

# Namespace ElectricDrill

## Structs

[StatChangeInfo](#)

# Struct StatChangeInfo

Namespace: [ElectricDrill](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public struct StatChangeInfo
```

## Constructors

StatChangeInfo(EntityStats, Stat, long, long)

```
public StatChangeInfo(EntityStats entity, Stat stat, long oldValue, long newValue)
```

### Parameters

entity [EntityStats](#)

stat [Stat](#)

oldValue long

newValue long

## Fields

EntityStats

```
public EntityStats EntityStats
```

### Field Value

[EntityStats](#)

NewValue

```
public long NewValue
```

Field Value

long

OldValue

```
public long OldValue
```

Field Value

long

Stat

```
public Stat Stat
```

Field Value

Stat

# Namespace ElectricDrill.SimpleRPGCore.SimpleRpgCore.Utils

## Classes

### [Cache<KType, VType>](#)

A generic cache class that provides basic caching functionality.

# Class Cache<KType, VType>

Namespace: [ElectricDrill.SimpleRPGCore.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

A generic cache class that provides basic caching functionality.

```
public class Cache<KType, VType>
```

## Type Parameters

### KType

The type of the keys in the cache.

### VType

The type of the values in the cache.

## Inheritance

object ← Cache<KType, VType>

# Properties

## this[KType]

Gets or sets the value for the specified key in the cache.

```
public VType this[KType key] { get; set; }
```

## Parameters

### key KType

The key of the item to get or set.

## Property Value

### VType

The value associated with the specified key.

## Methods

### Get(KType)

Gets the value for the specified key from the cache. If the key is not found, a KeyNotFoundException is thrown.

```
public VType Get(KType key)
```

Parameters

**key** KType

The key of the item to get.

Returns

VType

The value associated with the specified key.

Exceptions

KeyNotFoundException

Thrown when the key is not found in the cache.

### Has(KType)

Determines whether the cache contains an item with the specified key.

```
public bool Has(KType key)
```

Parameters

**key** KType

The key to locate in the cache.

Returns

bool

true if the cache contains an item with the key; otherwise, false.

## Invalidate(KType)

Invalidates the cache item for the specified key. If key is not found, nothing happens.

```
public void Invalidate(KType key)
```

Parameters

**key** KType

The key of the item to invalidate.

## InvalidateAll()

Invalidates all items in the cache.

```
public void InvalidateAll()
```

## Set(KType, VType)

Sets the value for the specified key in the cache. UPSERT operation.

```
public void Set(KType key, VType value)
```

Parameters

**key** KType

The key of the item to set.

**value** VType

The value to set for the specified key.

## TryGet(KType, out VType)

Tries to get the value for the specified key from the cache.

```
public bool TryGet(KType key, out VType value)
```

### Parameters

**key** KType

The key of the item to get.

**value** VType

When this method returns, contains the value associated with the specified key, if the key is found; otherwise, the default value for the type of the value parameter.

### Returns

bool

true if the cache contains an item with the specified key; otherwise, false.

# Namespace ElectricDrill.SimpleRpgCore

## Classes

### [BoundedValue](#)

#### [Class](#)

Represents a character class in the RPG system. It can be used to define how attributes and stats grow over levels.

#### [ClassMenuItems](#)

Provides a menu item in the Unity Editor to create a Class asset.

#### [EntityClass](#)

#### [EntityCore](#)

The core component for any entity in the game. It is the mandatory base class for all entities, providing essential functionality. Provides easy access to the entity's level, stats, and attributes.

#### [EntityLevel](#)

Represents the level and experience of an entity.

#### [ExpSource](#)

#### [GrowthFormula](#)

Represents a formula to calculate growth values for different levels, up to a maximum level.

#### [GrowthFormulaMenuItems](#)

Provides a menu item in the Unity Editor to create a GrowthFormula asset.

## Interfaces

### [IAttributes](#)

### [IClassSource](#)

### [IEntityCore](#)

### [IExpSource](#)

### [ILevel](#)

### [ILevelable](#)

### [IStatSetSource](#)

# Class BoundedValue

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public abstract class BoundedValue : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← BoundedValue

## Derived

[Attribute](#), [Stat](#)

## Properties

### Has.MaxValue

```
public bool Has.MaxValue { get; }
```

Property Value

bool

### Has.MinValue

```
public bool Has.MinValue { get; }
```

Property Value

bool

### MaxValue

```
public long MaxValue { get; }
```

Property Value

long

MinValue

```
public int MinValue { get; }
```

Property Value

int

# Class Class

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Represents a character class in the RPG system. It can be used to define how attributes and stats grow over levels.

```
public class Class : ScriptableObject, IStatSetSource
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← Class

## Implements

[IStatSetSource](#)

## Fields

### \_attributeSet

The set of attributes associated with this class.

```
[SerializeField]  
protected AttributeSet _attributeSet
```

### Field Value

[AttributeSet](#)

### \_maxHpGrowthFormula

The growth formula for the maximum HP. This is a baseline value that can be used to calculate the maximum HP of characters of this class at different levels.

```
[SerializeField]  
protected GrowthFormula _maxHpGrowthFormula
```

## Field Value

[GrowthFormula](#)

### \_statSet

The set of stats associated with this class.

```
[SerializeField]  
protected StatSet _statSet
```

## Field Value

[StatSet](#)

## Properties

### AttributeSet

Gets the AttributeSet for this class.

```
public virtual AttributeSet AttributeSet { get; }
```

### Property Value

[AttributeSet](#)

### StatSet

Gets the StatSet for this class.

```
public virtual StatSet StatSet { get; }
```

### Property Value

[StatSet](#)

# Methods

## GetAttributeAt(Attribute, int)

Calculates the value of a specific attribute at a given level.

```
public virtual long GetAttributeAt(Attribute attribute, int level)
```

Parameters

**attribute** [Attribute](#)

The attribute to calculate.

**level** int

The level to calculate the attribute value for.

Returns

long

The value of the attribute at the specified level.

## GetMaxHpAt(int)

Calculates the maximum HP for a given level.

```
public long GetMaxHpAt(int level)
```

Parameters

**level** int

The level to calculate the max HP for.

Returns

long

The maximum HP at the specified level.

## GetStatAt(Stat, int)

Calculates the value of a specific stat at a given level.

```
public virtual long GetStatAt(Stat stat, int level)
```

### Parameters

**stat** [Stat](#)

The stat to calculate.

**level** int

The level to calculate the stat value for.

### Returns

long

The value of the stat at the specified level.

### Exceptions

ArgumentException

Thrown when the growth formula for the stat is not set.

# Class ClassMenuItems

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Provides a menu item in the Unity Editor to create a Class asset.

```
public static class ClassMenuItems
```

## Inheritance

object ← ClassMenuItems

## Methods

### CreateClass()

Creates a new Class asset.

```
[MenuItem("Assets/Create/Simple RPG Core/Class", false, 5)]  
public static void CreateClass()
```

# Class EntityClass

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class EntityClass : MonoBehaviour, IClassSource
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← EntityClass

## Implements

[IClassSource](#)

## Properties

### Class

```
public Class Class { get; }
```

### Property Value

[Class](#)

## Operators

### implicit operator Class(EntityClass)

```
public static implicit operator Class(EntityClass entityClass)
```

### Parameters

entityClass [EntityClass](#)

### Returns



# Class EntityCore

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

The core component for any entity in the game. It is the mandatory base class for all entities, providing essential functionality. Provides easy access to the entity's level, stats, and attributes.

```
public class EntityCore : MonoBehaviour, ILevel, IAttributes
```

Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← EntityCore

Implements

[ILevel](#), [IAttributes](#)

## Properties

### Attributes

Gets the [EntityAttributes](#) component which manages the entity's attributes.

```
public virtual EntityAttributes Attributes { get; }
```

Property Value

[EntityAttributes](#)

### Level

Gets the [EntityLevel](#) class which manages the entity's level and experience.

```
public virtual EntityLevel Level { get; }
```

Property Value

## [EntityLevel](#)

### Name

Gets the name of the entity's GameObject.

```
public string Name { get; }
```

### Property Value

string

### Stats

Gets the [EntityStats](#) component which manages the entity's stats.

```
public virtual EntityStats Stats { get; }
```

### Property Value

[EntityStats](#)

## Methods

### Awake()

```
protected virtual void Awake()
```

### Start()

```
protected virtual void Start()
```

## Update()

```
protected virtual void Update()
```

# Class EntityLevel

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Represents the level and experience of an entity.

```
[Serializable]  
public class EntityLevel : ILevelable
```

Inheritance

object ← EntityLevel

Implements

[ILevelable](#)

## Properties

### CurrentTotalExperience

The current total experience of the entity.

```
public long CurrentTotalExperience { get; }
```

Property Value

long

### Level

The current level of the entity.

```
public virtual int Level { get; set; }
```

Property Value

int

## OnLevelUp

Event raised when the entity levels up. The parameter is the new level.

```
public virtual Action<int> OnLevelUp { get; set; }
```

Property Value

Action<int>

## Methods

### AddExp(long)

Adds experience to the entity. This will be affected by experience gain modifiers. If the entity levels up, the OnLevelUp event will be raised.

```
public void AddExp(long amount)
```

Parameters

**amount** long

The base amount of experience to add.

### CurrentLevelTotalExperience()

Gets the total experience required to reach the current level.

```
public long CurrentLevelTotalExperience()
```

Returns

long

The total experience for the current level.

## NextLevelTotalExperience()

Gets the total experience required to reach the next level. If the entity is already at the maximum level, it returns the total experience for the current level.

```
public long NextLevelTotalExperience()
```

Returns

long

The total experience for the next level. If at max level, returns the total experience for the current level.

## SetTotalCurrentExp(long)

Sets the total current experience and updates the level accordingly.

```
public void SetTotalCurrentExp(long totalCurrentExperience)
```

Parameters

**totalCurrentExperience** long

The total experience to set.

## ValidateExperience()

Validates that the current total experience corresponds to the current level. If not, it resets the experience to the base value for the current level.

```
public void ValidateExperience()
```

# Operators

## implicit operator int(EntityLevel)

Implicitly converts an EntityLevel to its integer level value.

```
public static implicit operator int(EntityLevel entityLevel)
```

### Parameters

**entityLevel** [EntityLevel](#)

The EntityLevel instance.

### Returns

int

# Class ExpSource

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class ExpSource : MonoBehaviour, IExpSource
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← ExpSource

## Implements

[IExpSource](#)

## Properties

### Exp

```
public long Exp { get; }
```

Property Value

long

### Harvested

```
public bool Harvested { get; set; }
```

Property Value

bool

# Class GrowthFormula

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Represents a formula to calculate growth values for different levels, up to a maximum level.

```
public class GrowthFormula : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← GrowthFormula

## Properties

### GrowthFoValues

Gets the pre-calculated growth formula values for each level. Values are refreshed on validation. The array is indexed from 0 to maxLevel - 1, where index 0 corresponds to level 1.

```
public double[] GrowthFoValues { get; }
```

Property Value

double[]

## Methods

### GetGrowthValue(int)

Gets the growth value for a specific level.

```
public long GetGrowthValue(int level)
```

Parameters

**level** int

The level for which to get the growth value.

Returns

long

The growth value for the specified level.

# Class GrowthFormulaMenuItems

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Provides a menu item in the Unity Editor to create a GrowthFormula asset.

```
public static class GrowthFormulaMenuItems
```

## Inheritance

object ← GrowthFormulaMenuItems

## Methods

### CreateGrowthFormula()

Creates a new GrowthFormula asset.

```
[MenuItem("Assets/Create/Simple RPG Core/Growth Formula", false, 6)]  
public static void CreateGrowthFormula()
```

# Interface IAttributes

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IAttributes
```

## Properties

### Attributes

```
EntityAttributes Attributes { get; }
```

### Property Value

[EntityAttributes](#)

# Interface IClassSource

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IClassSource
```

## Properties

### Class

```
Class Class { get; }
```

### Property Value

[Class](#)

# Interface IEntityCore

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IEntityCore
```

## Properties

### Attributes

```
EntityAttributes Attributes { get; }
```

Property Value

[EntityAttributes](#)

### Level

```
EntityLevel Level { get; }
```

Property Value

[EntityLevel](#)

### Name

```
string Name { get; }
```

Property Value

string

## Stats

```
EntityStats Stats { get; }
```

Property Value

[EntityStats](#)

# Interface IExpSource

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IExpSource
```

## Properties

### Exp

```
long Exp { get; }
```

Property Value

long

### Harvested

```
bool Harvested { get; set; }
```

Property Value

bool

# Interface ILevel

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface ILevel
```

## Properties

### Level

```
EntityLevel Level { get; }
```

#### Property Value

[EntityLevel](#)

# Interface ILevelable

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface ILevelable
```

## Properties

### CurrentTotalExperience

```
long CurrentTotalExperience { get; }
```

Property Value

long

### Level

```
int Level { get; set; }
```

Property Value

int

## Methods

### AddExp(long)

```
void AddExp(long amount)
```

Parameters

**amount** long

## CurrentLevelTotalExperience()

**long CurrentLevelTotalExperience()**

Returns

long

## NextLevelTotalExperience()

**long NextLevelTotalExperience()**

Returns

long

## SetTotalCurrentExp(long)

**void SetTotalCurrentExp(long totalCurrentExperience)**

Parameters

**totalCurrentExperience** long

## ValidateExperience()

**void ValidateExperience()**

# Interface IStatSetSource

Namespace: [ElectricDrill.SimpleRpgCore](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IStatSetSource
```

## Properties

### StatSet

```
StatSet StatSet { get; }
```

### Property Value

[StatSet](#)

# Namespace ElectricDrill.SimpleRpgCore. Attributes

## Classes

[Attribute](#)

[AttributeMenuItems](#)

[AttributePointsTracker](#)

[AttributeSet](#)

[AttributeSetInstance](#)

[AttributeSetInstanceExtensions](#)

[AttributeSetMenuItems](#)

[EntityAttributes](#)

Manages the attributes of an entity. This component calculates attribute values by combining base values, flat modifiers, percentage modifiers, and spent attribute points. It requires an ElectricDrill.SimpleRpgCore.Attributes.EntityAttributes.EntityCore component on the same GameObject.

## Structs

[AttributeChangeInfo](#)

## Interfaces

[IAttributeContainer](#)

# Class Attribute

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[Serializable]
public class Attribute : BoundedValue
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [BoundedValue](#) ← Attribute

## Inherited Members

[BoundedValue.HasValue](#) , [BoundedValue.MaxValue](#) , [BoundedValue.HasMinValue](#) ,  
[BoundedValue.MinValue](#)

# Struct AttributeChangeInfo

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public readonly struct AttributeChangeInfo
```

## Constructors

AttributeChangeInfo(EntityAttributes, Attribute, long, long)

```
public AttributeChangeInfo(EntityAttributes source, Attribute attribute, long oldValue,  
                           long newValue)
```

Parameters

source [EntityAttributes](#)

attribute [Attribute](#)

oldValue long

newValue long

## Properties

Attribute

```
public Attribute Attribute { get; }
```

Property Value

[Attribute](#)

## NewValue

```
public long NewValue { get; }
```

Property Value

long

## OldValue

```
public long OldValue { get; }
```

Property Value

long

## Source

```
public EntityAttributes Source { get; }
```

Property Value

[EntityAttributes](#)

# Class AttributeMenuItems

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public static class AttributeMenuItems
```

## Inheritance

object ← AttributeMenuItems

## Methods

### CreateAttribute()

```
[MenuItem("Assets/Create/Simple RPG Core/Attribute ^&A", false, 0)]  
public static void CreateAttribute()
```

# Class AttributePointsTracker

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[Serializable]
public class AttributePointsTracker
```

## Inheritance

object ← AttributePointsTracker

## Properties

### Available

```
public int Available { get; }
```

Property Value

int

### SpentAttributesKeys

```
public Dictionary<Attribute, int>.KeyCollection SpentAttributesKeys { get; }
```

Property Value

Dictionary<[Attribute](#), int>.KeyCollection

## Methods

### AddPoints(int)

```
public void AddPoints(int amount)
```

Parameters

amount int

## GetSpentOn(Attribute)

```
public long GetSpentOn(Attribute attribute)
```

Parameters

attribute [Attribute](#)

Returns

long

## Refund(Attribute)

```
public void Refund(Attribute attribute)
```

Parameters

attribute [Attribute](#)

## Refund(Attribute, int)

```
public void Refund(Attribute attribute, int amount)
```

Parameters

attribute [Attribute](#)

amount int

## RefundAll()

```
public void RefundAll()
```

## SpendOn(Attribute, int)

```
public void SpendOn(Attribute attribute, int amount)
```

### Parameters

attribute [Attribute](#)

amount int

# Class AttributeSet

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class AttributeSet : ScriptableObject, IAttributeContainer
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← AttributeSet

## Implements

[IAttributeContainer](#)

## Properties

### Attributes

```
public IReadOnlyList<Attribute> Attributes { get; }
```

#### Property Value

IReadOnlyList<[Attribute](#)>

## Methods

### Contains(Attribute)

```
public bool Contains(Attribute attribute)
```

#### Parameters

attribute [Attribute](#)

#### Returns

bool

## Get(Attribute)

```
public Attribute Get(Attribute attribute)
```

Parameters

attribute [Attribute](#)

Returns

[Attribute](#)

# Class AttributeSetInstance

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class AttributeSetInstance : IAttributeContainer
```

## Inheritance

object ← AttributeSetInstance

## Implements

[IAttributeContainer](#)

## Constructors

### AttributeSetInstance(AttributeSet)

```
public AttributeSetInstance(AttributeSet attrSet)
```

## Parameters

attrSet [AttributeSet](#)

## Properties

### Attributes

```
public Dictionary<Attribute, long> Attributes { get; }
```

## Property Value

Dictionary<[Attribute](#), long>

## this[Attribute]

```
public long this[Attribute attribute] { get; set; }
```

Parameters

**attribute** [Attribute](#)

Property Value

long

## Methods

AddValue(Attribute, long)

```
public void AddValue(Attribute attribute, long value)
```

Parameters

**attribute** [Attribute](#)

**value** long

Clone()

```
public AttributeSetInstance Clone()
```

Returns

[AttributeSetInstance](#)

Contains(Attribute)

```
public bool Contains(Attribute stat)
```

Parameters

**stat** [Attribute](#)

Returns

bool

## Get(Attribute)

```
public long Get(Attribute attribute)
```

Parameters

**attribute** [Attribute](#)

Returns

long

## GetAsPercentage(Attribute)

```
public Percentage GetAsPercentage(Attribute stat)
```

Parameters

**stat** [Attribute](#)

Returns

[Percentage](#)

## GetEnumerator()

```
public IEnumarator<KeyValuePair<Attribute, long>> GetEnumerator()
```

Returns

IEnumerator<KeyValuePair<[Attribute](#), long>>

## Operators

operator +(AttributeSetInstance, AttributeSetInstance)

```
public static AttributeSetInstance operator +(AttributeSetInstance a,  
AttributeSetInstance b)
```

Parameters

a [AttributeSetInstance](#)

b [AttributeSetInstance](#)

Returns

[AttributeSetInstance](#)

explicit operator

AttributeSetInstance(SerializableDictionary<Attribute, long>)

```
public static explicit operator AttributeSetInstance(SerializableDictionary<Attribute,  
long> dictionary)
```

Parameters

dictionary [SerializableDictionary<Attribute, long>](#)

Returns

[AttributeSetInstance](#)

# Class AttributeSetInstanceExtensions

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public static class AttributeSetInstanceExtensions
```

## Inheritance

object ← AttributeSetInstanceExtensions

## Methods

ToAttributeSetInstance(SerializableDictionary<Attribute, long>, AttributeSet)

```
public static AttributeSetInstance ToAttributeSetInstance(this  
SerializableDictionary<Attribute, long> dictionary, AttributeSet attributeSet)
```

### Parameters

dictionary [SerializableDictionary<Attribute, long>](#)

attributeSet [AttributeSet](#)

### Returns

[AttributeSetInstance](#)

# Class AttributeSetMenuItems

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public static class AttributeSetMenuItems
```

## Inheritance

object ← AttributeSetMenuItems

## Methods

### CreateAttributeSet()

```
[MenuItem("Assets/Create/Simple RPG Core/Attribute Set", false, 1)]  
public static void CreateAttributeSet()
```

# Class EntityAttributes

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Manages the attributes of an entity. This component calculates attribute values by combining base values, flat modifiers, percentage modifiers, and spent attribute points. It requires an ElectricDrill.SimpleRpgCore.Attributes.EntityAttributes.EntityCore component on the same GameObject.

```
[RequireComponent(typeof(EntityCore))]  
public class EntityAttributes : MonoBehaviour
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← EntityAttributes

## Properties

### AttributeSet

Gets the [AttributeSet](#) used by this entity. The source of the attribute set depends on whether ElectricDrill.SimpleRpgCore.Attributes.EntityAttributes.useClassBaseAttributes is true. If true, it returns the attribute set from the [EntityClass](#). Otherwise, it returns the ElectricDrill.SimpleRpgCore.Attributes.EntityAttributes.fixedBaseAttributeSet.

```
public virtual AttributeSet AttributeSet { get; }
```

### Property Value

[AttributeSet](#)

### AttributesCache

Provides access to the cache for calculated attribute values. Caching can be enabled or disabled with the ElectricDrill.SimpleRpgCore.Attributes.EntityAttributes.useCache flag.

```
public Cache<Attribute, long> AttributesCache { get; }
```

## Property Value

[Cache<Attribute, long>](#)

## EntityClass

Gets the class for this entity. The class is used to determine base attributes when ElectricDrill.SimpleRpg.Core.Attributes.EntityAttributes.useClassBaseAttributes is true.

```
public IClassSource EntityClass { get; }
```

## Property Value

[IClassSource](#)

## FlatModifiers

Gets the instance of flat modifiers for the attributes. Flat modifiers are added directly to the base attribute values.

```
protected virtual AttributeSetInstance FlatModifiers { get; }
```

## Property Value

[AttributeSetInstance](#)

## OnAttributeChanged

```
public AttributeChangedGameEvent OnAttributeChanged { get; }
```

## Property Value

[AttributeChangedGameEvent](#)

## PercentageModifiers

Gets the instance of percentage modifiers for the attributes. Percentage modifiers are applied after flat modifiers.

```
protected virtual AttributeSetInstance PercentageModifiers { get; }
```

Property Value

[AttributeSetInstance](#)

## Methods

### AddFlatModifier(Attribute, long)

Adds a flat modifier to an attribute.

```
public void AddFlatModifier(Attribute attribute, long value)
```

Parameters

**attribute** [Attribute](#)

The attribute to modify.

**value** long

The flat value to add.

### AddPercentageModifier(Attribute, Percentage)

Adds a percentage modifier to an attribute.

```
public void AddPercentageModifier(Attribute attribute, Percentage value)
```

Parameters

**attribute** [Attribute](#)

The attribute to modify.

**value** [Percentage](#)

The percentage value to add.

## Get(Attribute)

Gets the final calculated value of a specific attribute. The final value includes base value, flat and percentage modifiers, and spent attribute points. The result is clamped to the attribute's min/max values.

```
public long Get(Attribute attribute)
```

Parameters

**attribute** [Attribute](#)

The attribute to retrieve.

Returns

long

The final value of the attribute.

## GetBase(Attribute)

Gets the base value of an attribute at the entity's current level. Base attributes don't consider flat or percentage modifiers, nor spent attribute points.

```
public long GetBase(Attribute attribute)
```

Parameters

**attribute** [Attribute](#)

The attribute to retrieve the base value for.

Returns

long

The base value of the attribute, clamped to its min/max values if necessary.

## SpendOn(Attribute, int)

Spends a certain amount of attribute points on a specific attribute.

```
public void SpendOn(Attribute attribute, int amount)
```

Parameters

**attribute** [Attribute](#)

The attribute to increase.

**amount** int

The number of points to spend.

# Interface IAttributeContainer

Namespace: [ElectricDrill.SimpleRpgCore.Attributes](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IAttributeContainer
```

## Methods

### Contains(Attribute)

```
bool Contains(Attribute attribute)
```

Parameters

**attribute** [Attribute](#)

Returns

bool

# Namespace ElectricDrill.SimpleRpgCore.Events

## Classes

[AttributeChangedGameEvent](#)

The attribute that changed, the attribute's previous value, and the attribute's new value

[AttributeChangedGameEventListener](#)

The attribute that changed, the attribute's previous value, and the attribute's new value

[EntityCoreGameEvent](#)

[EntityCoreGameEventListener](#)

[GameEvent](#)

[GameEventGenerator](#)

[GameEventGenerator.EventParameter](#)

[GameEventGenerator.GameEventDefinition](#)

[GameEventGeneric1<T>](#)

[GameEventGeneric2<T, U>](#)

[GameEventGeneric3<T, U, W>](#)

[GameEventGeneric4<T, U, W, K>](#)

[GameEventListener](#)

[GameEventListenerGeneric1<T>](#)

[GameEventListenerGeneric2<T, U>](#)

[GameEventListenerGeneric3<T, U, W>](#)

[GameEventListenerGeneric4<T, U, W, K>](#)

[IntGameEvent](#)

[IntGameEventListener](#)

[StatChangedGameEvent](#)

The stat that changed, the stat's previous value, and the stat's new value

## [StatChangedGameEventListener](#)

The stat that changed, the stat's previous value, and the stat's new value

## Interfaces

[IRaisable<T>](#)

[IRaisable<T, U>](#)

[IRaisable<T, U, V>](#)

[IRaisable<T, U, V, W>](#)

## Enums

[GameEventGenerator.EventParameter.NativeType](#)

[GameEventGenerator.EventParameter.ParameterType](#)

# Class AttributeChangedGameEvent

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

The attribute that changed, the attribute's previous value, and the attribute's new value

```
[CreateAssetMenu(fileName = "AttributeChanged Game Event", menuName = "Simple RPG Core/Events/Generated/AttributeChanged")]
public class AttributeChangedGameEvent : GameEventGeneric1<AttributeChangeInfo>, IRaisable<AttributeChangeInfo>
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [GameEventGeneric1<AttributeChangeInfo>](#) ← AttributeChangedGameEvent

## Implements

[IRaisable<AttributeChangeInfo>](#)

## Inherited Members

[GameEventGeneric1<AttributeChangeInfo>.OnEventRaised](#) ,  
[GameEventGeneric1<AttributeChangeInfo>.Raise\(AttributeChangeInfo\)](#) ,  
[GameEventGeneric1<AttributeChangeInfo>.RegisterListener\(GameEventListenerGeneric1<AttributeChangeInfo>\)](#) ,  
[GameEventGeneric1<AttributeChangeInfo>.UnregisterListener\(GameEventListenerGeneric1<AttributeChangeInfo>\)](#)

# Class AttributeChangedGameEventListener

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

The attribute that changed, the attribute's previous value, and the attribute's new value

```
public class AttributeChangedGameEventListener :  
GameEventListenerGeneric1<AttributeChangeInfo>
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ←  
[GameEventListenerGeneric1<AttributeChangeInfo>](#) ← AttributeChangedGameEventListener

## Inherited Members

[GameEventListenerGeneric1<AttributeChangeInfo>.event](#) ,  
[GameEventListenerGeneric1<AttributeChangeInfo>.response](#) ,  
[GameEventListenerGeneric1<AttributeChangeInfo>.OnEventRaised\(AttributeChangeInfo\)](#)

# Class EntityCoreGameEvent

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[CreateAssetMenu(fileName = "EntityCore Game Event", menuName = "Simple RPG
Core/Events/Generated/EntityCore")]
public class EntityCoreGameEvent : GameEventGeneric1<EntityCore>, IRaisable<EntityCore>
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [GameEventGeneric1<EntityCore>](#) ← EntityCoreGameEvent

## Implements

[IRaisable<EntityCore>](#)

## Inherited Members

[GameEventGeneric1<EntityCore>.OnEventRaised](#) , [GameEventGeneric1<EntityCore>.Raise\(EntityCore\)](#) ,
[GameEventGeneric1<EntityCore>.RegisterListener\(GameEventListenerGeneric1<EntityCore>\)](#) ,
[GameEventGeneric1<EntityCore>.UnregisterListener\(GameEventListenerGeneric1<EntityCore>\)](#).

# Class EntityCoreGameEventListener

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class EntityCoreGameEventListener : GameEventListenerGeneric1<EntityCore>
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ←  
[GameEventListenerGeneric1<EntityCore>](#) ← EntityCoreGameEventListener

## Inherited Members

[GameEventListenerGeneric1<EntityCore>.event](#) , [GameEventListenerGeneric1<EntityCore>.response](#) ,  
[GameEventListenerGeneric1<EntityCore>.OnEventRaised\(EntityCore\)](#).

# Class GameEvent

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[CreateAssetMenu(fileName = "New Game Event", menuName = "Simple RPG
Core/Events/Game Event")]
public class GameEvent : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← GameEvent

## Methods

### Raise()

```
public void Raise()
```

### RegisterListener(GameEventListener)

```
public void RegisterListener(GameEventListener listener)
```

#### Parameters

listener [GameEventListener](#)

### UnregisterListener(GameEventListener)

```
public void UnregisterListener(GameEventListener listener)
```

#### Parameters

listener [GameEventListener](#)

# Class GameEventGenerator

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[CreateAssetMenu(fileName = "GameEventGenerator", menuName = "Simple RPG  
Core/Events/GameEventGenerator")]  
public sealed class GameEventGenerator : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← GameEventGenerator

## Fields

### baseSaveLocation

```
public string baseSaveLocation
```

#### Field Value

string

### eventsToGenerate

```
public List<GameEventGenerator.GameEventDefinition> eventsToGenerate
```

#### Field Value

List<[GameEventGenerator.GameEventDefinition](#)>

### menubasePath

```
public string menubasePath
```

Field Value

string

## rootNamespace

```
public string rootNamespace
```

Field Value

string

## Methods

### GenerateGameEvents()

```
public void GenerateGameEvents()
```

### RemoveGeneratedEventFiles(string, int)

```
public void RemoveGeneratedEventFiles(string eventName, int parameterCount)
```

Parameters

eventName string

parameterCount int

# Class GameEventGenerator.EventParameter

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[Serializable]
public class GameEventGenerator.EventParameter
```

## Inheritance

object ← GameEventGenerator.EventParameter

## Fields

### monoScript

```
public MonoScript monoScript
```

#### Field Value

MonoScript

### nativeType

```
public GameEventGenerator.EventParameter.NativeType nativeType
```

#### Field Value

[GameEventGenerator.EventParameter.NativeType](#)

### parameterType

```
public GameEventGenerator.EventParameter.ParameterType parameterType
```

## Field Value

[GameEventGenerator.EventParameter.ParameterType](#)

# Enum GameEventGenerator.EventParameter.NativeType

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public enum GameEventGenerator.EventParameter.NativeType
```

## Fields

bool = 3

float = 2

int = 0

long = 1

# Enum GameEventGenerator.EventParameter.ParameterType

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public enum GameEventGenerator.EventParameter.ParameterType
```

## Fields

MonoScript = 1

Native = 0

# Class

## GameEventGenerator.GameEventDefinition

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[Serializable]
public class GameEventGenerator.GameEventDefinition
```

### Inheritance

object ← GameEventGenerator.GameEventDefinition

## Fields

### documentation

```
[HideInInspector]
public string documentation
```

#### Field Value

string

### eventName

```
public string eventName
```

#### Field Value

string

### isGenerated

```
[HideInInspector]  
public bool isGenerated
```

Field Value

bool

parameters

```
public List<GameEventGenerator.EventParameter> parameters
```

Field Value

List<[GameEventGenerator.EventParameter](#)>

# Class GameEventGeneric1<T>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public abstract class GameEventGeneric1<T> : ScriptableObject, IRaisable<T>
```

## Type Parameters

T

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← GameEventGeneric1<T>

## Implements

[IRaisable](#)<T>

## Derived

[AttributeChangedGameEvent](#), [EntityCoreGameEvent](#), [IntGameEvent](#), [StatChangedGameEvent](#)

## Methods

### Raise(T)

```
public void Raise(T context)
```

#### Parameters

context T

### RegisterListener(GameEventListenerGeneric1<T>)

```
public void RegisterListener(GameEventListenerGeneric1<T> listener)
```

#### Parameters

listener [GameEventListenerGeneric1](#)<T>

## UnregisterListener(GameEventListenerGeneric1<T>)

```
public void UnregisterListener(GameEventListenerGeneric1<T> listener)
```

Parameters

listener [GameEventListenerGeneric1](#)<T>

## Events

### OnEventRaised

```
public event Action<T> OnEventRaised
```

Event Type

Action<T>

# Class GameEventGeneric2<T, U>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public abstract class GameEventGeneric2<T, U> : ScriptableObject, IRaisable<T, U>
```

## Type Parameters

T

U

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← GameEventGeneric2<T, U>

## Implements

[IRaisable](#)<T, U>

# Methods

## Raise(T, U)

```
public void Raise(T context1, U context2)
```

## Parameters

context1 T

context2 U

## RegisterListener(GameEventListenerGeneric2<T, U>)

```
public void RegisterListener(GameEventListenerGeneric2<T, U> listener)
```

## Parameters

listener [GameEventListenerGeneric2](#)<T, U>

UnregisterListener(GameEventListenerGeneric2<T, U>)

```
public void UnregisterListener(GameEventListenerGeneric2<T, U> listener)
```

Parameters

listener [GameEventListenerGeneric2](#)<T, U>

## Events

OnEventRaised

```
public event Action<T, U> OnEventRaised
```

Event Type

Action<T, U>

# Class GameEventGeneric3<T, U, W>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public abstract class GameEventGeneric3<T, U, W> : ScriptableObject, IRaisable<T, U, W>
```

## Type Parameters

T

U

W

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← GameEventGeneric3<T, U, W>

## Implements

[IRaisable](#)<T, U, W>

# Methods

## Raise(T, U, W)

```
public void Raise(T contextT, U contextU, W contextW)
```

## Parameters

contextT T

contextU U

contextW W

## RegisterListener(GameEventListenerGeneric3<T, U, W>)

```
public void RegisterListener(GameEventListenerGeneric3<T, U, W> listener)
```

Parameters

listener [GameEventListenerGeneric3](#)<T, U, W>

## UnregisterListener(GameEventListenerGeneric3<T, U, W>)

```
public void UnregisterListener(GameEventListenerGeneric3<T, U, W> listener)
```

Parameters

listener [GameEventListenerGeneric3](#)<T, U, W>

## Events

### OnEventRaised

```
public event Action<T, U, W> OnEventRaised
```

Event Type

Action<T, U, W>

# Class GameEventGeneric4<T, U, W, K>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public abstract class GameEventGeneric4<T, U, W, K> : ScriptableObject, IRaisable<T, U, W, K>
```

## Type Parameters

T

U

W

K

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← GameEventGeneric4<T, U, W, K>

## Implements

[IRaisable](#)<T, U, W, K>

# Methods

## Raise(T, U, W, K)

```
public void Raise(T contextT, U contextU, W contextW, K contextK)
```

## Parameters

contextT T

contextU U

contextW W

contextK K

## RegisterListener(GameEventListenerGeneric4<T, U, W, K>)

```
public void RegisterListener(GameEventListenerGeneric4<T, U, W, K> listener)
```

### Parameters

listener [GameEventListenerGeneric4<T, U, W, K>](#)

## UnregisterListener(GameEventListenerGeneric4<T, U, W, K>)

```
public void UnregisterListener(GameEventListenerGeneric4<T, U, W, K> listener)
```

### Parameters

listener [GameEventListenerGeneric4<T, U, W, K>](#)

## Events

### OnEventRaised

```
public event Action<T, U, W, K> OnEventRaised
```

### Event Type

Action<T, U, W, K>

# Class GameEventListener

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class GameEventListener : MonoBehaviour
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← GameEventListener

## Methods

### OnEventRaised()

```
public void OnEventRaised()
```

# Class GameEventListenerGeneric1<T>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class GameEventListenerGeneric1<T> : MonoBehaviour
```

## Type Parameters

T

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← GameEventListenerGeneric1<T>

## Derived

[AttributeChangedGameEventListener](#), [EntityCoreGameEventListener](#), [IntGameEventListener](#),  
[StatChangedGameEventListener](#)

## Fields

### \_event

```
[SerializeField]  
protected GameEventGeneric1<T> _event
```

### Field Value

[GameEventGeneric1<T>](#)

### \_response

```
[SerializeField]  
protected UnityEvent<T> _response
```

### Field Value

## Methods

### OnEventRaised(T)

```
public void OnEventRaised(T context)
```

#### Parameters

**context** T

# Class GameEventListenerGeneric2<T, U>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class GameEventListenerGeneric2<T, U> : MonoBehaviour
```

## Type Parameters

T

U

## Inheritance

```
object < Object < Component < Behaviour < MonoBehaviour <
GameEventListenerGeneric2<T, U>
```

## Fields

### \_event

```
[SerializeField]
protected GameEventGeneric2<T, U> _event
```

### Field Value

[GameEventGeneric2<T, U>](#)

### \_response

```
[SerializeField]
protected UnityEvent<T, U> _response
```

### Field Value

[UnityEvent<T, U>](#)

# Methods

## OnEventRaised(T, U)

```
public void OnEventRaised(T contextT, U contextU)
```

### Parameters

contextT T

contextU U

# Class GameEventListenerGeneric3<T, U, W>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class GameEventListenerGeneric3<T, U, W> : MonoBehaviour
```

## Type Parameters

T

U

W

## Inheritance

```
object < Object < Component < Behaviour < MonoBehaviour <
GameEventListenerGeneric3<T, U, W>
```

## Fields

### \_event

```
[SerializeField]
protected GameEventGeneric3<T, U, W> _event
```

### Field Value

[GameEventGeneric3<T, U, W>](#)

### \_response

```
[SerializeField]
protected UnityEvent<T, U, W> _response
```

### Field Value

## Methods

### OnEventRaised(T, U, W)

```
public void OnEventRaised(T contextT, U contextU, W contextW)
```

#### Parameters

contextT T

contextU U

contextW W

# Class GameEventListenerGeneric4<T, U, W, K>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class GameEventListenerGeneric4<T, U, W, K> : MonoBehaviour
```

## Type Parameters

T

U

W

K

## Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric4<T, U, W, K>
```

## Fields

### \_event

```
[SerializeField]  
protected GameEventGeneric4<T, U, W, K> _event
```

### Field Value

[GameEventGeneric4<T, U, W, K>](#)

### \_response

```
[SerializeField]  
protected UnityEvent<T, U, W, K> _response
```

Field Value

UnityEvent<T, U, W, K>

## Methods

OnEventRaised(T, U, W, K)

```
public void OnEventRaised(T contextT, U contextU, W contextW, K contextK)
```

Parameters

contextT T

contextU U

contextW W

contextK K

# Interface IRaisable<T>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IRaisable<T>
```

Type Parameters

T

## Methods

Raise(T)

```
void Raise(T context)
```

Parameters

context T

# Interface IRaisable<T, U>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IRaisable<T, U>
```

Type Parameters

T

U

## Methods

Raise(T, U)

```
void Raise(T context1, U context2)
```

Parameters

context1 T

context2 U

# Interface IRaisable<T, U, V>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IRaisable<T, U, V>
```

## Type Parameters

T

U

V

## Methods

### Raise(T, U, V)

```
void Raise(T context1, U context2, V context3)
```

#### Parameters

context1 T

context2 U

context3 V

# Interface IRaisable<T, U, V, W>

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IRaisable<T, U, V, W>
```

## Type Parameters

T

U

V

W

## Methods

### Raise(T, U, V, W)

```
void Raise(T context1, U context2, V context3, W context4)
```

## Parameters

context1 T

context2 U

context3 V

context4 W

# Class IntGameEvent

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[CreateAssetMenu(fileName = "Int Game Event", menuName = "Simple
RPG Core/Events/Generated/Int")]
public class IntGameEvent : GameEventGeneric1<int>, IRaisable<int>
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [GameEventGeneric1<int>](#) ← IntGameEvent

## Implements

[IRaisable<int>](#)

## Inherited Members

[GameEventGeneric1<int>.OnEventRaised](#) , [GameEventGeneric1<int>.Raise\(int\)](#) ,  
[GameEventGeneric1<int>.RegisterListener\(GameEventListenerGeneric1<int>\)](#) ,  
[GameEventGeneric1<int>.UnregisterListener\(GameEventListenerGeneric1<int>\)](#).

# Class IntGameEventListener

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class IntGameEventListener : GameEventListenerGeneric1<int>
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ←  
[GameEventListenerGeneric1](#)<int> ← IntGameEventListener

## Inherited Members

[GameEventListenerGeneric1<int>.event](#) , [GameEventListenerGeneric1<int>.response](#) ,  
[GameEventListenerGeneric1<int>.OnEventRaised\(int\)](#)

# Class StatChangedGameEvent

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

The stat that changed, the stat's previous value, and the stat's new value

```
[CreateAssetMenu(fileName = "StatChanged Game Event", menuName = "Simple RPG Core/Events/Generated/StatChanged")]
public class StatChangedGameEvent : GameEventGeneric1<StatChangeInfo>,
IRaisable<StatChangeInfo>
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [GameEventGeneric1<StatChangeInfo>](#) ← StatChangedGameEvent

## Implements

[IRaisable<StatChangeInfo>](#)

## Inherited Members

[GameEventGeneric1<StatChangeInfo>.OnEventRaised](#) ,  
[GameEventGeneric1<StatChangeInfo>.Raise\(StatChangeInfo\)](#) ,  
[GameEventGeneric1<StatChangeInfo>.RegisterListener\(GameEventListenerGeneric1<StatChangeInfo>\)](#) ,  
[GameEventGeneric1<StatChangeInfo>.UnregisterListener\(GameEventListenerGeneric1<StatChangeInfo>\)](#) ,

# Class StatChangedGameEventListener

Namespace: [ElectricDrill.SimpleRpgCore.Events](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

The stat that changed, the stat's previous value, and the stat's new value

```
public class StatChangedGameEventListener : GameEventListenerGeneric1<StatChangeInfo>
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ←  
[GameEventListenerGeneric1<StatChangeInfo>](#) ← StatChangedGameEventListener

## Inherited Members

[GameEventListenerGeneric1<StatChangeInfo>.event](#) ,  
[GameEventListenerGeneric1<StatChangeInfo>.response](#) ,  
[GameEventListenerGeneric1<StatChangeInfo>.OnEventRaised\(StatChangeInfo\)](#)

# Namespace ElectricDrill.SimpleRpgCore.Scaling

## Classes

### [AttributeScalingComponentMenuItems](#)

Provides a menu item in the Unity Editor to create an AttributesScalingComponent asset.

### [AttributesScalingComponent](#)

A scaling component that calculates a value based on an entity's attributes.

### [ScalingComponent](#)

Abstract base class for all scaling components. A scaling component calculates a value based on an entity's properties.

### [ScalingFormula](#)

Represents a formula to calculate a value based on values returned by several scaling components. The calculation of each scaling component can either be based on the "self" entity (the entity that owns the formula) or the "target" entity (the entity that will be targeted by the action). For example, a scaling formula for an attack that deals the more damage the higher the enemy's armor is, can use the "self" entity to calculate the damage based on the attacker's stats and the "target" entity to calculate additional damage based on the target's armor. The final damage will be the sum of the value returned by the two scaling components.

### [ScalingFormulaMenuItems](#)

Provides a menu item in the Unity Editor to create a ScalingFormula asset.

### [SoSetScalingComponentBase<SetType, KeyType>](#)

### [StatScalingComponentMenuItems](#)

Provides a menu item in the Unity Editor to create a StatsScalingComponent asset.

### [StatsScalingComponent](#)

A scaling component that calculates a value based on an entity's stats.

# Class AttributeScalingComponentMenuItems

Namespace: [ElectricDrill.SimpleRpgCore.Scaling](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Provides a menu item in the Unity Editor to create an AttributesScalingComponent asset.

```
public static class AttributeScalingComponentMenuItems
```

## Inheritance

object ← AttributeScalingComponentMenuItems

## Methods

### CreateAttributeScalingComponent()

Creates a new AttributesScalingComponent asset.

```
[MenuItem("Assets/Create/Simple RPG Core/Scaling/Attribute Scaling Component", false, 1)]  
public static void CreateAttributeScalingComponent()
```

# Class AttributesScalingComponent

Namespace: [ElectricDrill.SimpleRpgCore.Scaling](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

A scaling component that calculates a value based on an entity's attributes.

```
public class AttributesScalingComponent : SoSetScalingComponentBase<AttributeSet, Attribute>
```

## Inheritance

```
object ← Object ← ScriptableObject ← ScalingComponent ←  
SoSetScalingComponentBase<AttributeSet, Attribute> ← AttributesScalingComponent
```

## Inherited Members

```
SoSetScalingComponentBase<AttributeSet, Attribute>.set ,  
SoSetScalingComponentBase<AttributeSet, Attribute>.CalculateValue(EntityCore) ,  
SoSetScalingComponentBase<AttributeSet, Attribute>.GetEntitySet(EntityCore) ,  
SoSetScalingComponentBase<AttributeSet, Attribute>.GetEntityValue(EntityCore, Attribute) ,  
SoSetScalingComponentBase<AttributeSet, Attribute>.OnValidate() ,  
SoSetScalingComponentBase<AttributeSet, Attribute>.GetSetItems() ,  
ScalingComponent.CalculateValue(EntityCore)
```

## Methods

### GetEntitySet(EntityCore)

Gets the AttributeSet from the specified entity.

```
protected override AttributeSet GetEntitySet(EntityCore entity)
```

#### Parameters

entity [EntityCore](#)

The entity.

#### Returns

## AttributeSet

The entity's AttributeSet.

## GetEntityValue(EntityCore, Attribute)

Gets the value of a specific attribute from the entity.

```
protected override long GetEntityValue(EntityCore entity, Attribute key)
```

Parameters

**entity** [EntityCore](#)

The entity.

**key** [Attribute](#)

The attribute to get the value of.

Returns

long

The value of the attribute.

## GetSetItems()

Gets the collection of attributes from the associated AttributeSet.

```
protected override IEnumerable<Attribute> GetSetItems()
```

Returns

IEnumerable<[Attribute](#)>

An enumerable of attributes.

# Class ScalingComponent

Namespace: [ElectricDrill.SimpleRpgCore.Scaling](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Abstract base class for all scaling components. A scaling component calculates a value based on an entity's properties.

```
public abstract class ScalingComponent : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← ScalingComponent

## Derived

[SoSetScalingComponentBase<SetType, KeyType>](#)

## Methods

### CalculateValue(EntityCore)

Calculates the value of this component based on the provided entity.

```
public abstract long CalculateValue(EntityCore entity)
```

#### Parameters

entity [EntityCore](#)

The entity whose properties are used for calculation.

#### Returns

long

The calculated value from this component.

# Class ScalingFormula

Namespace: [ElectricDrill.SimpleRpgCore.Scaling](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Represents a formula to calculate a value based on values returned by several scaling components. The calculation of each scaling component can either be based on the "self" entity (the entity that owns the formula) or the "target" entity (the entity that will be targeted by the action). For example, a scaling formula for an attack that deals the more damage the higher the enemy's armor is, can use the "self" entity to calculate the damage based on the attacker's stats and the "target" entity to calculate additional damage based on the target's armor. The final damage will be the sum of the value returned by the two scaling components.

```
public class ScalingFormula : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← ScalingFormula

## Properties

### TmpSelfScalingComponents

Gets the list of temporary scaling components for the "self" entity. These are not serialized and are used for runtime modifications.

```
public List<ScalingComponent> TmpSelfScalingComponents { get; }
```

### Property Value

List<[ScalingComponent](#)>

### TmpTargetScalingComponents

Gets the list of temporary scaling components for the "target" entity. These are not serialized and are used for runtime modifications.

```
public List<ScalingComponent> TmpTargetScalingComponents { get; }
```

## Property Value

List<[ScalingComponent](#)>

## Methods

### CalculateValue(EntityCore)

Calculates the final value using only the "self" entity. This method assumes there are no target scaling components.

```
public long CalculateValue(EntityCore self)
```

#### Parameters

**self** [EntityCore](#)

The entity providing values for self-scaling components.

#### Returns

long

The calculated value.

### CalculateValue(EntityCore, EntityCore)

Calculates the final value using both "self" and "target" entities.

```
public long CalculateValue(EntityCore self, EntityCore target)
```

#### Parameters

**self** [EntityCore](#)

The entity providing values for self-scaling components.

## **target** [EntityCore](#)

The entity providing values for target-scaling components.

Returns

long

The calculated value.

## **CalculateValue(EntityCore, EntityCore, int)**

Calculates the final value using both "self" and "target" entities, and a specific level for the base value. This method assumes the formula uses a scaling base value.

```
public long CalculateValue(EntityCore self, EntityCore target, int level)
```

Parameters

### **self** [EntityCore](#)

The entity providing values for self-scaling components.

### **target** [EntityCore](#)

The entity providing values for target-scaling components.

**level** int

The level to use for calculating the base value.

Returns

long

The calculated value.

## **CalculateValue(EntityCore, int)**

Calculates the final value using the "self" entity and a specific level for the base scaling value. This method assumes there are no target scaling components and the formula uses a scaling base value.

```
public long CalculateValue(EntityCore self, int level)
```

## Parameters

**self** [EntityCore](#)

The entity providing values for self-scaling components.

**level** int

The level to use for calculating the base value.

## Returns

long

The calculated value.

## ResetTmpScalings()

Clears all temporary scaling components for both "self" and "target".

```
public void ResetTmpScalings()
```

# Class ScalingFormulaMenuItems

Namespace: [ElectricDrill.SimpleRpgCore.Scaling](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Provides a menu item in the Unity Editor to create a ScalingFormula asset.

```
public static class ScalingFormulaMenuItems
```

## Inheritance

object ← ScalingFormulaMenuItems

## Methods

### CreateScalingFormula()

Creates a new ScalingFormula asset.

```
[MenuItem("Assets/Create/Simple RPG Core/Scaling/Scaling Formula &#S", false, 0)]  
public static void CreateScalingFormula()
```

# Class SoSetScalingComponentBase<SetType, KeyType>

Namespace: [ElectricDrill.SimpleRpgCore.Scaling](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public abstract class SoSetScalingComponentBase<SetType, KeyType> : ScalingComponent where
    SetType : ScriptableObject
```

## Type Parameters

**SetType**

**KeyType**

## Inheritance

```
object ← Object ← ScriptableObject ← ScalingComponent ←
SoSetScalingComponentBase<SetType, KeyType>
```

## Derived

[AttributesScalingComponent](#), [StatsScalingComponent](#)

## Fields

**\_set**

```
[SerializeField]
protected SetType _set
```

## Field Value

SetType

## Methods

## CalculateValue(EntityCore)

Calculates the value of this component based on the provided entity.

```
public override long CalculateValue(EntityCore entity)
```

Parameters

**entity** [EntityCore](#)

The entity whose properties are used for calculation.

Returns

long

The calculated value from this component.

## GetEntitySet(EntityCore)

```
protected abstract SetType GetEntitySet(EntityCore entity)
```

Parameters

**entity** [EntityCore](#)

Returns

SetType

## GetEntityValue(EntityCore, KeyType)

```
protected abstract long GetEntityValue(EntityCore entity, KeyType key)
```

Parameters

**entity** [EntityCore](#)

`key` KeyType

Returns

long

## GetSetItems()

```
protected abstract IEnumerable<KeyType> GetSetItems()
```

Returns

IEnumerable<KeyType>

## OnValidate()

```
protected virtual void OnValidate()
```

# Class StatScalingComponentMenuItems

Namespace: [ElectricDrill.SimpleRpgCore.Scaling](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Provides a menu item in the Unity Editor to create a StatsScalingComponent asset.

```
public static class StatScalingComponentMenuItems
```

## Inheritance

object ← StatScalingComponentMenuItems

## Methods

### CreateStatScalingComponent()

Creates a new StatsScalingComponent asset.

```
[MenuItem("Assets/Create/Simple RPG Core/Scaling/Stat Scaling Component", false, 2)]  
public static void CreateStatScalingComponent()
```

# Class StatsScalingComponent

Namespace: [ElectricDrill.SimpleRpgCore.Scaling](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

A scaling component that calculates a value based on an entity's stats.

```
public class StatsScalingComponent : SoSetScalingComponentBase<StatSet, Stat>
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [ScalingComponent](#) ←  
[SoSetScalingComponentBase](#)<[StatSet](#), [Stat](#)> ← StatsScalingComponent

## Inherited Members

[SoSetScalingComponentBase](#)<[StatSet](#), [Stat](#)>.set ,  
[SoSetScalingComponentBase](#)<[StatSet](#), [Stat](#)>.CalculateValue([EntityCore](#)) ,  
[SoSetScalingComponentBase](#)<[StatSet](#), [Stat](#)>.GetEntitySet([EntityCore](#)) ,  
[SoSetScalingComponentBase](#)<[StatSet](#), [Stat](#)>.GetEntityValue([EntityCore](#), [Stat](#)) ,  
[SoSetScalingComponentBase](#)<[StatSet](#), [Stat](#)>.OnValidate() ,  
[SoSetScalingComponentBase](#)<[StatSet](#), [Stat](#)>.GetSetItems() ,  
[ScalingComponent](#).CalculateValue([EntityCore](#))

## Methods

### GetEntitySet([EntityCore](#))

Gets the StatSet from the specified entity.

```
protected override StatSet GetEntitySet(EntityCore entity)
```

#### Parameters

[entity](#) [EntityCore](#)

The entity.

#### Returns

## StatSet

The entity's StatSet.

## GetEntityValue(EntityCore, Stat)

Gets the value of a specific stat from the entity.

```
protected override long GetEntityValue(EntityCore entity, Stat key)
```

Parameters

**entity** [EntityCore](#)

The entity.

**key** [Stat](#)

The stat to get the value of.

Returns

long

The value of the stat.

## GetSetItems()

Gets the collection of stats from the associated StatSet.

```
protected override IEnumerable<Stat> GetSetItems()
```

Returns

IEnumerable<[Stat](#)>

An enumerable of stats.

# Namespace ElectricDrill.SimpleRpgCore.Stats

## Classes

### [EntityStats](#)

Component that manages the statistics of an entity in the game. It handles base stats, flat stat modifiers, stat to stat modifiers, and percentage stat modifiers.

Base stats can either be fixed or come from the entity's class (if one is available on the Game Object).

When stats change because of a modifier of any kind, the assigned [StatChangedGameEvent](#) is raised.

### [Stat](#)

### [StatMenuItems](#)

### [StatSet](#)

### [StatSetInstance](#)

### [StatSetMenuItems](#)

### [StatToStatModifier](#)

### [StatToStatModifierMenuItems](#)

## Interfaces

### [IStatContainer](#)

# Class EntityStats

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

Component that manages the statistics of an entity in the game. It handles base stats, flat stat modifiers, stat to stat modifiers, and percentage stat modifiers.

Base stats can either be fixed or come from the entity's class (if one is available on the Game Object). When stats change because of a modifier of any kind, the assigned [StatChangedGameEvent](#) is raised.

```
[RequireComponent(typeof(EntityCore))]  
public class EntityStats : MonoBehaviour, IStatSetSource
```

## Inheritance

object ← [Object](#) ← [Component](#) ← [Behaviour](#) ← [MonoBehaviour](#) ← EntityStats

## Implements

[IStatSetSource](#)

## Fields

### \_entityClass

```
protected IClassSource _entityClass
```

#### Field Value

[IClassSource](#)

### \_flatModifiers

```
protected StatSetInstance _flatModifiers
```

#### Field Value

[StatSetInstance](#)

## \_percentageModifiers

```
protected StatSetInstance _percentageModifiers
```

Field Value

[StatSetInstance](#)

## Properties

### EntityClass

The class source of the entity. In most cases, this is the [EntityClass](#) component attached to the entity.

```
public IClassSource EntityClass { get; }
```

Property Value

[IClassSource](#)

### EntityCore

The entity core associated with this stats component.

```
public EntityCore EntityCore { get; }
```

Property Value

[EntityCore](#)

### OnStatChanged

Event raised when a stat changes due to a modifier.

```
public StatChangedGameEvent OnStatChanged { get; }
```

Property Value

[StatChangedGameEvent](#)

## StatSet

The stat set used to calculate the entity's stats.

```
public virtual StatSet StatSet { get; }
```

Property Value

[StatSet](#)

If useBaseStatsFromClass is true, it returns the stat set of the entity's class. Otherwise, it returns the fixed base stats stat set.

## StatsCache

```
public Cache<Stat, long> StatsCache { get; }
```

Property Value

[Cache<Stat, long>](#)

## UseClassBaseStats

Indicates whether to use base stats from the entity's class or the fixed base stats.

```
public bool UseClassBaseStats { get; }
```

Property Value

bool

## Methods

### AddFlatModifier(Stat, long)

Adds a flat modifier to a stat.

```
public void AddFlatModifier(Stat stat, long value)
```

#### Parameters

**stat** [Stat](#)

The stat to add the flat modifier to.

**value** long

The value of the flat modifier.

### AddPercentageModifier(Stat, Percentage)

Adds a [Percentage](#) modifier to a stat. Such modifiers consider the base value of the stat, the flat modifiers, and the stat-to-stat modifiers.

```
public void AddPercentageModifier(Stat stat, Percentage value)
```

#### Parameters

**stat** [Stat](#)

The stat to add the percentage modifier to.

**value** [Percentage](#)

The value of the percentage modifier.

### AddStatToStatModifer(Stat, Stat, Percentage)

Adds a stat-to-stat modifier. Such modifiers add a percentage of the source stat to the target stat. Such modifiers consider the base value and the flat modifiers of the source stat.

```
public void AddStatToStatModifier(Stat target, Stat source, Percentage percentage)
```

## Parameters

**target** [Stat](#)

The target stat.

**source** [Stat](#)

The source stat.

**percentage** [Percentage](#)

The [Percentage](#) of the source stat to add to the target stat.

## Get(Stat)

The final value of a stat, considering all the modifiers. Calculation is done in the following order:

1. Base value
2. Flat modifiers
3. Stat to stat modifiers
4. Percentage modifiers

```
public virtual long Get(Stat stat)
```

## Parameters

**stat** [Stat](#)

The stat to get the final value of.

## Returns

**long**

The final value of the stat. The value is clamped to the stat's min and max values.

## GetBase(Stat)

The base value is the value of the stat without any modifiers. If UseClassBaseStats is true, it returns the value from the entity's class. Otherwise, it returns the value from the fixed base stats.

```
public long GetBase(Stat stat)
```

Parameters

**stat** [Stat](#)

The stat to get the base value of.

Returns

long

The base value of the stat. The value is clamped to the stat's min and max values.

## OnLevelUp(int)

Callback method called when the entity levels up.

```
protected virtual void OnLevelUp(int level)
```

Parameters

**level** int

The new level of the entity.

## SetFixed(Stat, long)

Sets the value of a fixed base stat.

```
public void SetFixed(Stat s, long v)
```

Parameters

**s** Stat

The stat to set.

**v** long

The value to set.

# Interface IStatContainer

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public interface IStatContainer
```

## Methods

### Contains(Stat)

```
bool Contains(Stat stat)
```

Parameters

**stat** [Stat](#)

Returns

bool

# Class Stat

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class Stat : BoundedValue
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [BoundedValue](#) ← Stat

## Inherited Members

[BoundedValue.HasValue](#) , [BoundedValue.MaxValue](#) , [BoundedValue.HasValue](#) ,  
[BoundedValue.MinValue](#)

# Properties

## AttributesScaling

```
[CanBeNull]  
public AttributesScalingComponent AttributesScaling { get; }
```

## Property Value

[AttributesScalingComponent](#)

# Methods

## Equals(object)

```
public override bool Equals(object obj)
```

## Parameters

**obj** object

Returns

bool

## GetHashCode()

```
public override int GetHashCode()
```

Returns

int

## Operators

### operator ==(Stat, Stat)

```
public static bool operator ==(Stat a, Stat b)
```

Parameters

a [Stat](#)

b [Stat](#)

Returns

bool

### operator !=(Stat, Stat)

```
public static bool operator !=(Stat a, Stat b)
```

Parameters

a [Stat](#)

b [Stat](#)

Returns

bool

# Class StatMenuItems

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public static class StatMenuItems
```

## Inheritance

object ← StatMenuItems

## Methods

### CreateStat()

```
[MenuItem("Assets/Create/Simple RPG Core/Stat ^&S", false, 2)]  
public static void CreateStat()
```

# Class StatSet

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class StatSet : ScriptableObject, IStatContainer
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← StatSet

## Implements

[IStatContainer](#)

# Properties

## Stats

```
public IReadOnlyList<Stat> Stats { get; }
```

## Property Value

[IReadOnlyList](#)<[Stat](#)>

# Methods

## Contains(Stat)

```
public virtual bool Contains(Stat stat)
```

## Parameters

stat [Stat](#)

## Returns

bool

## Get(Stat)

```
public Stat Get(Stat stat)
```

Parameters

stat [Stat](#)

Returns

[Stat](#)

# Class StatSetInstance

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class StatSetInstance : IStatContainer
```

## Inheritance

object ← StatSetInstance

## Implements

[IStatContainer](#)

## Constructors

### StatSetInstance(StatSet)

```
public StatSetInstance(StatSet statSet)
```

## Parameters

statSet [StatSet](#)

## Properties

### this[Stat]

```
public long this[Stat stat] { get; set; }
```

## Parameters

stat [Stat](#)

### Property Value

long

## Stats

```
public Dictionary<Stat, long> Stats { get; }
```

### Property Value

Dictionary<[Stat](#), long>

## Methods

### AddValue(Stat, long)

Adds [value](#) to [stat](#). If the stat does not exist, it will be created and initialized with [value](#). Use negative values to subtract from the stat.

```
public void AddValue(Stat stat, long value)
```

### Parameters

[stat](#) [Stat](#)

The stat to add the value to.

[value](#) long

The value to add to the stat.

### Clone()

```
public StatSetInstance Clone()
```

### Returns

[StatSetInstance](#)

## Contains(Stat)

```
public bool Contains(Stat stat)
```

Parameters

**stat** [Stat](#)

Returns

bool

## Get(Stat)

```
public long Get(Stat stat)
```

Parameters

**stat** [Stat](#)

The stat to be retrieved.

Returns

long

The value of the **stat**

## GetAsPercentage(Stat)

```
public Percentage GetAsPercentage(Stat stat)
```

Parameters

**stat** [Stat](#)

Returns

## Percentage

### GetEnumerator()

```
public IEnumarator<KeyValuePair<Stat, long>> GetEnumerator()
```

Returns

IEnumarator<KeyValuePair<[Stat](#), long>>

## Operators

### operator +(StatSetInstance, StatSetInstance)

The addition operator for StatSetInstance. Considered the stats present in the StatSetInstance [a](#), their values will be summed with the values of the respective stats in the StatSetInstance [b](#). If a stat is present in the StatSetInstance [a](#) but not in the StatSetInstance [b](#), an exception will be thrown.

```
public static StatSetInstance operator +(StatSetInstance a, StatSetInstance b)
```

Parameters

[a](#) [StatSetInstance](#)

The first StatSetInstance

[b](#) [StatSetInstance](#)

The second StatSetInstance

Returns

[StatSetInstance](#)

A new StatSetInstance with the sum of the stats of [a](#) to the respective values of the stats of [b](#)

# Class StatSetMenuItems

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public static class StatSetMenuItems
```

## Inheritance

object ← StatSetMenuItems

## Methods

### CreateStatSet()

```
[MenuItem("Assets/Create/Simple RPG Core/Stat Set", false, 3)]  
public static void CreateStatSet()
```

# Class StatToStatModifier

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public class StatToStatModifier : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← StatToStatModifier

## Properties

### Percentage

```
public Percentage Percentage { get; }
```

#### Property Value

[Percentage](#)

### SourceStat

```
public Stat SourceStat { get; }
```

#### Property Value

[Stat](#)

### TargetStat

```
public Stat TargetStat { get; }
```

#### Property Value



# Class StatToStatModifierMenuItems

Namespace: [ElectricDrill.SimpleRpgCore.Stats](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public static class StatToStatModifierMenuItems
```

## Inheritance

object ← StatToStatModifierMenuItems

## Methods

### CreateStatToStatModifier()

```
[MenuItem("Assets/Create/Simple RPG Core/Stat to Stat Modifier ^&M", false, 4)]  
public static void CreateStatToStatModifier()
```

# Namespace ElectricDrill.SimpleRpgCore.Utils

## Classes

### [InitializationUtils](#)

#### [IntRef](#)

A reference to an integer value. Can either be a constant value or a reference to an [IntVar](#) ScriptableObject.

#### [IntVar](#)

ScriptableObject that holds an integer value. Can be used to share an integer value between different GameObjects and scenes.

#### [LongRef](#)

A reference to a long value. Can either be a constant value or a reference to a [LongVar](#) ScriptableObject.

#### [LongVar](#)

ScriptableObject that holds a long value. Can be used to share a long value between different GameObjects and scenes.

#### [Percentage](#)

The Percentage class represents a percentage value and provides various operators and conversions. Implicit long to Percentage value conversion is available. To express a 100% value, use 100L. Implicit Percentage to double conversion is available. When doing so, the percentage is automatically divided by 100.

### [SerializableDictionary< TKey, TValue >](#)

### [SerializableHashSet< T >](#)

## Structs

### [SerKeyValuePair< T, U >](#)

# Class InitializationUtils

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
public static class InitializationUtils
```

## Inheritance

object ← InitializationUtils

## Methods

RefreshInspectorReservedValues<TKey, TValue>(ref List<SerKeyValPair<TKey, TValue>>, IEnumerable<TKey>)

```
public static void RefreshInspectorReservedValues<TKey, TValue>(ref List<SerKeyValPair<TKey, TValue>> inspectorReservedValues, IEnumerable<TKey> keys)
```

### Parameters

**inspectorReservedValues** List<[SerKeyValPair](#)<TKey, TValue>>

**keys** IEnumerable<TKey>

### Type Parameters

**TKey**

**TValue**

# Class IntRef

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

A reference to an integer value. Can either be a constant value or a reference to an [IntVar](#) ScriptableObject.

```
[Serializable]  
public class IntRef
```

## Inheritance

object ← IntRef

## Fields

### ConstantValue

The constant integer value.

```
public int ConstantValue
```

#### Field Value

int

### UseConstant

If true, uses the constant value. Otherwise, uses the [IntVar](#) variable.

```
public bool UseConstant
```

#### Field Value

bool

# Variable

The [IntVar](#) variable.

```
public IntVar Variable
```

## Field Value

[IntVar](#)

# Properties

## Value

Gets or sets the value of the reference.

```
public int Value { get; set; }
```

## Property Value

int

# Operators

## implicit operator int(IntRef)

Allows implicit conversion from an IntRef to an int.

```
public static implicit operator int(IntRef reference)
```

## Parameters

**reference** [IntRef](#)

The IntRef to convert.

## Returns

int

## implicit operator IntRef(int)

Allows implicit conversion from an int to an IntRef, creating a constant reference.

```
public static implicit operator IntRef(int value)
```

### Parameters

**value** int

The integer value.

### Returns

[IntRef](#)

# Class IntVar

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

ScriptableObject that holds an integer value. Can be used to share an integer value between different GameObjects and scenes.

```
[CreateAssetMenu]  
public class IntVar : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← IntVar

## Properties

### Value

The integer value.

```
public int Value { get; set; }
```

Property Value

int

## Operators

### implicit operator int(IntVar)

Allows implicit conversion from an IntVar to an int.

```
public static implicit operator int(IntVar var)
```

Parameters

**var** [IntVar](#)

The IntVar to convert.

Returns

int

# Class LongRef

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

A reference to a long value. Can either be a constant value or a reference to a [LongVar](#) ScriptableObject.

```
[Serializable]
public class LongRef
```

## Inheritance

object ← LongRef

## Fields

### ConstantValue

The constant long value.

```
public long ConstantValue
```

### Field Value

long

### UseConstant

If true, uses the constant value. Otherwise, uses the [LongVar](#) variable.

```
public bool UseConstant
```

### Field Value

bool

# Variable

The [LongVar](#) variable.

```
public LongVar Variable
```

## Field Value

[LongVar](#)

# Properties

## Value

Gets or sets the value of the reference.

```
public long Value { get; set; }
```

## Property Value

long

# Operators

## implicit operator long(LongRef)

Allows implicit conversion from a LongRef to a long.

```
public static implicit operator long(LongRef reference)
```

## Parameters

[reference](#) [LongRef](#)

The LongRef to convert.

## Returns

long

# Class LongVar

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

ScriptableObject that holds a long value. Can be used to share a long value between different GameObjects and scenes.

```
[CreateAssetMenu]  
public class LongVar : ScriptableObject
```

## Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← LongVar

## Properties

### Value

The long value.

```
public long Value { get; set; }
```

Property Value

long

## Operators

### implicit operator long(LongVar)

Allows implicit conversion from a LongVar to a long.

```
public static implicit operator long(LongVar var)
```

Parameters

`var LongVar`

The LongVar to convert.

Returns

`long`

# Class Percentage

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

The Percentage class represents a percentage value and provides various operators and conversions. Implicit long to Percentage value conversion is available. To express a 100% value, use 100L. Implicit Percentage to double conversion is available. When doing so, the percentage is automatically divided by 100.

```
[Serializable]
public class Percentage
```

## Inheritance

object ← Percentage

## Constructors

### Percentage(long)

Initializes a new instance of the Percentage class with the specified value. To express a 100% value, use 100L.

```
public Percentage(long value)
```

## Parameters

**value** long

The value of the percentage.

## Methods

### CompareTo(Percentage)

Compares the current Percentage instance with another Percentage instance.

```
public int CompareTo(Percentage other)
```

## Parameters

### other Percentage

The other percentage to compare to.

## Returns

int

An integer indicating the relative order of the percentages.

## ToString()

Returns a string representation of the percentage value.

```
public override string ToString()
```

## Returns

string

A string representing the percentage value.

# Operators

## operator +(Percentage, Percentage)

Overrides the + operator to add two Percentage instances.

```
public static Percentage operator +(Percentage a, Percentage b)
```

## Parameters

### a Percentage

The first percentage.

#### b [Percentage](#)

The second percentage.

Returns

#### [Percentage](#)

A new Percentage instance representing the sum.

## explicit operator long(Percentage)

Explicit conversion from Percentage to long. The conversion does not divide the value by 100.

```
public static explicit operator long(Percentage percentage)
```

Parameters

#### percentage [Percentage](#)

The percentage to convert.

Returns

long

## implicit operator double(Percentage)

Implicit conversion from Percentage to double. The conversion automatically divides the value by 100.

```
public static implicit operator double(Percentage percentage)
```

Parameters

#### percentage [Percentage](#)

The percentage to convert.

Returns

double

## implicit operator Percentage(long)

Implicit conversion from long to Percentage. To express a 100% value, use 100L.

```
public static implicit operator Percentage(long value)
```

Parameters

**value** long

The value to convert.

Returns

[Percentage](#)

## operator -(Percentage, Percentage)

Overrides the - operator to subtract one Percentage from another.

```
public static Percentage operator -(Percentage a, Percentage b)
```

Parameters

**a** [Percentage](#)

The first percentage.

**b** [Percentage](#)

The second percentage.

Returns

[Percentage](#)

A new Percentage instance representing the difference.

## operator -(Percentage)

Overrides the unary - operator to negate a Percentage.

```
public static Percentage operator -(Percentage a)
```

Parameters

a [Percentage](#)

The percentage to negate.

Returns

[Percentage](#)

A new Percentage instance representing the negated value.

# Struct SerKeyValuePair<T, U>

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[Serializable]
public struct SerKeyValuePair<T, U>
```

## Type Parameters

T

U

## Constructors

### SerKeyValuePair(T, U)

```
public SerKeyValuePair(T key, U value)
```

## Parameters

key T

value U

## Fields

### Key

```
public T Key
```

## Field Value

T

## Value

```
public U Value
```

Field Value

U

## Operators

implicit operator KeyValuePair<T, U>(SerKeyValPair<T, U>)

```
public static implicit operator KeyValuePair<T, U>(SerKeyValPair<T, U> serKeyValPair)
```

Parameters

**serKeyValPair** [SerKeyValPair](#)<T, U>

Returns

KeyValuePair<T, U>

implicit operator SerKeyValPair<T, U>(KeyValuePair<T, U>)

```
public static implicit operator SerKeyValPair<T, U>(KeyValuePair<T, U> keyValuePair)
```

Parameters

**keyValuePair** KeyValuePair<T, U>

Returns

[SerKeyValPair](#)<T, U>

# Class SerializableDictionary<TKey, TValue>

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[Serializable]
public class SerializableDictionary<TKey, TValue>
```

## Type Parameters

TKey

TValue

## Inheritance

object ← SerializableDictionary<TKey, TValue>

## Properties

### this[TKey]

```
public TValue this[TKey key] { get; set; }
```

#### Parameters

key TKey

#### Property Value

TValue

## Keys

```
public Dictionary<TKey, TValue>.KeyCollection Keys { get; }
```

## Property Value

Dictionary<TKey, TValue>.KeyCollection

## Values

```
public Dictionary<TKey, TValue>.ValueCollection Values { get; }
```

## Property Value

Dictionary<TKey, TValue>.ValueCollection

## Methods

### Clear()

```
public void Clear()
```

### ContainsKey(TKey)

```
public bool ContainsKey(TKey key)
```

#### Parameters

key TKey

#### Returns

bool

### GetEnumerator()

```
public IEnumerator<KeyValuePair<TKey, TValue>> GetEnumerator()
```

Returns

IEnumerator<KeyValuePair<TKey, TValue>>

## OnAfterDeserialize()

Implement this callback to transform data back into runtime data types after an object is serialized.

```
public void OnAfterDeserialize()
```

## OnBeforeSerialize()

Implement this callback to transform data into serializable data types immediately before an object is serialized.

```
public void OnBeforeSerialize()
```

## TryGetValue(TKey, out TValue)

```
public bool TryGetValue(TKey key, out TValue value)
```

Parameters

**key** TKey

**value** TValue

Returns

bool

## Operators

implicit operator Dictionary<TKey, TValue>

## (SerializableDictionary<TKey, TValue>)

```
public static implicit operator Dictionary<TKey, TValue>(SerializableDictionary<TKey, TValue> serializableDictionary)
```

### Parameters

**serializableDictionary** [SerializableDictionary<TKey, TValue>](#)

### Returns

[Dictionary<TKey, TValue>](#)

## implicit operator SerializableDictionary<TKey, TValue> (Dictionary<TKey, TValue>)

```
public static implicit operator SerializableDictionary<TKey, TValue>(Dictionary<TKey, TValue> dictionary)
```

### Parameters

**dictionary** [Dictionary<TKey, TValue>](#)

### Returns

[SerializableDictionary<TKey, TValue>](#)

# Class SerializableHashSet<T>

Namespace: [ElectricDrill.SimpleRpgCore.Utils](#)

Assembly: ElectricDrill.SimpleRPGCore.Runtime.dll

```
[Serializable]
public class SerializableHashSet<T>
```

## Type Parameters

T

## Inheritance

object ← SerializableHashSet<T>

## Properties

### Count

```
public int Count { get; }
```

### Property Value

int

### IsReadOnly

```
public bool IsReadOnly { get; }
```

### Property Value

bool

## Methods

## Add(T)

```
public void Add(T item)
```

### Parameters

item T

## Clear()

```
public void Clear()
```

## Contains(T)

```
public bool Contains(T item)
```

### Parameters

item T

### Returns

bool

## CopyTo(T[], int)

```
public void CopyTo(T[] array, int arrayIndex)
```

### Parameters

array T[]

arrayIndex int

## GetEnumerator()

```
public IEnumarator<T> GetEnumerator()
```

Returns

IEnumerator<T>

## GetObjectData(SerializationInfo, StreamingContext)

```
public void GetObjectData(SerializationInfo info, StreamingContext context)
```

Parameters

**info** SerializationInfo

**context** StreamingContext

## OnAfterDeserialize()

Implement this callback to transform data back into runtime data types after an object is deserialized.

```
public void OnAfterDeserialize()
```

## OnBeforeSerialize()

Implement this callback to transform data into serializable data types immediately before an object is serialized.

```
public void OnBeforeSerialize()
```

## Remove(T)

```
public bool Remove(T item)
```

Parameters

**item** T

Returns

bool

## RemoveWhere(Predicate<T>)

```
public int RemoveWhere(Predicate<T> match)
```

Parameters

**match** Predicate<T>

Returns

int