prefab for selecting count

**Count Container** – Parent transform for count container (if any)

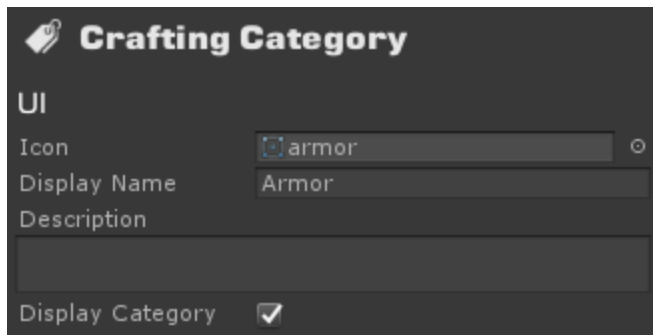
**Available Stock** – Default stock to start with (and replenish if set)

## Crafting

The crafting system built-in to Inventory Cog™ allows your players to combine objects to either upgrade an item or create new items. Crafting can be completed instantly or over time (both game and real-world time).

### Crafting Category

Like Inventory Items, Crafting Recipes need to have a category assigned to them for sorting and separating. If you aren't using multiple categories a single master category is still needed. You can create a category by right-clicking in the "Project" pane and selecting Create/TOCK/Inventory/Crafting Category.



**Icon** – Icon to display for category in UI

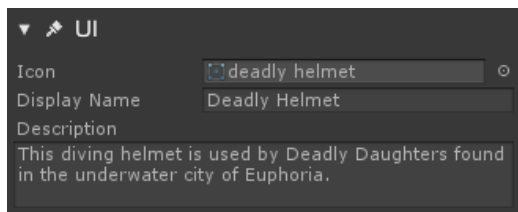
**Display Name** – Name to display in UI

**Description** – Description of category

**Display Category** – When checked the category will appear in category lists

### Crafting Recipes

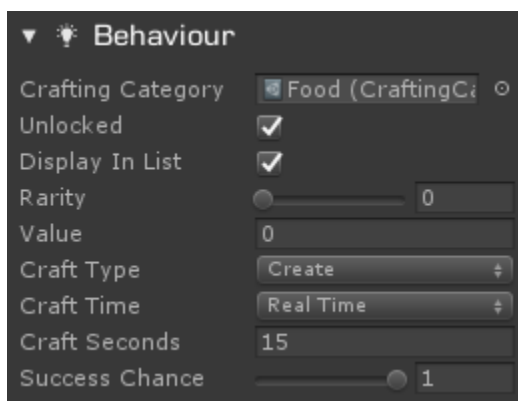
In order to craft an upgrade or new item a recipe is needed for Inventory Cog™. Like other larger items Crafting Recipe is divided into sections.



**Icon** – Icon to display in UI

**Display Name** – Name to display in UI

**Description** – Description of the recipe



**Crafting Category** – Category the recipe belongs to

**Unlocked** – Recipes cannot be crafted unless they are unlocked

**Display In List** – Sets if the recipe should be displayed in UI lists

**Rarity** – Rarity to assign to recipe

**Value** – Currency value of recipe

**Craft Type** – Type of recipe (Create or Upgrade)

**Craft Time** – Time used to complete crafting (Instant, Game Time, Real Time)

**Craft Seconds** – Seconds needed to complete crafting (Game Time and Real Time only)

**Success Chance** – Chance the recipe will craft successful (0 – never, 1 – always)



*This section only shows for “Upgrade” types*

**Base Item** – Item to be removed and upgraded

**Min Condition** – Minimum condition of item required to upgrade

**Max Condition** – Maximum allowed

condition of item to upgrade

**Min Rarity** – Minimum rarity of item required to upgrade

**Max Rarity** – Maximum allowed rarity of item to upgrade

*Called Additional Components for “Upgrade” types*

**Component Type** – Type of component reference to use (Standard/Advanced)

**Standard Components** – Lists items (and counts) required to complete recipe

**Advanced Components** – Lists items, counts, min/max condition and min/max rarity required to complete recipe

**Success Result** – Lists items to be added to inventory if the recipe is completed successful

**Item** – Inventory Item to add

**Count** – Number of the item to add to inventory

**Condition** – Change to apply to base condition (Unmodified, Average Of Components, Lowest of Components, Highest Of Components)

Highest Of Components)

**Rarity** – Change to apply to base rarity (Unmodified, Average Of Components, Lowest of Components, Highest Of Components)

**Value** – Change to apply to base value (Unmodified, Average Of Components, Lowest of Components, Highest Of Components)

When Stats Cog™ is installed additional options will be available to set certain items as either “Static” values (normal) or based on Stat Values / Stat Expressions. These will operate in the same way as Stat Values & Expressions defined in Stats Cog™ documentation.

## UI Components

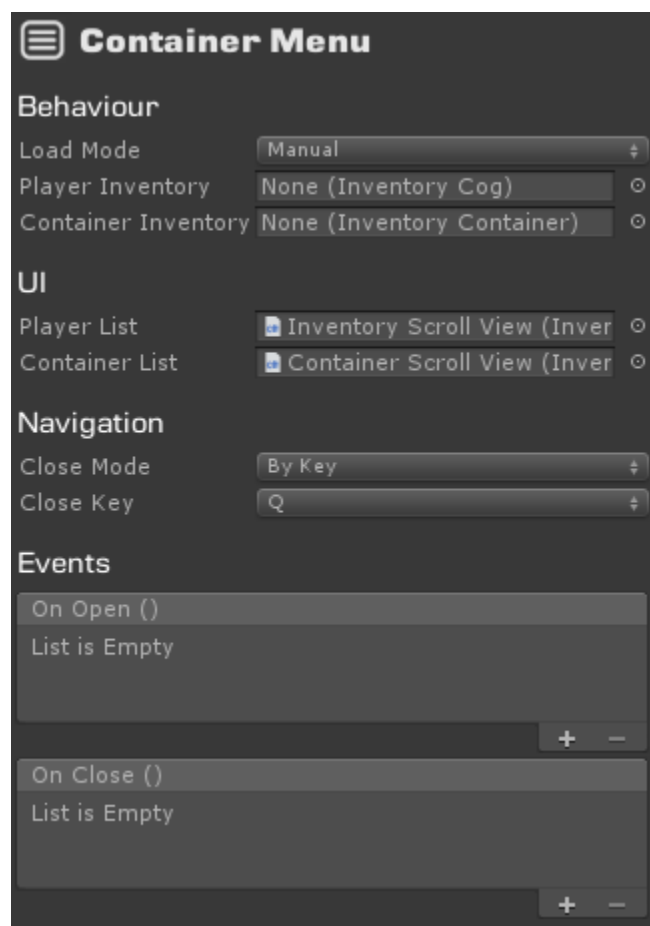
Inventory Cog™ comes with an array of built-in UI components for you to use. You are not required to use any of them and can easily roll your own with the API or using the built-in options as base classes.

### Menu Components

Several portions of Inventory Cog™ allow for you to provide menu interfaces. These interface not only display information to your player but will also pass information down to child components in many cases.

#### Container Menu UI

Used in association with Inventory Containers.

The image shows a configuration window titled "Container Menu" with a hamburger menu icon in the top left. The window is divided into several sections: "Behaviour", "UI", "Navigation", and "Events". Under "Behaviour", there are three settings: "Load Mode" set to "Manual", "Player Inventory" set to "None (Inventory Cog)", and "Container Inventory" set to "None (Inventory Container)". Under "UI", there are two settings: "Player List" set to "Inventory Scroll View (Inver" and "Container List" set to "Container Scroll View (Inver". Under "Navigation", there are two settings: "Close Mode" set to "By Key" and "Close Key" set to "Q". Under "Events", there are two event lists. The first list is for "On Open ()" and contains one event "List is Empty". The second list is for "On Close ()" and also contains one event "List is Empty". Each event list has a "+" button to add more events and a "-" button to remove them.

**Load Mode** – Determines how to load data into the UI (Manual or On Enable); Inventory Cog™ will automatically call load methods when the menu is opened via Inventory Cog™

**Player Inventory** – Reference to the player's Inventory Cog™ (automatically populated if opened via Inventory Cog™)

**Container Inventory** – Reference to the Inventory Container being interacted with (automatically populated if opened via Inventory Cog™)

**Player List** – Reference to Inventory Item List used to display the player's inventory

**Container List** – Reference to Inventory Item List used to display the container's inventory

**Close Mode** – Determines how to close the menu (Manually, By Button, By Key)

**Close Button** – Button to monitor for close (By Button only)

**Close Key** – Key to monitor for close (By

Key only)

**On Open** – Event fired when the menu is opened

**On Close** – Event fired when the menu is closed

## Inventory Menu UI

This is the main menu for displaying inventory to your player. All Inventory Item List, Equip Point UI, Recipe List and Slot Item UI components that are children of this object will automatically be given a reference to the Inventory Cog™. Additionally, Slot Item UI components will have their LoadSlot method called.



**Close Mode** – Determines how to close the menu (Manually, By Button, By Key)

**Close Button** – Button to monitor for close (By Button only)

**Close Key** – Key to monitor for close (By Key only)

## Item Menu UI

Provides support for context menus within menus. These are used in the Stylized Adventure Inventory demo.



**Allow Auto Wrap** – When checked navigation from first item to last or vise-versa will be allowed

**Enable Click Select** – If checked menu options will be monitored for click events

**Navigation Mode** – Determines how to navigation between options (Manually, By Button, By Key)

**Nav Button** – Button to monitor for navigation (By Button only)

**Back Key** – Key to monitor to select previous (By Key only)

**Next Key** – Key to monitor to select next (By Key only)

**Invert Input** – If true input is inverted

**Allow Auto Repeat** – If true input will automatically be repeated

**Repeat Delay** – Seconds to wait before repeating input

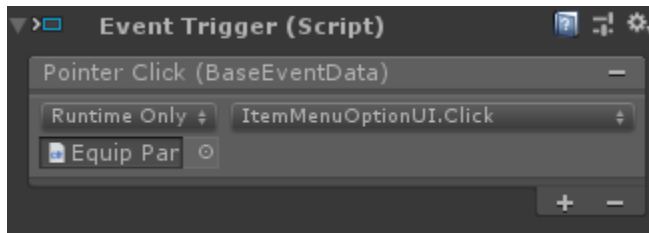
**Selection Mode** – Determines how to submit selection (Manually, By Button, By Key)

**Button Submit** – Button to monitor for submit (By Button only)

**Key Submit** – Key to monitor for submit (By Key only)

## Item Menu Option

This component needs to be added to each element in the Item Menu UI that is available for selection.

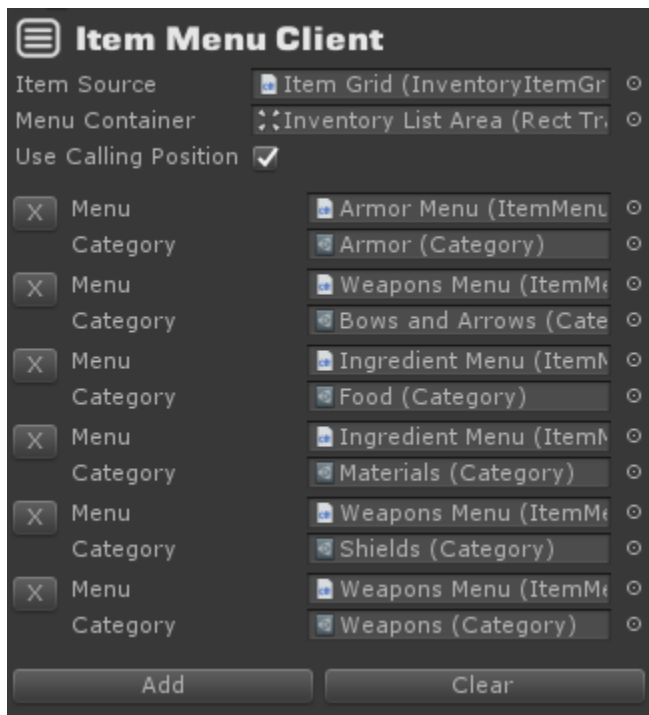


There are four events available on this component: On Selected, On Un Selected, On Submit and On Click.

If you wish to enable mouse support, you will need to add an Event Trigger that calls the Click event (as seen left)

## Item Menu Client

The Item Menu Client provides a middleman between the original menu and the Item Menu being called. This can be seen in action in the Stylized Adventure Inventory demo Inventory Menu prefab.



**Item Source** – Inventory Item List to monitor for item submission

**Menu Container** – Transform to assign as parent to opened menu

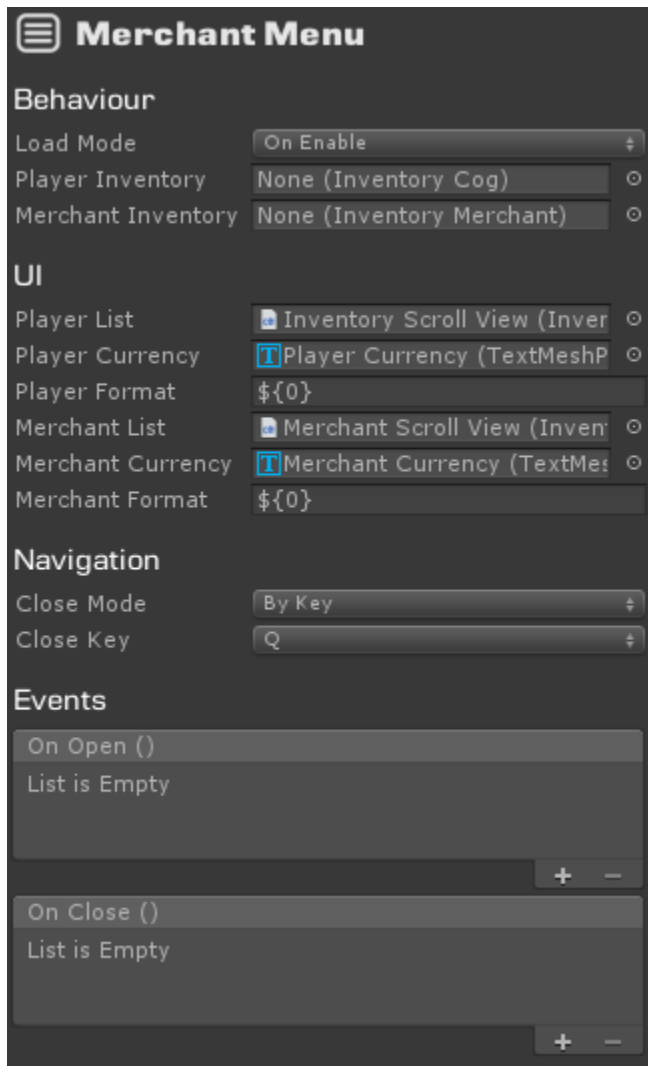
**Use Calling Position** – When checked the spawned menu will be placed at the center of the submitted item

**Menu** – Item Menu UI to display

**Category** – Category of item to display menu for

## Merchant Menu UI

Used in association with Inventory Merchants.



**Load Mode** – Determines how to load data into the UI (Manual or On Enable); Inventory Cog™ will automatically call load methods when the menu is opened via Inventory Cog™

**Player Inventory** – Reference to the player's Inventory Cog™ (automatically populated if opened via Inventory Cog™)

**Merchant Inventory** – Reference to the Inventory Merchant being interacted with (automatically populated if opened via Inventory Cog™)

**Player List** – Reference to Inventory Item List used to display the player's inventory

**Player Currency** – TextMeshPro component to display player's available currency

**Player Format** – Format to use when displaying currency, {0} is replaced with currency value

**Merchant List** – Reference to Inventory Item List used to display the merchant's inventory

**Merchant Currency** – TextMeshPro component to display merchant's available currency

**Merchant Format** – Format to use when displaying currency, {0} is replaced with currency value

**Close Mode** – Determines how to close the menu (Manually, By Button, By Key)

**Close Button** – Button to monitor for close (By Button only)

**Close Key** – Key to monitor for close (By Key only)

**On Open** – Event fired when the menu is opened

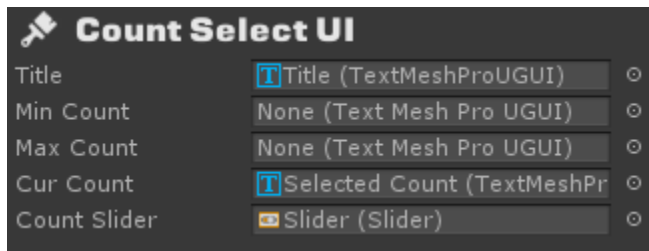
**On Close** – Event fired when the menu is closed

## Pickup Menu UI

Used for menus opened by Loot Item With UI. No parameters are used with this component. However, references to the Inventory Cog™ and Loot Item With UI are passed in.

## Count Select UI

Used with modals to select counts



**Title** – TextMeshPro object to place title on

**Min Count** – TextMeshPro object to place minimum allowed count on

**Max Count** – TextMeshPro object to place maximum allowed count on

**Cur Count** – TextMeshPro object to

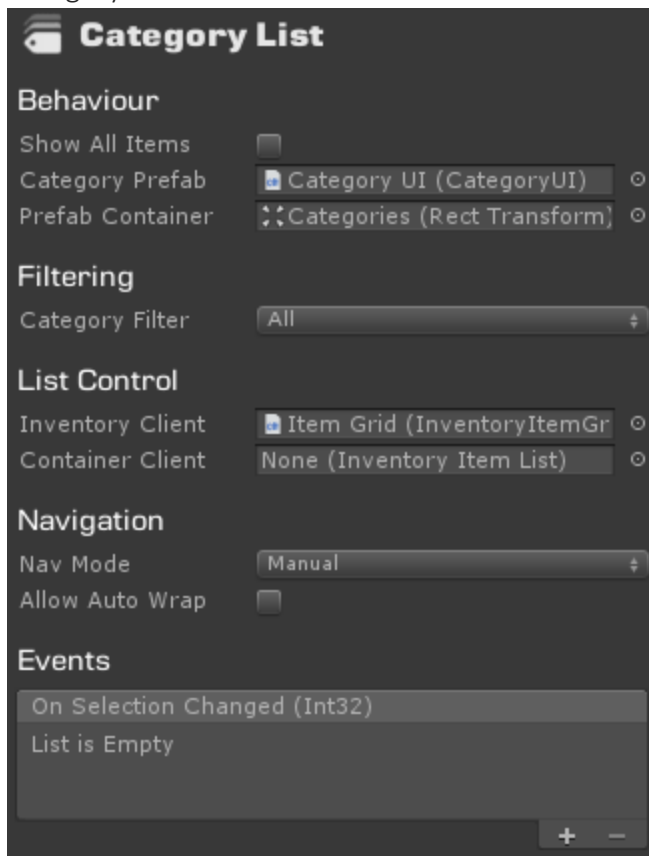
place current selected count on

**Count Slider** – Slider used to adjust selected count

## List Components

In many cases you will want to display a list of items in the Player's inventory, a container, etc. Components in this section aid with that display.

### Category List



**Show All Items** – If checked an additional “All Items” category will be displayed at the start of the list

**All Items Icon** – Icon to display for the “All Items” category

**All Items Text** – Text to display for the “All Items” category

**Category Prefab** – Category UI prefab to use for each item in list

**Prefab Container** – Transform parent to place items in

**Category Filter** – Filter to use when displaying categories (All, In List, Not In List) when either In List or Not In List is selected Inventory Categories can be dragged and dropped into the list.

**List Control** – The Inventory Item List, if provided, will have its category updated to reflect the category selected by this component

**Container Control** – The Inventory Item List, if provided, will have its category updated to reflect the category

selected by this component

**Navigation Mode** – Determines how to navigate categories (Manually, By Button, By Key)

**Nav Button** – Button to monitor for navigation (By Button only)

**Back Key** – Key to monitor to select previous (By Key only)

**Next Key** – Key to monitor to select next (By Key only)

**Allow Auto Wrap** – If check movement between first/last items is allowed

**On Selection Changed** – Event fired when selected category changes

Crafting Category List

The image shows the Unity Inspector for the 'Crafting Category List' component. It is divided into several sections: 'Behaviour', 'Filtering', 'Navigation', and 'Events'. In the 'Behaviour' section, 'Show All Items' is unchecked, and 'Category Prefab', 'Prefab Container', and 'Recipe Client' are all set to 'None'. The 'Filtering' section has 'Category Filter' set to 'All'. The 'Navigation' section has 'Nav Mode' set to 'By Key', 'Back Key' set to 'Z', 'Next Key' set to 'C', and 'Allow Auto Wrap' is unchecked. The 'Events' section shows an event 'On Selection Changed (Int32)' with a listener 'List is Empty'. At the bottom right of the Events section are '+' and '-' buttons.

Section	Property	Value
Behaviour	Show All Items	<input type="checkbox"/>
	Category Prefab	None (Crafting Category UI)
	Prefab Container	None (Transform)
	Recipe Client	None (Recipe List)
Filtering	Category Filter	All
Navigation	Nav Mode	By Key
	Back Key	Z
	Next Key	C
	Allow Auto Wrap	<input type="checkbox"/>
Events	On Selection Changed (Int32)	List is Empty

**Show All Items** – If checked an additional “All Items” category will be displayed at the start of the list

**All Items Icon** – Icon to display for the “All Items” category

**All Items Text** – Text to display for the “All Items” category

**Category Prefab** – Category UI prefab to use for each item in list

**Prefab Container** – Transform parent to place items in

**Recipe Client** – Recipe List, if provided, will have its category updated to reflect the category selected by this component

**Category Filter** – Filter to use when displaying categories (All, In List, Not In List) when either In List or Not In List is selected Inventory Categories can be dragged and dropped into the list.

**Navigation Mode** – Determines how to

navigate categories (Manually, By Button, By Key)

**Nav Button** – Button to monitor for navigation (By Button only)

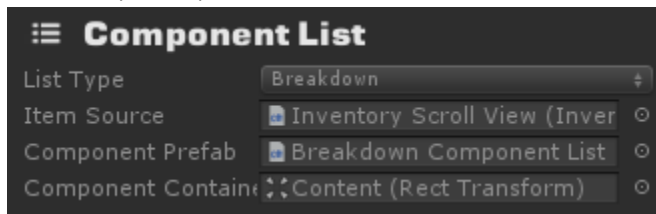
**Back Key** – Key to monitor to select previous (By Key only)

**Next Key** – Key to monitor to select next (By Key only)

**Allow Auto Wrap** – If check movement between first/last items is allowed

**On Selection Changed** – Event fired when selected category

## Inventory Component List



**List Type** – Type of list to display for (Breakdown or Repair)

**Item Source** – Inventory Item List to monitor for selection

**Component Prefab** – Prefab to display for each component associated with

item selected in Item Source

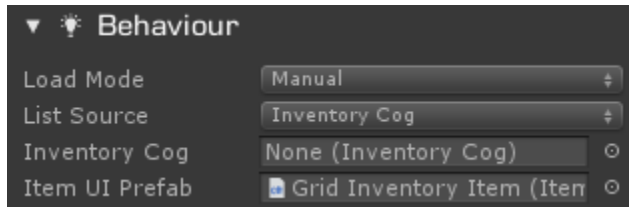
**Component Container** – Transform parent for prefabs

## Inventory Item List

This base class is used to support the Inventory Item Grid and Inventory Item Scroll List components. It should **not** be added to any object directly.

## Inventory Item Grid

Displays inventory items in a grid. As a larger component this item is divided into sections. As covered in the Menu Components section Inventory Cog™, Container and Merchant can be automatically populated, and load methods automatically called when opened by the Inventory Cog™. The Category Filter can also be overridden by a Category List.



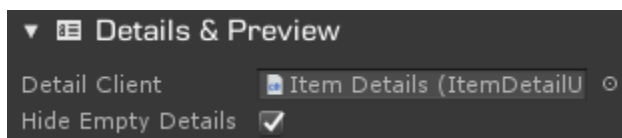
**Load Mode** – Determines when to load items (Manual or On Enable)

**List Source** – Data source for list (Inventory Cog, Inventory Container or Inventory Merchant)

**Inventory Cog/Container/Merchant** –

Reference to relevant selected source

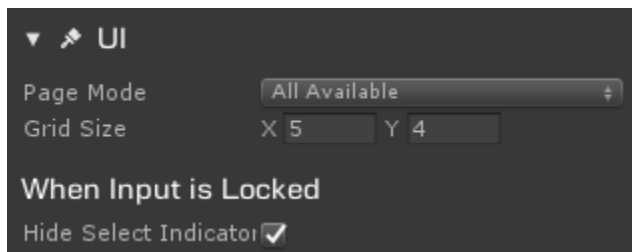
**Item UI Prefab** – Prefab to instance for each item



**Detail Client** – Item Detail UI reference to populated with selected item

**Hide Empty Details** – If checked the Detail Client will be hidden when an

empty slot is selected



**Page Mode** – Determines how to handle paging of items (All Available or Only Used)

**Grid Size** – Layout of grid, should reflect grid per page or all items if not using paging

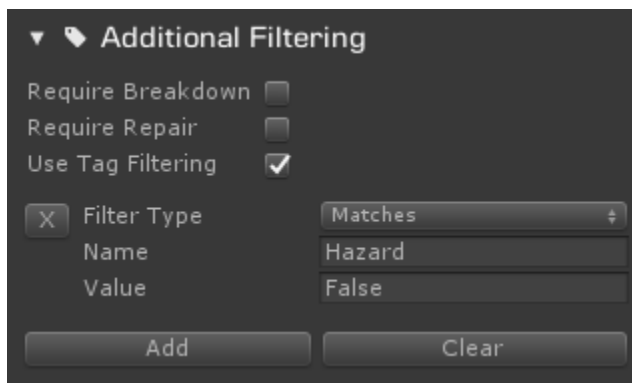


**Hide Select Indicator** – When checked the selected item indicator will not be displayed if input is locked



**Category Filter** – Determines which categories to display items for (All, In List, Not In List)

**Categories** – Categories can be dragged and dropped for In List and Not In List



**Require Breakdown** – If checked only items that support breakdown will be displayed

**Require Repair** – If checked only items that support repair **and** have a condition of less than 1 will be displayed

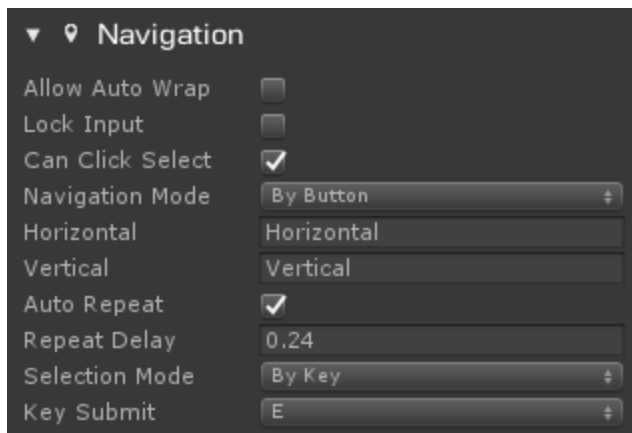
**Use Tag Filtering** – If checked only items that match **all** tag filters will be displayed

**Filter Type** – Type of filter to apply

(Matches, Exists, Does Not Exist)

**Name** – Name of tag to check for

**Value** – Value of tag to compare (Matches only)



**Allow Auto Wrap** – When checked movement between first/last items will be allowed

**Lock Input** – When checked input to component will be disabled

**Can Click Select** – If checked items will be monitored for click events

**Navigation Mode** – Determines how to navigate items (Manually, By Button, By Key)

**Horizontal** – Button to monitor for horizontal movement (By Button only)

**Vertical** – Button to monitor for vertical movement (By Button only)

**Left** – Key to monitor for left movement (By Key only)

**Right** – Key to monitor for right movement (By Key only)

**Up** – Key to monitor for up movement (By Key only)

**Down** – Key to monitor for down movement (By Key only)

**Allow Auto Repeat** – If true input will automatically be repeated

**Repeat Delay** – Seconds to wait before repeating input

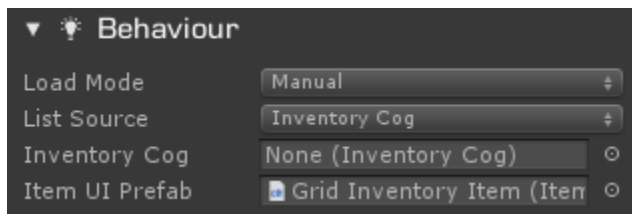
**Selection Mode** – Determines how to submit selection (Manually, By Button, By Key)

**Button Submit** – Button to monitor for submit (By Button only)

**Key Submit** – Key to monitor for submit (By Key only)

### Inventory Item Scroll List

Displays inventory items in a scroll view. As a larger component this item is divided into sections. As covered in the Menu Components section Inventory Cog™, Container and Merchant can be automatically populated, and load methods automatically called when opened by the Inventory Cog™. The Category Filter can also be overridden by a Category List.



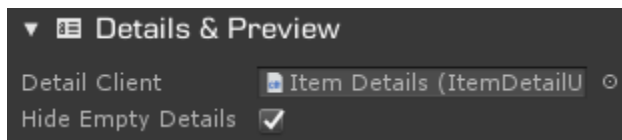
**Load Mode** – Determines when to load items (Manual or On Enable)

**List Source** – Data source for list (Inventory Cog, Inventory Container or Inventory Merchant)

**Inventory Cog/Container/Merchant** –

Reference to relevant selected source

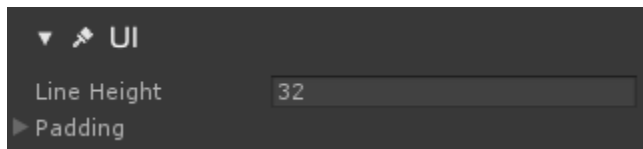
**Item UI Prefab** – Prefab to instance for each item



**Detail Client** – Item Detail UI reference to populated with selected item

**Hide Empty Details** – If checked the Detail Client will be hidden when an

empty slot is selected



**Line Height** – Height of each line including any padding in the Horizontal/Vertical Layout Group.

**Padding** – Extra padding to apply



**Category Filter** – Determines which categories to display items for (All, In List, Not In List)

**Categories** – Categories can be dragged and dropped for In List and Not In List

**Require Breakdown** – If checked only items that support breakdown will be displayed

**Require Repair** – If checked only items that support repair **and** have a condition of less than 1 will be displayed

**Use Tag Filtering** – If checked only items that match **all** tag filters will be displayed

**Filter Type** – Type of filter to apply

(Matches, Exists, Does Not Exist)

**Name** – Name of tag to check for

**Value** – Value of tag to compare (Matches only)

**Allow Auto Wrap** – When checked movement between first/last items will be allowed

**Lock Input** – When checked input to component will be disabled

**Can Click Select** – If checked items will be monitored for click events

**Navigation Mode** – Determines how to navigate items (Manually, By Button, By Key)

**Nav Button** – Button to monitor for

movement (By Button only)

**Back Key** – Key to monitor to select previous (By Key only)

**Next Key** – Key to monitor to select next (By Key only)

**Allow Auto Repeat** – If true input will automatically be repeated

**Repeat Delay** – Seconds to wait before repeating input

**Selection Mode** – Determines how to submit selection (Manually, By Button, By Key)

**Button Submit** – Button to monitor for submit (By Button only)

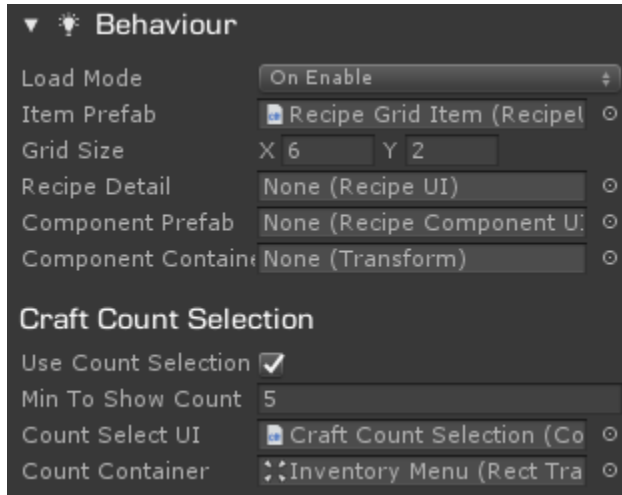
**Key Submit** – Key to monitor for submit (By Key only)

## Recipe List

This base class is used to support the Recipe Item Grid and Recipe Scroll List components. It should **not** be added to any object directly.

## Recipe Item Grid

Displays crafting recipes in a grid. As a larger component this item is divided into sections. As covered in the Menu Components section Inventory Cog™ can be automatically populated, and load methods automatically called when opened by the Inventory Cog™. The Category Filter can also be overridden by a Category List.



**Load Mode** – Determines when to load items (Manual or On Enable)

**Item Prefab** – Prefab to instance for each item

**Grid Size** – Layout of grid, should reflect grid per page or all items if not using paging

**Recipe Detail** – Recipe UI reference to update when selected item is changed

**Component Prefab** – Prefab to use to display required components

**Component Container** – Transform parent for prefab instances

**Use Count Selection** – When checked system will prompt user how many items to buy/sell

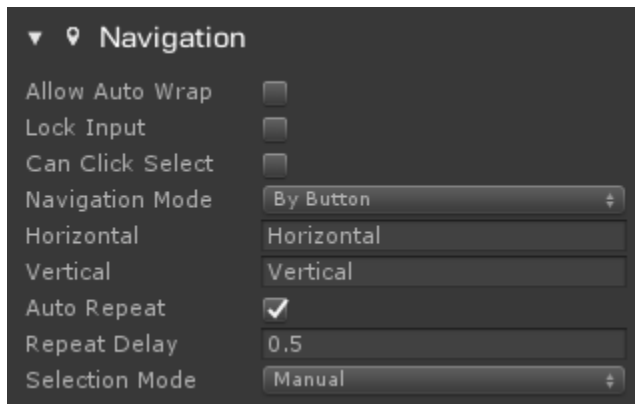
**Min To Show Count** – The minimum number of an item a user must be able to buy/sell in order to show count prompt (otherwise 1 item will be bought/sold each time)

**Count Select UI** – Prefab for selecting count

**Count Container** – Parent transform for count container (if any)



**Category Filtering** – List of strings corresponding to the scriptable object name of Crafting Categories to display. Note: this is the name in your “Project” pane **not** the Display Name



**Allow Auto Wrap** – When checked movement between first/last items will be allowed

**Lock Input** – When checked input to component will be disabled

**Can Click Select** – If checked items will be monitored for click events

**Navigation Mode** – Determines how to navigate items (Manually, By Button, By Key)

**Horizontal** – Button to monitor for

horizontal movement (By Button only)

**Vertical** – Button to monitor for vertical movement (By Button only)

**Left** – Key to monitor for left movement (By Key only)

**Right** – Key to monitor for right movement (By Key only)

**Up** – Key to monitor for up movement (By Key only)

**Down** – Key to monitor for down movement (By Key only)

**Allow Auto Repeat** – If true input will automatically be repeated

**Repeat Delay** – Seconds to wait before repeating input

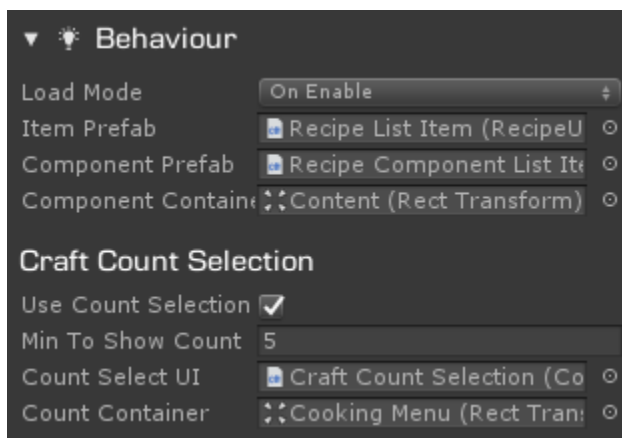
**Selection Mode** – Determines how to submit selection (Manually, By Button, By Key)

**Button Submit** – Button to monitor for submit (By Button only)

**Key Submit** – Key to monitor for submit (By Key only)

## Recipe Scroll List

Displays crafting recipes in a scroll view. As a larger component this item is divided into sections. As covered in the Menu Components section Inventory Cog™ can be automatically populated, and load methods automatically called when opened by the Inventory Cog™. The Category Filter can also be overridden by a Category List.



**Load Mode** – Determines when to load items (Manual or On Enable)

**List Source** – Data source for list (Inventory Cog, Inventory Container or Inventory Merchant)

**Inventory Cog/Container/Merchant** – Reference to relevant selected source

**Item UI Prefab** – Prefab to instance for each

**Title** – TextMeshPro object to place title on

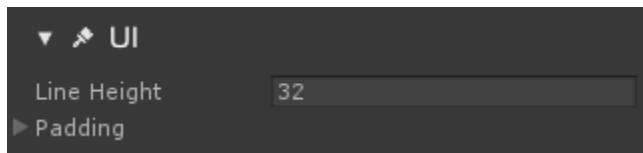
**Min Count** – TextMeshPro object to

place minimum allowed count on

**Max Count** – TextMeshPro object to place maximum allowed count on

**Cur Count** – TextMeshPro object to place current selected count on

**Count Slider** – Slider used to adjust selected count

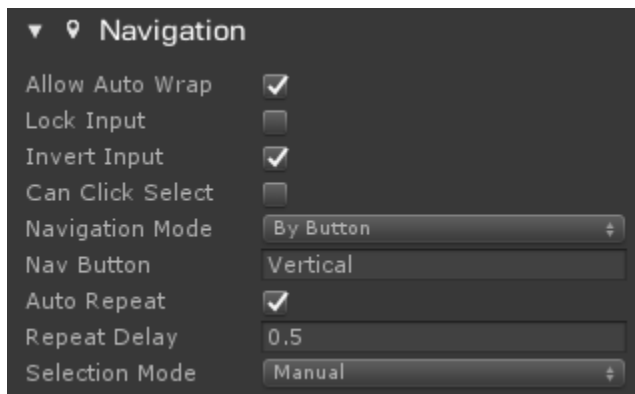


**Line Height** – Height of each line including any padding in the Horizontal/Vertical Layout Group.

**Padding** – Extra padding to apply



**Category Filtering** – List of strings corresponding to the scriptable object name of Crafting Categories to display. Note: this is the name in your “Project” pane **not** the Display Name



**Allow Auto Wrap** – When checked movement between first/last items will be allowed

**Lock Input** – When checked input to component will be disabled

**Can Click Select** – If checked items will be monitored for click events

**Navigation Mode** – Determines how to navigate items (Manually, By Button, By Key)

**Nav Button** – Button to monitor for

movement (By Button only)

**Back Key** – Key to monitor to select previous (By Key only)

**Next Key** – Key to monitor to select next (By Key only)

**Allow Auto Repeat** – If true input will automatically be repeated

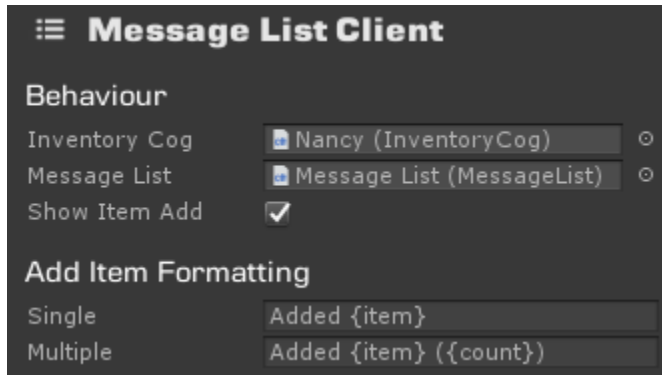
**Repeat Delay** – Seconds to wait before repeating input

**Selection Mode** – Determines how to submit selection (Manually, By Button, By Key)

**Button Submit** – Button to monitor for submit (By Button only)

**Key Submit** – Key to monitor for submit (By Key only)

Inventory Message List Client  
Subscribes to events to display messages



**Inventory Cog** – Instance of Inventory Cog™ to monitor

**Message List** – Message list to send events to

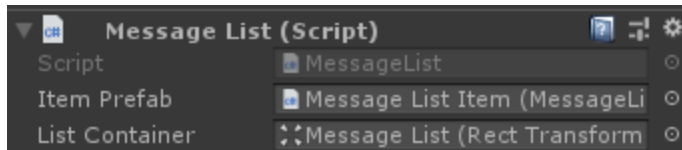
**Show Add Item** – When selected messages will be displayed when new items are added

**Single** – Text format to display when a single count is added

**Multiple** – Text format to display when

multiple counts are added

Message List  
Displays a list of messages



**Item Prefab** – Prefab to instantiate for each message

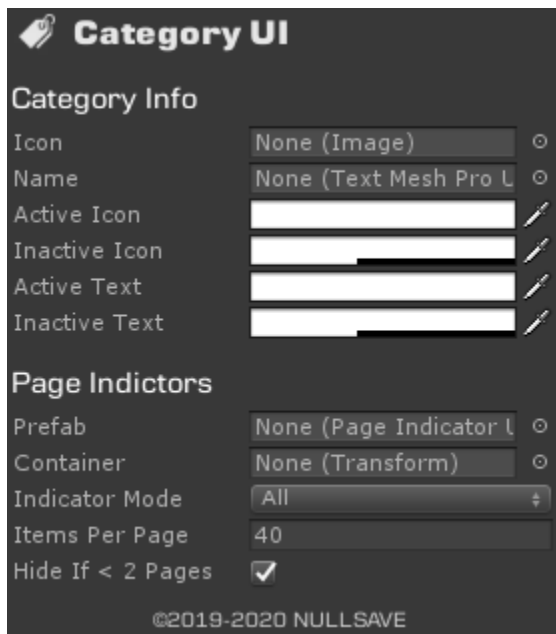
**List Container** – Transform parent for prefab instances

## Item Elements

Each list or grid needs item elements to display. The components in this section provide those.

### Category UI

Representing a single Category entry, the CategoryUI allows you to specify how an element is displayed in a category list or other custom UI. All elements are optional, use only the ones you wish.



**Icon** – Displays the Category Icon

**Name** – Displays the Category Display Name

**Active Icon** – Color to assign Icon if the Category is selected in the list

**Inactive Icon** – Color to assign Icon when the Category is **not** selected in the list

**Active Text** – Color to assign Display Name if the Category is selected in the list

**Inactive Text** – Color to assign Display Name when the Category is **not** selected in the list

**Page Indicators** – The Page indicators section allows you to specify a prefab that can be repeated in a list to display the pages associated with the Category.

**Prefab** – GameObject to instantiate for each page

**Container** – Transform to parent the

instantiated objects to

**Indicator Mode** – All (displays all pages even locked ones), All Unlocked (displayed all available unlocked paged), Only Used (displays instance for used pages only)

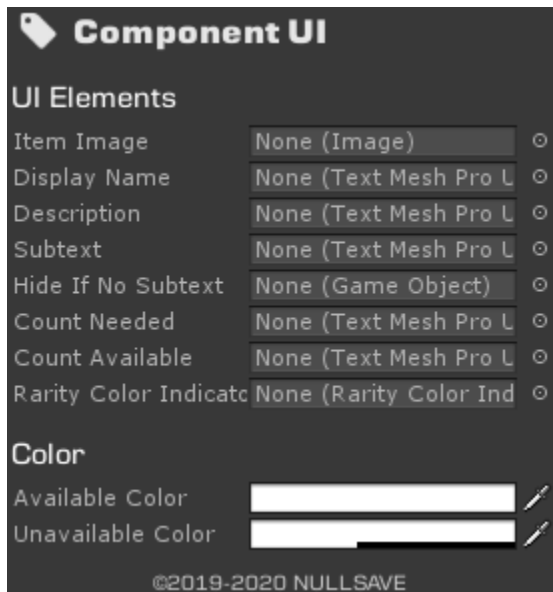
**Items Per Page** – The number of items that are displayed on each page

**Hide If < 2 Pages** – When checked no indicators will be created unless there are at least 2 pages to display

### Component UI

Representing a single crafting component entry, the ComponentUI allows you to specify how an element is displayed in a component list or other custom UI. All elements are optional, use only the ones you wish.



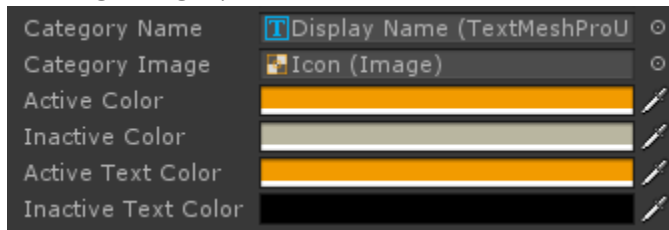


**Item Image** - Displays the InventoryItem icon  
**Display Name** - Show the item's Display Name  
**Description** - Shows the item's description  
**Subtext** - Shows the item's subtext  
**Hide If No Subtext** - Specifies a GameObject to set inactive if the item's subtext is empty or null  
**Count Needed** - Displays the number of components required  
**Count Available** - Displays the number of components in inventory  
**Rarity Color Indicator** - Specifies a RarityColorIndicator to update with the item  
**Available Color** - Color to apply when the available count is greater or equal to the required count. Applies to Image Icon, Display

Name, Description, Subtext, Count Needed and Count Available

**Unavailable Color** - Color to apply when the available count is less than the required count. Applies to the same objects as Available Color

#### Crafting Category UI



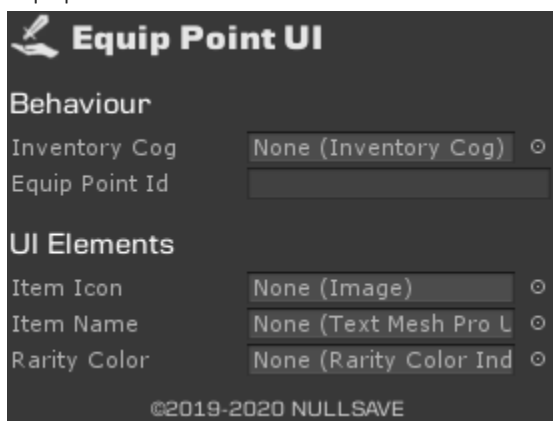
**Category Name** - TextMeshPro object to display Category Name  
**Category Image** - Image to display Category Icon  
**Active Color** - Color to assign image when category is selected

**Inactive Color** - Color to assign to image when category is not selected

**Active Text Color** - Color to assign text when category is selected

**Inactive Text Color** - Color to assign to text when category is not selected

#### Equip Point UI



**Inventory Cog** - Reference to inventory to monitor (can be auto-populated by Inventory Menu)

**Equip Point Id** - ID of the equip point to monitor

**Item Icon** - Displays the item's Icon

**Item Name** - Shows the item's Display Name

**Rarity Color** - Binds to a Rarity Color Indicator

## Item Detail UI

The Item Detail UI shows detailed information for an Inventory Item and is generally used on a prefab for lists. All elements are optional.

**Item Detail UI**

**Category**

Sprite: None (Image) ○

Name: None (Text Mesh) ○

**Item**

Sprite: None (Image) ○

Name: Item Name (Text Mesh) ○

Description: Item Description (Text Mesh) ○

Subtext: None (Text Mesh) ○

Item Preview: None (Item Preview UI) ○

Recipe UI: None (Recipe UI) ○

**Base Stats**

Custom Stat 1: [Empty] ○

Stat 1 Display: None (Text Mesh) ○

Custom Stat 2: [Empty] ○

Stat 2 Display: None (Text Mesh) ○

Ammo Type: None (Text Mesh) ○

Value: None (Text Mesh) ○

Weight: None (Text Mesh) ○

**Condition**

Text: None (Text Mesh) ○

Slider: None (Slider) ○

**Rarity**

Color Indicator: Rarity (Rarity Color) ○

Slider: None (Slider) ○

Sprite: None (Sprite) ○

Sprite Container: None (Transform) ○

Sprite Size: X 16 Y 16

**Tags**

Tag Prefab: None (Item Tag) ○

Tag Container: None (Transform) ○

©2019-2020 NULLSAVE

**Category Sprite** – Shows the icon associated with the item's Category

**Category Name** – Shows the Category's display name

**Item Sprite** – Shows the item's icon

**Item Name** – Shows the item's Display Name

**Item Description** – Shows the item's Description

**Item Subtext** – Shows the item's subtext

**Item Preview** – Supplies the Item Preview UI prefab to use when displaying the item's preview model

**Recipe UI** – UI to use for any associated Display Recipe

**Custom Stat 1** – Specifies the name of the stat to display

**Stat 1 Display** – Shows the specified Custom Stats value

**Custom Stat 2** – Specifies the name of the stat to display

**Stat 2 Display** – Shows the specified Custom Stats value

**Ammo Type** – Shows the item's ammo type

**Value** – Displays the item's value

**Weight** – Shows the item's weight

**Condition Text** – Shows the item's condition value

**Condition Slider** – Sets a slider's value to reflect the item's condition

**Rarity Color Indicator** – Binds to a Rarity Color Indicator

**Rarity Slider** – Binds a slider to the item's rarity

**Rarity Sprite** – Specifies a sprite to be repeated for each 1x in the rarity value

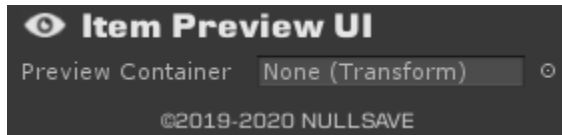
**Rarity Sprite Container** – Sets the created sprite(s) parent transform

**Sprite Size** – Sets the sprite's RectTransform sizeDelta

**Tag Prefab** – Specifies an Item Tag UI prefab to display for each tag in the item

**Tag Container** – Sets the created tag(s) parent transform

## Item Preview UI

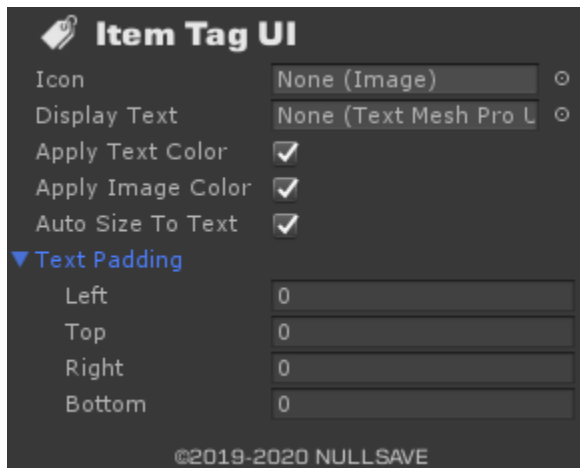


The Item Preview UI is used to create a prefab for displaying an item's preview model.

**Preview Container** – Sets the transform parent for the preview model

## Item Tag UI

Used on a prefab, Item Tag UI, specifies how information for an Item Tag is displayed.



**Icon** – Shows the item's icon

**Display Text** – Shows the item's Display Name

**Apply Text Color** – When checked the "Display Text" color will be updated to reflect the color in Item Tag

**Apply Image Color** – When check the "Icon" color will reflect that in the Item Tag

**Auto Size To Text** – When check the object's RectTransform will be resized to fit the text

**Text Padding** – Padding to add to the RectTransform's size when "Auto Size To Text" is checked

### Item UI

#### Item Info

Icon

Name

Description

Subtext

Hide If No Subtext

Recipe UI

#### Count

Prefix

Text

Suffix

Hide if Count < 2

#### Indicators

Selected

Can Equip

Is Equipped

Rarity Color Indicator

Condition Slider

Hide if Condition 0 ☐

Rarity Slider

Hide if Rarity 0 ☐

#### Modifiers

Custom Stat 1

Stat 1 Display

Hide If Stat 1 Zero

Custom Stat 2

Stat 2 Display

Hide If Stat 2 Zero

#### Tags

Prefab

Container

#### Events

On Click (ItemUI)

List is Empty

On Loaded Item (InventoryItem)

List is Empty

A less detailed Item information behaviour, Item UI, is used on prefabs; often ones consumed by lists. All elements are options

**Icon** – Shows the item's icon

**Name** – Shows the item's Display Name

**Description** – Shows the item's description

**Subtext** – Shows the item's subtext

**Hide If No Subtext** – Hides specified GameObject if subtext is null or empty

**Recipe UI** – UI to use for any associated Display Recipe

**Prefix** – Text to display before item's count

**Text** – Element to display item's count (with prefix and suffix)

**Suffix** – Text to display after item's count

**Hide if Count < 2** – Specifies GameObject to display if item's count is less than 2

**Selected** – GameObject to show when item is selected in list

**Can Equip** – GameObject to show if the item can be equipped

**Is Equipped** – GameObject to show if the item is currently equipped or stored

**Rarity Color Indicator** – Rarity Color Indicator to bind to item

**Condition Slider** – Slider to bind to item's condition

**Custom Stat 1** – Specifies the name of the stat to display

**Stat 1 Display** – Shows the specified Custom Stat's value

**Hide If Stat 1 Zero** – GameObject to hide if stat's value is zero

**Custom Stat 2** – Specifies the name of the stat to display

**Stat 2 Display** – Shows the specified Custom Stat's value

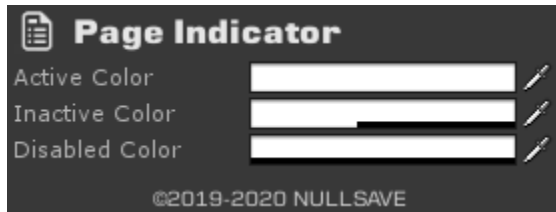
**Hide If Stat 2 Zero** – GameObject to hide if stat's value is zero

**Tag Prefab** – Specifies an Item Tag UI prefab to display for each tag in the item

**Tag Container** – Sets the created tag(s) parent transform

## Page Indicator UI

The Page Indicator UI is used with prefabs and requires an Image object.

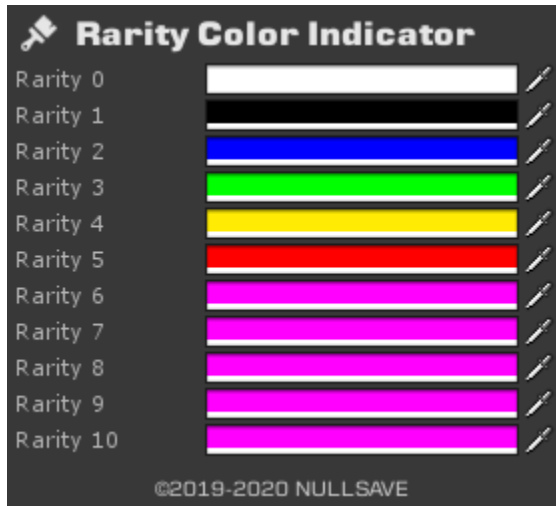


**Active Color** – Color to set image when page is active

**Inactive Color** – Color to set image when page is not active

**Disabled Color** – Color to set image if page is disabled

## Rarity Color Indicator



The Rarity Color Indicator is used with prefabs and requires an Image object.

Each possible Rarity value is given its own color which you can specify.



**Slot Item UI**

**Item Info**

Slot Id: 0

Icon: Item Icon (Image)

Name: None (Text Mesh Pro UGUI)

Description: None (Text Mesh Pro UGUI)

Subtext: None (Text Mesh Pro UGUI)

Hide If No Subtext: None (Game Object)

**Count**

Prefix: x

Text: Item Count (TextMeshPro)

Suffix:

Hide if Count < 2: Item Count

**Indicators**

Selected: Selection

Can Equip: None (Game Object)

Is Equipped: Equipped

Rarity Color Indicator: None (Rarity Color Indicator)

Condition Slider: Condition Slider (Slider)

Hide if Condition 0: ☒

Rarity Slider: None (Slider)

Hide if Rarity 0: ☐

**Modifiers**

Custom Stat 1:

Stat 1 Display: None (Text Mesh Pro UGUI)

Hide If Stat 1 Zero: None (Game Object)

Custom Stat 2:

Stat 2 Display: None (Text Mesh Pro UGUI)

Hide If Stat 2 Zero: None (Game Object)

**Tags**

Prefab: None (Item Tag UI)

Container: None (Transform)

**Slot Id** – Id of the slot to monitor

**Icon** – Shows the item's icon

**Name** – Shows the item's Display Name

**Description** – Shows the item's description

**Subtext** – Shows the item's subtext

**Hide If No Subtext** – Hides specified GameObject if subtext is null or empty

**Prefix** – Text to display before item's count

**Text** – Element to display item's count (with prefix and suffix)

**Suffix** – Text to display after item's count

**Hide if Count < 2** – Specifies GameObject to display if item's count is less than 2

**Selected** – GameObject to show when item is selected in list

**Can Equip** – GameObject to show if the item can be equipped

**Is Equipped** – GameObject to show if the item is currently equipped or stored

**Rarity Color Indicator** – Rarity Color Indicator to bind to item

**Condition Slider** – Slider to bind to item's condition

**Custom Stat 1** – Specifies the name of the stat to display

**Stat 1 Display** – Shows the specified Custom Stat's value

**Hide If Stat 1 Zero** – GameObject to hide if stat's value is zero

**Custom Stat 2** – Specifies the name of the stat to display

**Stat 2 Display** – Shows the specified Custom Stat's value

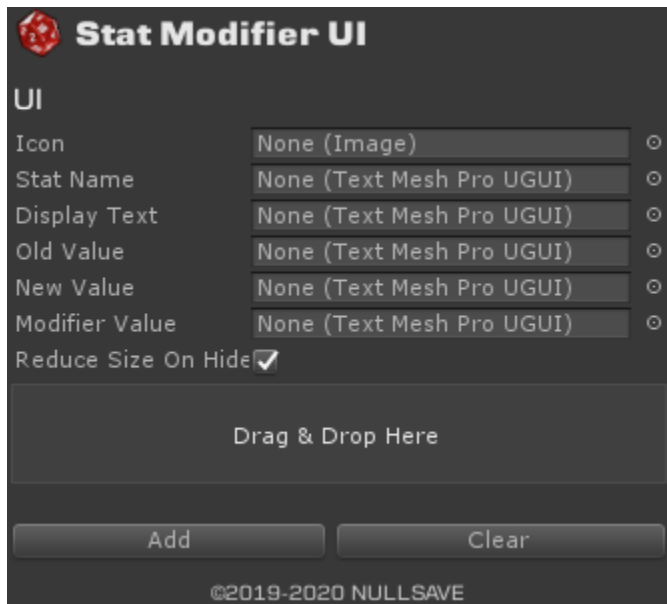
**Hide If Stat 2 Zero** – GameObject to hide if stat's value is zero

**Tag Prefab** – Specifies an Item Tag UI prefab to display for each tag in the item

**Tag Container** – Sets the created tag(s) parent transform

## Stat Modifier UI Editor

The Stat Modifier UI, which requires Stats Cog™ to function, is used with prefabs to display the Stat Modifiers associated with an Inventory Item.



**Icon** – Shows the modifiers icon  
**Stat Name** – Shows the name of the StatValue targeted by the StatModifier  
**Display Text** – Shows the modifiers Display Text  
**Old Value** – Shows the value of the StatValue **without** the modifier  
**New Value** – Shows the value of the StatValue **with** the modifier  
**Modifier Value** – Not currently used  
**Reduce Size On Hide** – Not currently used  
**Hide On Equipped** – GameObjects dropped here will be hidden when the item is equipped.

## Recipe Component UI

Representing a single crafting component entry, the RecipeComponentUI allows you to specify how an element is displayed in a component list or other custom UI. All elements are optional, use only the ones you wish.



**Item Image** – Displays the InventoryItem icon  
**Display Name** – Show the item's Display Name  
**Description** – Shows the item's description  
**Subtext** – Shows the item's subtext  
**Hide If No Subtext** – Specifies a GameObject to set inactive if the item's subtext is empty or null  
**Count Needed** – Displays the number of components required  
**Count Available** – Displays the number of components in inventory  
**Rarity Color Indicator** – Specifies a RarityColorIndicator to update with the item  
**Available Color** – Color to apply when the available count is greater or equal to the required count. Applies to Image Icon, Display

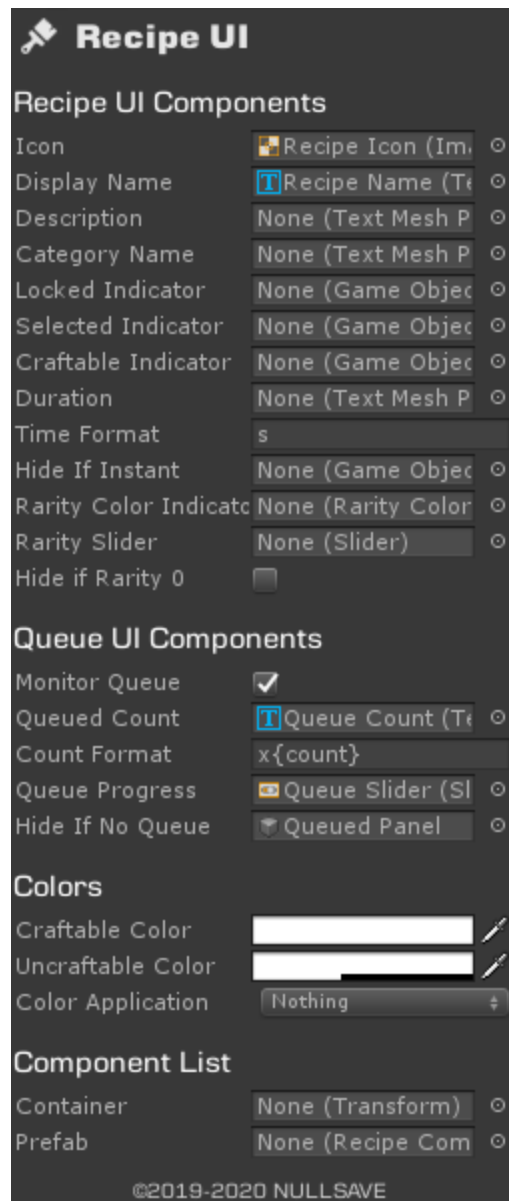
Name, Description, Subtext, Count Needed and Count Available

**Unavailable Color** – Color to apply when the available count is less than the required count. Applies to the same objects as Available Color



## Recipe UI

Recipe UI items are used to display details about a Recipe and can be used with Recipe Lists.



**Icon** – Shows the recipe icon

**Display Name** – Shows the recipe display name

**Description** – Shows the recipe description

**Category Name** – Shows the recipe category name

**Locked Indicator** – GameObject to show if the recipe is locked

**Selected Indicator** – GameObject to show if the recipe is selected in the list

**Craftable Indicator** – GameObject to show if the recipe can be crafted

**Duration** – TextMesh Pro object to display crafting time duration

**Time Format** – Format used to display crafting time

**Hide If Instant** – GameObject to hide if recipe crafts instantly

**Rarity Color Indicator** – Object to display recipe's rarity

**Rarity Slider** – Slider to display recipe's rarity

**Hide if Rarity 0** – When checked color indicator and slider are hidden if rarity is zero.

**Monitor Queue** – If checked object will subscribe to crafting queue for the recipe and display updates

**Queued Count** – TextMesh Pro object used to display queued count

**Count Format** – Format to use when displaying queued count

**Queue Progress** – Slider to display queue progress

**Hide if No Queue** – GameObject to hide if there

are not items in the queue.

**Craftable Color** – Color to set display items to when the recipe can be crafted

**Uncraftable Color** – Color to set display items to when the recipe cannot be crafted

**Container** – Container to place component prefabs in

**Prefab** – Prefab to instance for each component in the recipe.



## Quick API Reference

AddItemToInventory	Adds an instance of an item to the local inventory. An integer value is returned of the number of that object that could not be added for space or weight reasons. This can be used to update the existing drop object (handled automatically by LootItem)
BreakdownItem	Removes the supplied item from inventory and adds any breakdown components specified. If item is not marked as "Can Breakdown" the request is ignored, and the original item remains in inventory
Craft	Attempts to craft from either a recipe or a list of items. Returns a list of crafted items
ConsumeItem	Consumes a "consumable" item. If you have Stats Cog installed this will also apply an instant effects as well.
DropItem	Drops an item from the inventory
EquipItem	Equips an item (if equipable)
GetBreakdownItems	Returns a list of items that can be broken down (you may optionally supply a list of categories to filter by)
GetCategory	Returns an instantiated copy of the Category
GetDisplayedCategories	Returns a list of Categories to be displayed
GetCraftableCount	Returns the number of times a player can craft the supplied recipe with the items currently in their inventory
GetCraftableRecipes	Returns a list of all recipes the player can currently craft with inventory on hand
GetCraftableRecipesByCategory	Gets all recipes in a list of categories the player can craft with on hand inventory
GetGroupedCounts	Returns a list of ItemReferences with the overall count of each item (including the count over multiple stacks as a single result)
GetItemByInstanceId	Used by save/load this returns a specific instance of an Inventory Item
GetItemByName	Retrieves a <b>non-instantiated</b> copy of an Inventory Item by its object name
GetItemTotal	Returns total count of a single object across all stacks
GetItemFromInventory	Returns the first instantiated copy of an item by its name or non-instantiated reference
GetItems	Returns a list of all items by list of categories
GetItemsWithTag	Returns a list of all items in a list of categories with supplied tag and value
GetPointById	Returns Equip Point on character by Equip Point Id
GetPointToUse	Returns the recommend Equip Point to use for an item based on current point usage
GetRecipe	Returns first recipe that matches a list of components
GetRecipesByCategory	Returns a list of all Crafting Recipes with a specified crafting category list

<b>GetRecipeCraftable</b>	Returns true if Inventory contains the proper components and counts to craft supplied recipe
<b>GetRepairable</b>	Returns true if the supplied item can be repaired, has a condition of less than 1 and the required components are on hand
<b>GetRepairableItems</b>	Returns a list of all repairable items with an option to filter by list of categories
<b>GetRepairMaxIncrements</b>	Returns the total number of times an item can be repaired based on the inventory on hand and the items current condition
<b>GetRepairNeededItems</b>	Returns a list of all items needing repair (even if the items needed for repair are not available) with the option to filter by a list of categories
<b>GetUnlockedRecipesByCategory</b>	Returns a list of only recipes that have been unlocked, filtered by a list of categories
<b>GetSelectedAmmo</b>	Gets the currently selected Ammo Inventory Object based on the ammoType
<b>InventoryStateLoad</b>	Loads inventory from a file or stream
<b>InventoryStateSave</b>	Saves inventory to a file or stream
<b>Menu Close</b>	Closes the inventory menu
<b>Menu Open</b>	Opens the inventory menu
<b>RemoveItem</b>	Removes an item from the inventory without resulting Drop or Consume
<b>RepairItem</b>	Repairs supplied item as much as possible with inventory on hand
<b>SetSelectedAmmo</b>	Set the current selected Ammo Inventory Item
<b>StorePointById</b>	Returns Store Point on character by Store Point Id
<b>TransferMenuClose</b>	Closes the Transfer (container) menu
<b>TransferMenuOpen</b>	Opens the Transfer (container) menu
<b>UITextShow</b>	Shows the pickup UI
<b>UITextHid</b>	Hides the pickup UI
<b>UnequipItem</b>	Unequips an item
<b>UnstorePointById</b>	Moves item from Store Point to Equip Point by Store Point Id
<b>UseItem</b>	Removes item from inventory

## Change Log

### Version 1.12

#### Key New Features

- Added ability for player to give items custom names
- Added name modifiers to attachments
- Added skill slots
- Added support for loadouts

#### Updates

- Added support for auto creating attachment prefab
- Added "Always Fill Page" option when using "All Available" page mode in Item Grid
- Added "Inventory List Monitor" component, useful for updating UI command options
- Improved menu and prompt behavior
- You can now specify an offset for objects to spawn when dropped (default: 0, 1.5, 2)
- Added ItemContainerMenuUI component
- Added "Item Container UI" to item lists
- Added OpenItemContainer method to item lists
- Added InventoryItemScrollGrid component
- Added "Exclude Containers" to item lists filtering section
- Added "Enable Drag and Drop" to Item UI and Slot Item UI
- Added Sort UI component
- Improved Attachments and Attachment UI

#### Fixes

- Fixed error with Inventory DB Editor when first opening
- Corrected rebinding issue on Skinned Equip Points
- Corrected display issue with Show Locked Slots on Item Grid
- Fixed issue where onSelectionChanged didn't always fire for Item Grids
- Corrected SetDetail issue on Item Scroll List when a null item is selected
- Corrected issue where Item Scroll List did not show details after hiding on a null item
- Fixed Stat Name and Modifier Value not being populated on Stat Modifier UI
- Fixed issue where in certain cases an item could be equipped to multiple slots at the same time
- Corrected issue with Equip Point UI not properly reflecting unequip

## Version 1.11

### Major Improvements

- Added Attachments
- Added Effects to Inventory Items & Attachments
- Stats Cog now included!

### Updates & Fixes

- Added PromptUI
- Updated Store Point gizmo to reflect rotation
- Made Spawn Drop Manually into a dropdown call Spawn Drop (Automatic, Manual)
- Editor improvement
- Added "Refresh from Project" button to Inventory DB
- Top level Equip Object has location & rotation reset on equip
- Hierarchy Icons can now be enabled/disabled from Tools/NullSave menu
- Added ability to select Category with a click/tap
- Added Locked Indicator to ItemUI
- Added "Show Locked Slots" to Inventory item Lists
- Fixed "Hide Selection When Locked" on Item Grid

### Breaking Changes

- Inventory Cog PickupUI, ContainerUI, MerchantUI is now required to be a PromptUI
- Removed useAutoPickup (replaced by Automatic option in Pickup Mode)
- Removed Stat Modifiers (use Stat Effects)
- Removed Custom Stats (use Stat Effects)
- Save/Load not backwards compatible

## Version 1.10

### Improvements

- Updated support for Game Cog
- Corrected issues with Display Recipe in UI

## Version 1.9.1

## **Improvements**

- Added DLC support
- Added Icon to Recipe UI's editor
- Added Rarity to Recipe
- Added Rarity Slider & Color Indicator to Recipe UI
- Added Secondary Color to Recipe Component UI
- Added SetRarity(int) method to RarityColorIndicator
- Added null check to AddToInventory in InventoryCog
- Added ability to auto-load Inventory DB at runtime
- Added ability to Inventory DB to auto-create instance
- Updated save file version to 1.4 to support new abilities

## **Version 1.9**

### **Improvements**

- Removed ActivateByTag option to MenuOpenType enum
- Added null check for CraftingRecipe unsubscribe
- Updated InventoryMenuUI to try and find InventoryCog on "Player" tag if no inventory supplied
- Added empty checks to InventoryItemGrid navigation
- Added onZeroCount event to ItemUI
- Added Display Recipe to Inventory Item (UI section)
- Added Recipe UI to ItemUI and ItemDetailUI

### **Fixes**

- Corrected issue where consuming an item to zero count would not remove from UI until UI was reopened

## **Version 1.8**

### **Improvements**

- Updated classes with OnTriggerEnter2D and OnTriggerExit2D methods
- Added welcome screen
- Removed DataStore from Shared
- Made EquipPoints property public in Inventory Cog
- Added RigFallback to Skinned Equip Point
- Added TakeAll method to InventoryContainer
- Removed InventoryCog parameter from InventoryContainer.StateLoad
- Added GetEquipPoint method to InventoryCog
- Added onColliderEnabled and onColliderDisabled events to DamageDealer

- Added 2D support to DamageDealer
- Added onTakeDamage to DamageReceiver
- Updated base TakeDamage method on DamageReceiver

## Version 1.7

### Improvements

- Added ValueSource enum (Static, StatValue)
- Added BooleanSource enum (Static, StatExpression)
- Added StatValue options for Max Slots and Unlocked Slots to ItemCategory
- Added StatsCog, MaximumSlots, and UnlockedSlots properties to ItemCategory
- Made StatsCog reference in InventoryCog public
- Added StatValue options for Value and Success Chance to Crafting Recipe
- Added GetValue and GetSuccessChance methods to Crafting Recipe
- Added InventoryCog property to Inventory Item
- Added Initialize method to Inventory Item
- Added CanEquip, CanRepair and CanBreakdown properties to Inventory Item
- Added BooleanSource options for Can Equip, Can Repair, and Can Breakdown to Inventory Item
- Added FirstCrafted, LastCrafted, SuccessCount and FailCount variables to Crafting Recipe
- Added StateSave and StateLoad methods to Crafting Recipe
- Added new Crafting Recipe items to save/load (backwards compatible)
- Added Initialize method to Crafting Recipe
- Added InventoryCog property to Crafting Recipe
- Added BooleanSource option for Unlocked to Crafting Recipe
- Added DisplayInList to Crafting Recipe (with support for BooleanSource)

## Version 1.6

### Improvements

- Added Pickup, Container and Merchant raycasting
- Added ammo info to Equip Point UI
- Added GetEquippedAmmoCount to InventoryCog
- Updated EquipPoints to set damage dealer parent automatically (if StatsCog installed)
- Added craftTime (Instant, GameTime, RealTime) to Crafting Recipe
- Added craftType (Create, Upgrade) to Crafting Recipe

- Added craftSeconds to Crafting Recipe
- Added CraftingQueue to InventoryCog
- Added value to Crafting Recipe
- Added GetQueuedCount method to InventoryCog
- Added GetQueuedFirstProgress method to InventoryCog
- Added save/load support for CraftingQueue (backwards compatible)
- Added onCraftQueued event to InventoryCog
- Added onQueuedCraftComplete event to InventoryCog
- Added CompleteQueuedRecipe method to InventoryCog
- Added SlotItemUI MonoBehaviour (based on ItemUI)
- Added new "Cute Crafting" demo
- Added IsSlotIdUsed method to InventoryCog
- Added GetFirstFreeSlotId method to InventoryCog
- Added SlotId info to debug view
- Added Color Application to RecipeUI
- Added Crafting Categories to Inventory DB
- Updated Recipe Item Grid to load all categories if no list supplied
- Added Crafting Category List MonoBehaviour
- Added ItemCanUpgrade to InventoryCog
- Added Component Type (Standard/Advanced) to Crafting Recipe
- Added Advanced components to Crafting Recipe (set minimum condition/rarity levels)
- Added overload to GetItemTotalCount with minCondition & minRarity parameters
- Added Condition/Rarity sliders to Component UI
- Added Condition/Rarity sliders to Recipe Component UI
- Added Rarity slider to Item UI
- Added ActionRaycastTrigger and ActionRaycastTarget MonoBehaviours
- Added Currency to InventoryCog (added to save/load, backwards compatible)
- Added Inventory Merchant MonoBehaviour (supports save/load)
- Added InventoryMerchant to ListSource enum
- Added Merchant to InventoryItemList
- Added set to InventoryItemList.SelectedItem property

## **Fixes**

- Fixed issue in Animator Mods where int and floats were being swapped
- Fixed error on InventoryItemList when trying to select/unselect on null list

## **Breaking Changes**

- Crafting Recipe's Category string has been replaced with a Scriptable Object that contains Display Name, Description, Icon and Display Category

## Version 1.5

### Improvements

- Added Drag & Drop support to Inventory Container editor
- Added scroll view to Inventory Container editor
- Updated editor scroll views to automatically scroll to bottom when adding a new items
- Added filtering to Category List
- Added Component list items to Recipe UI (allows you to show components on same item as recipe)

### Fixes

- Updated Recipe List "Craft" methods to ignore requests for Locked recipes
- Fixed Quaternion save issue

## Version 1.4

### Improvements

- Updated SkinnedEquipPoint to support multiple skinned meshes on a single object
- Added Recipes property to InventoryCog
- Added useAutoPickup to InventoryCog
- Added debug view to InventoryCog
- Added GetGroupedCounts to InventoryCog
- Added GetItemsWithTagValue to InventoryCog
- Added optional spawnDrop=true parameter to DropItem in InventoryCog
- Added CraftingFailed event to InventoryCog
- Added Container section to InventoryCog
- Added RemoveItem method to InventoryCog
- Added Breakdown Category to InventoryItem
- Added CustomTags to InventoryItem
- Added GetCustomTag to InventoryItem
- Added SetCustomTag to InventoryItem
- Added Preview Object and Preview Scale to InventoryItem
- Added freeSlotWhenEquipped to InventoryItem
- Added successChance to CraftingRecipe
- Added failed result to CraftingRecipe
- Added failResult to CraftingRecipe
- Replaced InventoryMasterList scriptable object with InventoryDB monobehaviour (InventoryMasterList still exists for easy transfer)
- Moved recipes list to InventoryDB



- Added scroll views to long lists in editors
- Added IsLocked property to BasicCamera
- Updated InventoryStateLoad to check if file exists
- Updated InventoryCog.Craft overload to accept addToInventory and removeFromCounts optional parameters
- Updated BasicPersonController to set input to zero when input locked
- Added InputLocked to BasicCamera
- Updated BasicCamera to prevent movement while locked
- Corrected a wait for zero glitch in UIGrid navigation
- Completely redesigned UI list and grid components
- Added filtering by categories and tags to new List and Grid components
- Added CustomTagFilter object for use with new UI components
- Added Vertical and Horizontal properties to Padding (get)
- Added ContainerMenuUI MonoBehaviour
- Added Display Name to InventoryContainer
- Added TransferMenuOpen and TransferMenuClose methods to InventoryCog
- Added TriggerByButton and TriggerByKey to shared scripts
- Added PaddingDrawer to shared
- Fixed issue where SkinnedEquipPoint would fail if bone order different between source and target
- Added SkinnedBoneRemapper MonoBehaviour
- Added ClearItems method to InventoryContainer
- Added ItemMenuUI and ItemMenuClient MonoBehaviours
- Added RecipeUI MonoBehaviour
- Added CountSelectUI MonoBehaviour
- Added InventoryComponentList MonoBehaviour
- Added Condition Slider to ItemUI
- Updated save/load to include custom stats & tags (backwards compatible)
- Added MessageList MonoBehaviour to Shared
- Added MessageListItem MonoBehaviour to Shared
- Added InventoryMessageListClient MonoBehaviour
- Added Hide if Count < 2 to ItemUI
- Added Rarity Indicator to ItemDetailUI
- Added ItemPreviewUI MonoBehaviour
- Added Item Preview to ItemDetailUI
- Added RecipeItemGrid MonoBehaviour

## Fixes

- Corrected issue when attempting to equip null item
- Fixed load issues in InventoryCog
- ItemsUI replaced by InventoryItemList
- Item Description now shows up on Item UI editor

## **Breaking Changes**

- ItemsGridUI replaced by InventoryItemGrid
- InventoryScrollListUI replaced by InventoryItemScrollList

## Version 1.3

### **Improvements**

- Added RarityColorIndicator MonoBehaviour
- Added RarityColorIndicator to ItemUI
- Added onLoadedItem event to ItemUI
- Added EquipPointUI MonoBehaviour
- Added freeSlotWhenEquipped to InventoryItem
- Added GetCategory to InventoryCog

### **Fixes**

- Fixed compatibility issue with Stats Cog

## Version 1.2

### **Improvements**

- Lowered minimum version to 2018.4.0f1
- Moved Category from UI view to General view for Inventory Item
- Added custom editor for Inventory Container

### **Fixes**

- Fixed expand/collapse issue with category groups
- Fixed onClick display on Item UI

## Version 1.1

### **Improvements**

- Updated demos with Item Detection (Demo 1) and Storage Transfer (Demo 2)
- Added Custom Stats list to Inventory Item

- Added version number to save file to prevent future breaking changes
- Added GetCustomStatValue and SetCustomStatValue to InventoryItem
- Added InventoryMasterList to make it easier to use multiple Inventory Cog instances
- Added Container item type and associated behavior to InventoryItem
- Added InventoryContainer for chests, boxes, etc
- Added TriggerOnHasItem MonoBehaviour
- Added onLoot event to LootItem
- Added GetItems to InventoryCog
- Added AnimatorMods to InventoryItem

### **Fixes**

- Removed unused Shared directory items
- EquipPoint events will now fire correctly

### **Breaking Changes**

- Removed damageMod and healthMod properties from InventoryItem
- Removed availableCategories and availableItems from InventoryCog, replaced by masterList
- Save/Load format changed, old saved data will not load