nullsave

Contents

Introduction

Stats Cog™ is designed to allow you to create interdependent, formula-based stats and effects for your characters and objects without coding. The system is very efficient calculating the value of each Stat at start up and storing the result for easy access. The values are only re-evaluated when they or one of their referenced Stats change with little to no overhead.

Take for example the formula used by the Accuracy Stat in the **RPG Stats Demo** scene. The *Value* is set to "2.5 + OBS + 0.185 * Level". When the instance of **Stats Cog™** starts, it will look to set the current and base minimum, maximum and actual values for the Stat. In this case we'll need the values from two other Stats, namely OBS and Level.

Stats Cog™ will now look through its list of Stats to find those dependances and do two things: first, it will get the value of that Stat (calculating it first if it hasn't already been) and second, subscribing to changes in that Stat's value.

Once all of this is done the result gets stored into publicly available floats properties for you or other Stats to access without having to go through the evaluation process again.

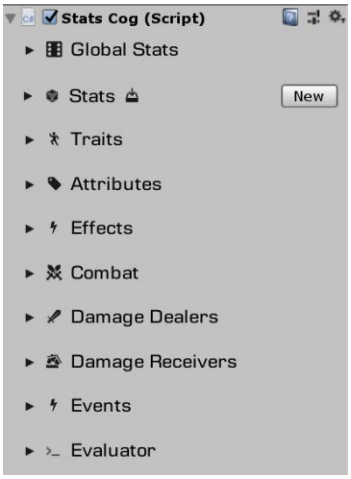
Anytime one of the values of a Stat changes, either from effects or other intervention, that Stat fires off an event. This way any other Stats they rely on the Stat that has just changed can also update their values. They too will fire off events notifying their change. In this way you can have Stats with unlimited interdependencies.

Stats can also be modified either temporarily or permanently by effects. For this reason, a second set of values is also stored by each Stat holding the base minimum, maximum and current values. In other words, the value of the Stat *without* any of the effects considered.

Having both sets (with and without effect modifiers) is useful for RPG type games where you may wish to display those items separately or even take the difference of them to show the overall cumulative change to the character's base stats.

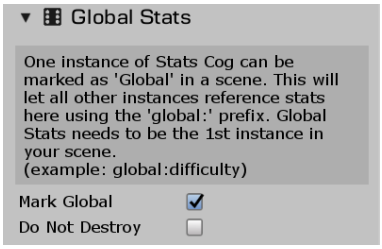
Quick Start

Stats Cog™ is designed to allow you to create interdependent, formula-based stats and effects for your characters and objects without coding. To begin simply add a **Stats Cog™** component to your desired Game Object.



The inspector is grouped into seven collapsible sections: **Stats** (holding stat values for your character such as Health, Level, etc.), **Effects** (which contains all available effects as well as instance specific starting effects and resistances), **Combat** (which controls character health during combat), **Damage Dealers** & **Damage Receivers** (which display child objects attached to dealing or receiving damage respectively), **Events** (which displays events you can subscribe to) and **Evaluator** (which allows you to evaluate expressions at any time).

Global Stats

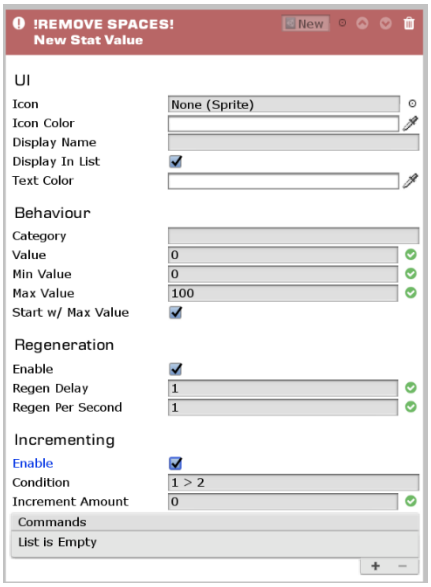


Mark As Global will set any instance of Stats Cog so that it can be addressed by the *global:* prefix. This is useful if you have stats that you want to have available for your entire level or even your game. And example of this would be a “Difficulty” setting.

Do Not Destroy instructs Unity to keep this instance of Stats Cog available between different scenes. Doing so ensures the same instance is available everywhere in your project.

Stat Values

You can create a new Stat by clicking the “New” button next to *Stats*. This creates a new Scriptable Object and automatically assigns it to this instance of the **Stats Cog™**. The same Stat can be used in as many instances as you like. Despite being Scriptable Objects, they are instantiated separately and will maintain their values independently from other instances.



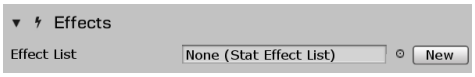
When naming your asset be sure not to include any spaces or special characters (alphanumeric only). You will use this name to find and address your stat. Stats with invalid names or other errors will have a warning icon and you will not be able to collapse the Stat in the view until all issues are corrected.

You can create as many Stats as you like for your character here and order them as well. The value fields are all text and can be set with static numbers or you can use formulas like “HP + (2.5 * Level)” and reference as many other Stats as you like.

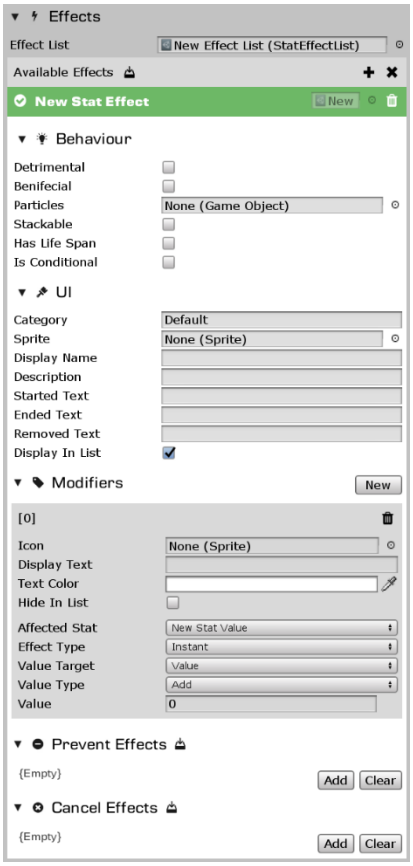
For in-depth coverage of each field and available methods see the Stat Values section.

Effects

Adding effects is an essential part of any stats system. These allow characters to react to things like spells, combat, consumables, and environmental changes. Since, unlike Stat Values, these are usually common to *all* characters the entire set of possible effects are contained in a single list which can be easily added to multiple instances of **Stats Cog™** without having to set them up more than once.



When you first add the component the Effect List will be empty. You can select one with Unity or, if this is your first instance, you can quickly create one by clicking the "New" button and it will automatically be added once created.



Once you have an Effect List created/assigned you can start creating and managing your effects. You can drag and drop existing effects in the top bar (where it says "Available Effects"), create a new effect and automatically have it added by clicking the plus symbol, clear all effects from the list by clicking the "X" or remove an effect by clicking the trash can symbol on any effect.

The focus of most effects are the Modifiers. Each effect can have as many modifiers as needed and each modifier can affect any Stat in one of several ways.

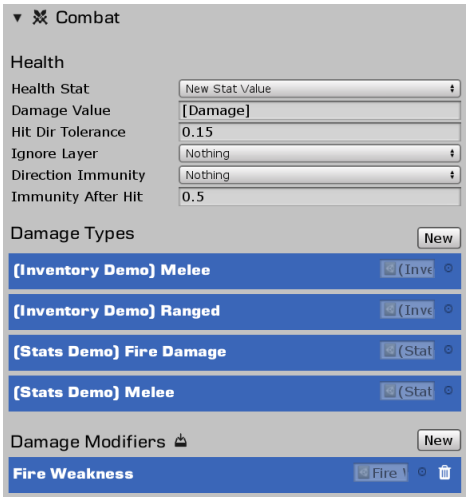
Instant modifiers apply some change to their target Stat the moment the Effect takes hold. These changes do **not** undo after the effect ends, nor do they show as part of active modifiers.

Recurring modifiers apply some change repeatedly over time to their target Stat. These changes continue to happen for as long as the Effect is alive. The value is applied partially each update. These changes do **not** undo after the effect ends, nor do they show as part of active modifiers.

Sustained modifiers are the only ones that undo at the end of the affect and show inside active modifiers. This value, whether an addition, subtraction, etc., is done immediately when the Effect starts and then removed entirely once the effect ends.

For in-depth coverage of each field and available methods see the Stat Effects section.

Combat



Another integral part of a stat system is combat. This section allows you set which value controls your character's health as well as how damage is applied, immunities, etc.

Health Stat – Stat value to use for character's health

Damage Value – Value to use when applying damage to character. `[Damage]`, the default value, is equal to base damage + any weaknesses – any resistances

Hit Dir Tolerance – Is a tolerance value used when calculating from which direction the character was hit

Ignore Layer – Any damage on the layer(s) selected here will be ignored completely

Direction Immunity – Damage dealt from any direction(s) selected here will be ignored completely

Immunity After Hit – Number of seconds after receiving damage that the character should be immune before receiving damage again

Damage Types – Lists **every** damage type found in the project. You can create and modify damage types here. These are simple text/image combos used by Damage Modifiers.

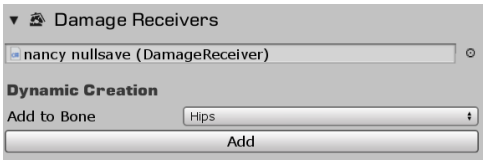
Damage Modifiers – List of resistances, weaknesses or even immunities to different types of damage. For example, your character could be immune to all physical attacks but take double damage from any magical source.

Damage Dealers

To damage other characters, you'll need Damage Dealers. This could be applied to the entire character where if you touch them, oops, that hurt. Or they could be applied to fists, feet, weapons, etc. Each dealer can be turned on or off.

If your Stats Cog is attached to a human rig, you'll be able to automatically create Damage Dealers on any bone you like. For detailed information please see the Damage Dealers section.

Damage Receivers



To receive damage from other sources, you'll need Damage Receivers. This could be applied to the entire character or to specific points on the character.

If your Stats Cog is attached to a human rig, you'll be able to automatically create Damage Receivers on any bone you like. For detailed information please see the Damage Receivers section.

Events

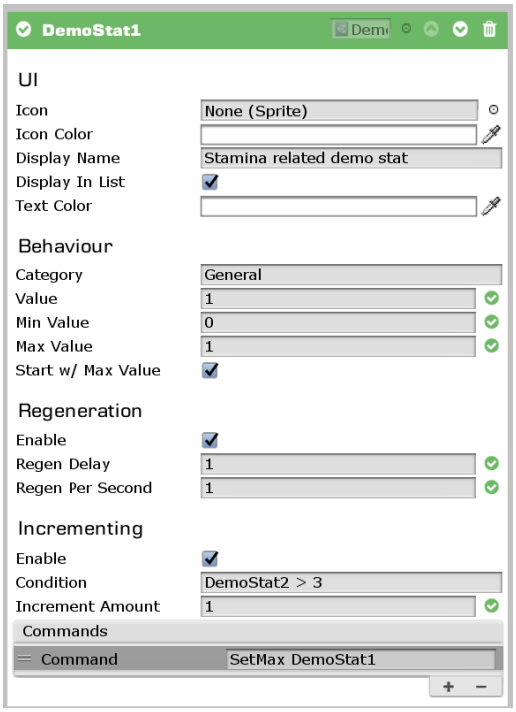
Stats Cog exposes several events to let you know what is happening to your character. Those that are most useful for UI or other reactions are displayed in the inspector. For a complete list please see the Stats Cog API section.

Evaluator

The evaluator is a built-in tool that allows you to evaluate any expression either at design or runtime.

Stat Values

Representing the current values of different Stats for your character Stat Values are the main driving force behind **Stats Cog™**.



Icon – Sprite used for display in the UI for this Stat

Icon Color – Color used to display the icon in the UI

Display Name – Name to display in the UI for this Stat

Display In List – This Stat will only appear in the UI when this option is checked. The Stat will still function even if not displayed in the UI

Text Color – Color used for the Display Name in the UI

Category – Plain text field that can be used in the UI to filter items that are displayed

Value – Represents the starting value of the Stat. If this is a numerical value it can be targeted by effects directly, alternatively you can specify a formula such as "Level + 0.5" and the value will automatically be updated whenever the value for Level changes

Min Value – Represents the minimum value for the Stat and follows the same rules as Value.

Max Value – Represents the maximum value for the Stat and follows the same rules as above.

Start w/ Max Value – If checked the value will automatically be set to be equal to the maximum value at startup. This only works for numeric values.

Enable (Regeneration) – If checked this Stat can regenerate up to its maximum value over time.

Regen Delay – Number of seconds to wait after the last time the value decreased before beginning regeneration

Regen Per Second – Amount to add to the value (up to the maximum value) per second while regeneration is active.

Enable (Incrementing) – If checked the value will automatically increase itself when a certain condition is met.

Condition – True/False evaluation that must be met to trigger incrementing

Increment Amount – Amount to add to the value when triggered

Commands – List of commands to sent to Stats Cog when increment is triggers. (For list of commands see Stats Cog API section)

Properties

```
public List<StatModifier> ActiveModifiers { get; }
```
Returns a list of active modifiers on this Stat

```
public float CurrentBaseMaximum { get; }
```
Current maximum value (without modifiers)

```
public float CurrentBaseMinimum { get; }
```
Current minimum value (without modifiers)

```
public float CurrentBaseRegenAmount { get; }
```
Current regeneration amount (without modifiers)

```
public float CurrentBaseRegenDelay { get; }
```
Current regeneration delay (without modifiers)

```
public float CurrentBaseValue { get; }
```
Current value (without modifiers)

```
public float CurrentMaximum { get; }
```
Current maximum value (with modifiers)

```
public float CurrentMinimum { get; }
```
Current minimum value (with modifiers)

```
public float CurrentRegenAmount { get; }
```
Current regeneration amount (with modifiers)

```
public float CurrentRegenDelay { get; }
```
Current regeneration delay (with modifiers)

```
public float CurrentValue { get; }
```
Current value (with modifiers)

```
public float Initialized { get; }
```
Returns true if the Stat has been initialized by **Stats Cog™**

```
public float Parent { get; }
```
Instance of **Stats Cog™** that owns this stat

Methods

```
public void AddInstantModifier(StatModifier mod, int count)
```
Add an instant modifier [count] times

```
public void AddModifier(StatModifier mod)
```
Add a new modifier

```
public void Initialize(StatsCog owner)
```
Initialize Stat Value (Internal use only)

```
public float GetModifierChange(StatModifier original, StatModifier replacement)
```
Calculate the change in value when replacing one modifier with another

```
public void Load(Stream stream, float version)
```
Load StatValue data from stream

```
public void RemoveModifier(StatModifier mod)
```
Removes modifier from StatValue

```
public void Save(Stream stream)
```
Save StatValue data to stream

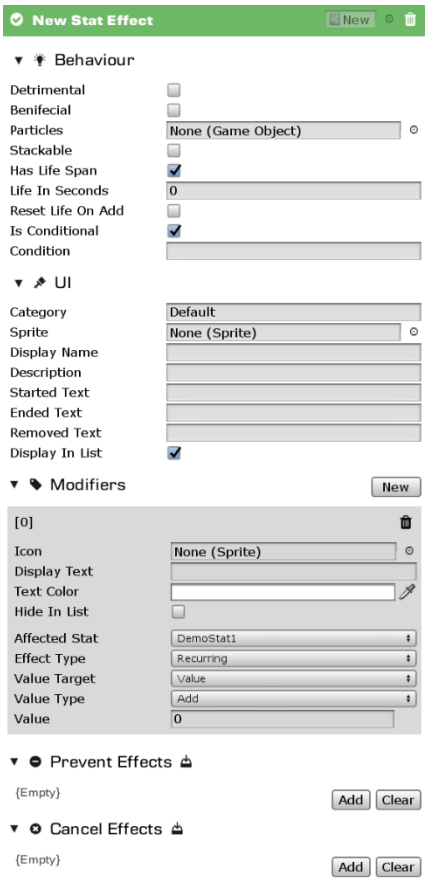
```
public void SendCommand(string command)
```
Relay command to parent for evaluation

Events

```
onBaseMaxValueChanged <float, float>
Raises when the base maximum value changes

onBaseMinValueChanged <float, float>
Raises when the base minimum value changes

onBaseValueChanged <float, float>
Raises when the base value changes

onInit
Raises when the stat is first initialized

onMaxValueChanged <float, float>
Raises when the maximum value changes

onMinValueChanged <float, float>
Raises when the minimum value changes

onValueChanged <float, float>
Raises when the value changes
```

Stat Effects

Stat Effects are the worlds way of interacting with and influencing your stats.



Detrimental – Flag denoting the effect as Detrimental

Beneficial – Flag denoting the effect as Beneficial

Particles – Game Object to spawn when the effect starts

Stackable – If checked a target can have multiple copies of the same effect active at once. Otherwise only one instance will be allowed at a time

Has Life Span – If checked effect has a set amount of time before it is automatically removed

Life In Seconds – Number of seconds after the effect starts before it is removed

Reset Life On Add – In the event that an effect is **not** stackable and **also** has a life span, checking this box will reset the amount of time the effect has left before being removed if another copy of the same effect tried to apply before the current instance is removed

Is Conditional – If checked a specific condition must be met or else the effect will not be applied

Condition – True/False condition that must be met. For example, HP > 12

Category – Simple text field useful for filtering list of effects in the UI

Sprite – Sprite to display in the UI for this effect

Display Name – Text to display for this effect in the UI

Description – Description of this effect to display in the UI

Starting Text – Text to display when the effect first start

Ended Text – Text to display when the effect ends on its own

Removed Text – Text to display when the effect is removed by some other means than its lifespan ending

Display In List – Effect will only display in UI if this option is checked (Effect will continue to function regardless of whether it is displayed in the UI)

Modifiers – List of modifiers applied by this effect. These are the main function of effects.

> **Icon** – Icon to display for the modifier in the UI
>
> **Display Text** – Text to display for the modifier in the UI
>
> **Text Color** – Color to use for the Display Text

Hide In List – If checked this modifier will not appear in the UI (but will still function)

Affected Stat – Stat affected by this modifier

Effect Type – Type of effect to apply with this modifier

Instant – Applies the change immediately and is **never** removed

Recurring – Applies the change repeatedly during the lifespan (applied as Value / Time.deltaTime each update)

Sustained – Applies the change immediately and is removed when the effect ends.

Value Target – Which value should this effect target

Value, Minimum Value, Maximum Value, Regen Delay, Regen Amount

Value Type – How should the modifier's value be applied to the target's value

Add – `TargetValue = TargetValue + ModifierValue`

Add Multiplier – `TargetValue = TargetValue + (TargetValue * ModifierValue)`

Subtract – `TargetValue = TargetValue - ModifierValue`

Subtract Multipler – `TargetValue = TargetValue - (TargetValue * ModifierValue)`

Value Set – `TargetValue = ModifierValue`

Prevent Events – Effects placed in this list will not be allowed to take effect while this Effect is active. If an Effect in this list is *already* active, it will be cancelled.

Cancel Events – If an Effect in this list is *already* active, it will be cancelled. It will **not** however stop a new instance from being added.

Properties

```
public bool IsDead { get; }
```
Returns true if the effect has passed its maximum lifespan in seconds

```
public float RemainingTime { get; set; }
```
Gets/Sets the number of seconds remaining before the effect should expire

Methods

```
public void Initialize()
```
Initialize effect

```
public bool IsEffectCancelled(string effectName)
```
Check if an effect should be cancelled by this effect

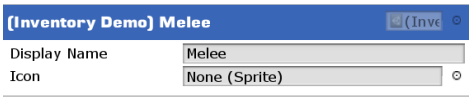
```
public bool IsEffectPrevented(string effectName)
```
Check if an effect should be prevented by this effect

```
public void Update()
```
Update remaining life of effect

Events

```
onEnd
Raises when the effect ends

onStart
Raises when the effect begins
```

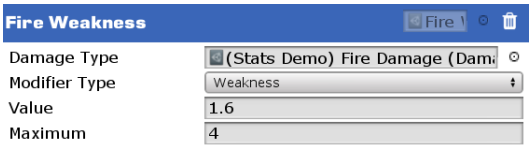
Damage Types



Display Name – Name to display for damage type

Icon – Sprite to display for damage type

Damage Modifiers



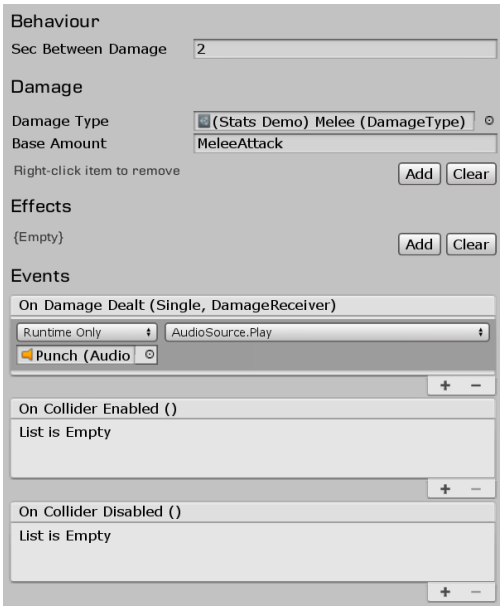
Damage Type – Type of damage
Modifier – Resistance or Weakness
Value – Amount to modify damage
Maximum – Maximum mod with other effects.

Damage Receiver

The Damage Receiver is a component that handles receiving damage for an object. Usually placed at the root of the character.

On Take Damage – Fired when damage is taken

Damage Dealer



The Damage Dealer is a component responsible for sending damage to another **Stats Cog™**. You will also need to add some form of Collider or Collider2D.

Sec Between Damage – Seconds to wait after dealing damage before dealing damage again.

Damage Type – Type of damage to deal

Base Amount – Amount of damage to deal (before weaknesses and resistances)

Effects – Effects to add when dealing damage

On Damage Dealt – Raises when damage is dealt to another source

On Collider Enabled – Raises when the attached collider become enabled

On Collider Disabled – Raises when the attached collider becomes disabled

Stats Cog API

Properties

```
public List<StatEffect> Effects { get; }
```
Returns a list of active status effects

```
public StatsCog LastDamageSource { get; }
```
Returns the last source to deal damage to this instance

```
public List<StatValue> Stats { get; }
```
Returns a list of active stats

Methods

```
public void AddEffect(string effectName)
public void AddEffect(StatEffect effect)
```
Add a new active effect

```
public void AddInventoryEffects(Inventory.InventoryItem item)
public void AddInventoryEffects(Inventory.InventoryItem item, int count)
```
Add effects from inventory item (requires Inventory Cog)

```
public void ClearEffects()
```
Clear all active effects

```
public bool EvaluateCondition(string expression)
```
Evaluate if an expression is true

```
public float GetExpressionValue(string expression)
```
Returns the value of an expression

```
public List<string> GetSubscriptionRequirements(string equation)
```
Get a list of stats needed for equation

```
public List<DamageModifier> FindDamageModifiers(string modifierName)
public List<DamageModifier> FindDamageModifiers(DamageModifier modifier)
```
Get a list of matching modifiers

```
public StatValue FindStat(string statName)
```
Returns active instance of a stat

```
public void Load(string filename)
public void Load(Stream stream)
```
Load saved data

```
public void EndEffect(string effectName)
public void EndEffect(StatEffect effect)
```
End an active effect

```
public List<StatEffect> GetEffectsByCategory(string categoryName)
```
Get a list of active effects by category

```
public float GetModifierChange(StatModifier original, StatModifier replacement)
```
Calculate the change in value when replacing one modifier with another

```
public List<StatValue> GetValuesByCategory(string categoryName)
```
Get a list of stat values by category

```
public void RemoveBenificialEffects()
```
Remove all active effects flagged as benificial

```
public void RemoveEffectsByCategory(string category)
```
Remove all active effects in a category

```
public void RemoveEffect(string effectName)
public void RemoveEffect(StatEffect effect)
```
Remove an active effect

```
public void RemoveEffectAll(string effectName)
```
Remove all active effects

```
public void RemoveDetrimentalEffects()
```
Remove all active effects flagged as detrimental (negative/harmful)

```
public void RemoveInventoryEffects(Inventory.InventoryItem item)
```
Remove effects caused by inventory item (requires Inventory Cog)

```
public void RemoveNeutralEffects()
```
Remove all affects without benificial or detrimental flags

```
public void Save(string filename)
public void Save(Stream stream)
```
Save state

```
public void SendCommand(string command)
```
Send a command to StatsCog

Known Commands

Add
Parameters: Effect Name
Description: Adds the effect supplied

Clear
Parameters: Effect Name
Description: Remove all current effects

Max
Parameters: Stat Name
Description: Sets the stat's maximum value

Max
Parameters: Stat Name
Description: Sets the stat's minimum value

Remove
Parameters: Effect Name
Description: Removes the effect supplied

RemoveAll
Parameters: None
Description: Remove all current effects

Restore-Min
Parameters: Stat Name, Expression
Description: Sets the stat's value to the expression (only if the current value is lower)

SetMax
Parameters: Stat Name
Description: Sets the stat's value to maximum

SetMax
Parameters: Stat Name, Expression
Description: Sets the stat's value to the expression

```
public void SendCommandToLastDamageDealer(string command)
```
Send a command to the last StatsCog that dealt damage to this one

```
private void TakeDamage(DamageDealer damageDealer, GameObject damageSourceObject)
```
Take damage

```
public void UpdateDamageDealers()
```
Update all damage dealers in children

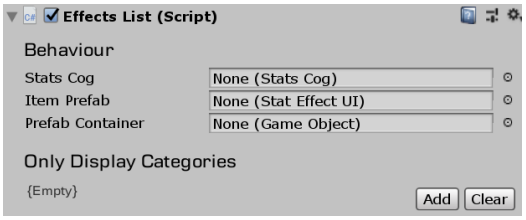
```
public bool ValidateExpression(string expression)
```
Check if an expression is valid

UI Components

Stats Cog™ comes with several UI components to help you visualize your stats.

Effects List

Displays a list of active effects.



Stats Cog – Instance providing the list of effects

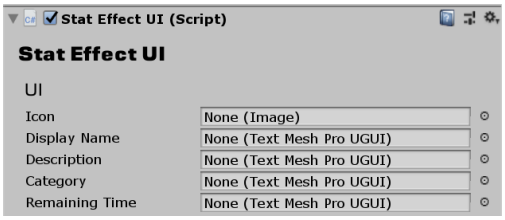
Item Prefab – Prefab to create for each effect

Prefab Container – RectTransform to place prefab instances in. (Add a Layout Group to help make the display nicer)

Only Display Categories – List of categories to restrict display to (effects in all categories will display if this list is empty)

Stat Effect UI

Used on prefabs to display effect data.



Icon – Image to use when displaying the Effect icon/sprite

Display Name – Text Mesh Pro target for effect's display name

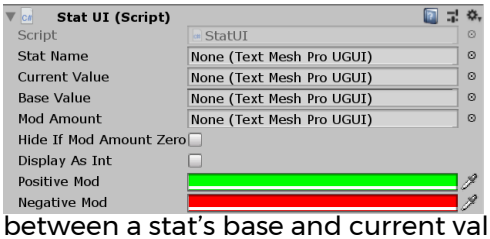
Description – Text Mesh Pro target for effect's display description

Category – Text Mesh Pro target for effect's category

Display Name – Text Mesh Pro target the time left (in seconds) before the effect expires

Stat UI

Used on prefabs to display stat data.



Stat Name - Text Mesh Pro target for stat's display name

Current Value - Text Mesh Pro target for stat's value

Base Value - Text Mesh Pro target for stat's base value

Mod Amount - Text Mesh Pro target for difference between a stat's base and current value

Hide if Mod Amount Zero – Hides the "Mod Amount" if the value is zero when checked

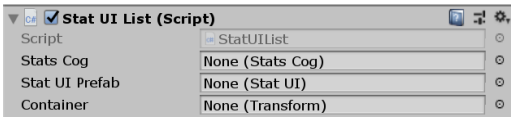
Display As Int – Converts values to Integers when checked

Positive Mod – Color to apply to "Mod Amount" when value is greater than 0

Negative Mod - Color to apply to "Mod Amount" when value is less than 0

Stat UI List

Displays a list of stats



Stats Cog – Instance providing the list of stats

Stat UI Prefab – Prefab to create for each stat

Container – RectTransform to place prefab instances in. (Add a Layout Group to help make the display nicer)

Change Log

2021.Q3.R1

Changes

- Removed "Stats & Effects Editor"
- Added ValidateExpression to StatsCog
- Removed Debug view from StatsCog
- Added tooltips for StatsCog
- Updated StatsCog Editor UI
- Added custom editor for Effects List
- Added ability to filter Effects List by category

Fixes

- Updated to work with negative values

Breaking Changes

- Removed "Effect Display" use "Stat Effect UI" instead

Version 1.12

Improvements

- Added Projectile and Projectile Launcher components
- Added Damage Shield
- Added Hit Dir Tolerance to Stats Cog
- Performance tweaks
- Added Stats & Effects Editor for streamlined creation
- Added Stats UI List component
- Added Stat UI component
- Added Display In List to StatValue

Version 1.10

Improvements

- Additional editor updates
- Compatibility update
- Added OnDeath event to Stats Cog
- Added SendCommandToLastDamageDealer to Stats Cog
- Updated demo to award XP on kills & changed XP level requirements
- Added reprocess check to StatValue auto increment

- Re-added Burn effect to fireball
- Re-added RPG Demo
- Hierarchy Icons can now be enabled/disabled from Tools/NullSave menu

Version 1.9

Breaking Changes

- Changed Damage Resistance/Weakness options to DamageModifier ScriptableObject
- Added DamageType ScriptableObject
- Updated HitDirection enum values to be compatible with bitmasking
- Removed Damage Resistance/Weakness from Stats Cog, replaced with Damage Modifiers
- Changed damageType from string to DamageType in Damage
- Removed FindResistence and FindWeakness

Fixes

- Fixed issue where effects with no modifiers but has cancel/remove were not added to active
- Fixed bug where current modifiers were not ended on Load
- Fixed issue where stats did not honor an Instant modifier at the same time as a Sustained modifier

Updates

- Added Direction Immunity to Stats Cog
- Added Effect List component
- Added Effect UI component
- Streamlined editor GUI
- Added Damage Dealers/Receivers list to Stats Cog editor
- Changed "Health Stat" to be a dropdown with all available stats
- Improved save/load

Version 1.8

Improvements

- StatsCog no longer derives from Damage Receiver
- StatsCog now subscribes to all Damage Receiver children
- Added restore-min command
- Fixed issue with Category not persisting for StatValues
- Fixed issue with inspecting StatsCog prefab at runtime
- Added more custom editors
- Additional 2D support

- Added welcome screen
- Removed DataStore from Shared
- 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 ExpressionSubcription class
- Added GetSubscriptionRequirements method to StatsCog

Version 1.6

Improvements

- Added Immunity After Hit to Stats Cog
- Added HitDirection enum (FrontLeft,Front,FrontRight,Left,BackLeft,Back,BackRight,Right)
- Added onHitDirection event to StatsCog (raises direction of impact even if no damage taken)
- Added onHitDamageDirection event to StatsCog (raises direction of impact when damage is taken)
- Added resetLifeOnAdd to StatEffect

Version 1.5

Updates

- Updated Editors

Version 1.4.1

Updates

- Updated compatibility with Inventory Cog 1.3

Version 1.4

Improvements

- Lowered minimum version to 2018.4.0f1
- Added Starting Effects to StatsCog
- Created new demo with "Survival" stats and combat
- Moved "Hierarchy Icons" into Shared directory
- Updated custom editors
- Regen now uses Time.deltaTime to pause regen in menus, etc
- Added StatMonitorTMP MonoBehaviour
- Added custom editor to TextStatMonitor
- Added Effect Resistance to StatsCog
- Added onEffectResist event to StatsCog
- Added TriggerByCondition MonoBehaviour
- Made Debug the default tab while player is running for StatsCog
- Added Stat Effect List to streamline adding effects to multiple StatCogs
- Replaced Available Effects with Stat Effect List on StatsCog
- Added Events view to StatsCog
- Added UI fields to StatValue (icon, iconColor, displayName, textColor)
- Added TriggerByEffect MonoBehaviour

Fixes

- Fixed issue where instant negative modifiers didn't apply properly in some cases
- Fixed issue where negative modifers didn't reset RegenDelay
- Fixed issue with expanding/collapsing sections in custom editors
- Fixed issue in custom editor when inspecting a disabled item at runtime in debug view

Version 1.3

Improvements

- Updated editor to make Commands a reorderable list for StatValues
- Added GetEffectsByCategory to StatsCog
- Added GetValueByCategory to StatsCog
- Added version to Save/Load data to prevent future breaking changes
- Added Damage class to Shared
- Added DamageDealer MonoBehaviour to Shared
- Added DamageReceiver MonoBehaviour to Shared
- Added DamageResistance class to Shared
- Added DamageWeakness class to Shared
- Added StatDamageResistence
- Added StatDamageWeakness
- Updated StatsCog to derive from DamageReceiver
- Added resistances to StatsCog
- Added weaknesses to StatsCog
- Added FindResistance to StatsCog

- Added FindWeakness to StatsCog
- Updated AddEffect to apply to resistances and weaknesses
- Added new Initialize overloads to StatModifier for resistances and weaknesses
- Added new CalculateAppliedValue overloads to StatModifer for resistances and weaknesses
- Added new GetSustainedValue overloads to StatModifier for resistances and weaknesses
- Added UpdateDamageDealers to StatsCog
- Added healthStat and damageValue to StatsCog
- Added onDamageTaken and onImmuneToDamage events to StatsCog

Fixes

- Corrected Category to appear in editor for StatValues and StatEffects
- Corrected editor issue displaying Modifiers

Breaking Changes

- Default save/load structure changed, current saved data will not load properly. Replace with new save.

Version 1.2

Improvements

- Added ability to regen by percent
- Added ability to increment by percent
- Added category to StatEffects
- Added RemoveEffectsByCategory
- Added category to StatValues (game use only, does not affect StatsCog)
- Added icon, displayText, textColor, and showInList to StatModifier
- Added GetModiferChange method to StatsCog and StatValue
- Added AddInventoryMods and RemoveInventoryMods to StatsCog
- Added support for Inventory Cog
- Added CalculateAppliedValue to StatModifier
- Changed ShowInList to HideInList on StatModifer
- Added AddInstantInventoryMods to StatsCog
- Added AddInstantModifer to StatValue

Fixes

- Corrected RemoveNeutrallEffects to RemoveNeutralEffects

Breaking Changes

- Default save/load structure changed, current saved data will not load properly. Replace with new save.

Version 1.1

Improvements

- Added 3rd person demo
- Added onEffectAdded event to StatsCog
- Added onEffectEnded event to StatsCog
- Added onEffectRemoved event to StatsCog
- Added EndEffect methods to StatsCog
- Added addedText, endedText, and removedText to StatEffect
- Added effectParticles to StatEffect
- Added isDetrimental to StatEffect
- Added isBenificial to StatEffec
- Added isRemoveable ot StatEffect
- Added RemoveBenificialEffects to StatsCog
- Added RemoveDetrimentalEffects to StatsCog
- Added RemoveNeutralEffects to StatsCog
- Added incrementCommand to StatValue
- Added a debug editor that allows you to see current and base values, active effects and send commands
- Updated all StatValue Changed events to return old and new values
- Replaced EditorIcons with our new HeirarchyIcons
- Re-ordered file structure
- Changed LogicExtensions namespace to NullSave.TOCK.Stats
- Moved TextStatMonitor and SliderStat into main project
- Added Hierarchy icons to TextStatMonitor and SliderStat
- Made StatValue's increment command an array to allow for running multiple commands on increment

Fixes

- Fixed issue where changing a StatValue on one GameObject affected the same StatValue on other GameObjects
- Fixed issue that prevented stacking effects.
- Fixed an issue that allowed stat values to regenerate too quickly
- Fixed issue that allowed stat value to be set outside of min/max values
- Fixed issue where StatValues would subscribe multiple times if the same StatValue was used multiple times in an equation
- Fixed issue when removing a modifier that would have pushed value beyond maximum value is calculated incorrectly
- Fixed issue where RegenAmount and RegenDelay sustained effects were not honored.

Breaking Changes

- StatsCog.stats should no longer be reference directly, please use StatsCog.Stats
- StatsCog.ActiveEffects has been renamed StatsCog.Stats