Namespace ElectricDrill

Structs

StatChangeInfo

Contains information about a stat change event, including the entity, stat, and old/new values. Used for tracking and responding to stat modifications.

Struct StatChangeInfo

Namespace: ElectricDrill

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Contains information about a stat change event, including the entity, stat, and old/new values. Used for tracking and responding to stat modifications.

```
public struct StatChangeInfo
```

Constructors

StatChangeInfo(EntityStats, Stat, long, long)

Initializes a new instance of the StatChangeInfo structure.

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

Parameters

`entity` EntityStats

The EntityStats component that owns the stat.

`stat` Stat

The stat that was changed.

`oldValue` long

The previous value of the stat.

`newValue` long

The new value of the stat.

Fields

EntityStats

The EntityStats component that owns the changed stat.

```
public EntityStats EntityStats
```

Field Value

EntityStats

NewValue

The value of the stat after the change.

```
public long NewValue
```

Field Value

long

OldValue

The value of the stat before the change.

```
public long OldValue
```

Field Value

long

Stat

The stat that was changed.

```
public Stat Stat
```

Field Value

Stat

Namespace ElectricDrill.SoapRpgFramework

Classes

BoundedValue

Abstract base class for values that can be bounded by minimum and maximum limits. Provides functionality to define optional min/max constraints and clamp values within those bounds.

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

Component that represents the class of an entity in the game. Implements IClassSource to provide access to the entity's class definition.

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 system of an entity in the RPG framework. Handles experience gain, level progression, and experience-based calculations. Implements ILevelable to provide standard leveling functionality.

ExpSource

A MonoBehaviour component that provides a source of experience points. Implements IExpSource and can only be harvested once. Once harvested, subsequent calls to Exp will return 0.

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

Interface for entities that have an attributes system. Provides access to the entity's attributes component.

IClassSource

Interface for objects that provide a source of a class. In RPG games, a class typically defines the role or profession of an entity, such as a warrior, mage, or rogue.

IExpSource

Interface for objects that can provide experience points. Defines the contract for experience sources that can be harvested.

ILevel

Interface for entities that have a level system. Provides access to the entity's level management component.

ILevelable

Interface for entities that have a leveling system based on experience points. Provides the core functionality for level progression and experience management.

IStatSetSource

Interface for objects that provide a source of stat sets. Defines the contract for accessing a StatSet that contains stat definitions.

Class BoundedValue

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Abstract base class for values that can be bounded by minimum and maximum limits. Provides functionality to define optional min/max constraints and clamp values within those bounds.

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

Inheritance

object ← Object ← ScriptableObject ← BoundedValue

Derived

Attribute, Stat

Properties

HasMaxValue

Gets or sets whether this value has a maximum limit.

```
public bool HasMaxValue { get; }
```

Property Value

bool

HasMinValue

Gets or sets whether this value has a minimum limit.

```
public bool HasMinValue { get; }
```

Property Value

bool

MaxValue

Gets or sets the maximum allowed value when HasMaxValue is true.

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

Property Value

long

MinValue

Gets or sets the minimum allowed value when HasMinValue is true.

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

Property Value

int

Class Class

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.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/Soap RPG Framework/Class", false, 5)]
public static void CreateClass()
```

Class EntityClass

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Component that represents the class of an entity in the game. Implements IClassSource to provide access to the entity's class definition.

```
[DisallowMultipleComponent]
public class EntityClass : MonoBehaviour, IClassSource
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← EntityClass

Implements

IClassSource

Properties

Class

Gets or sets the class definition for this entity.

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

Property Value

Class

Operators

implicit operator Class(EntityClass)

Implicit conversion operator that allows an EntityClass to be used directly as a Class.

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

Parameters

`entityClass` EntityClass

The EntityClass to convert.

Returns

Class

The Class associated with the EntityClass.

Class EntityCore

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.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.

```
[DisallowMultipleComponent]
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.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Represents the level and experience system of an entity in the RPG framework. Handles experience gain, level progression, and experience-based calculations. Implements ILevelable to provide standard leveling functionality.

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

Inheritance

object ← EntityLevel

Implements

ILevelable

Properties

CurrentTotalExperience

Gets the current total experience of the entity. This represents all experience gained since the entity was created.

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

Property Value

long

Level

Gets or sets the current level of the entity. Level must be between 1 and the maximum level. Setting the level will trigger the OnLevelUp event.

```
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. The experience will be modified by any experience gain modifiers. If the entity gains enough experience to level up, the OnLevelUp event will be raised. Can trigger multiple level ups if enough experience is added.

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

Parameters

`amount` long

The base amount of experience to add before modifiers are applied.

CurrentLevelTotalExperience()

Gets the total experience required to reach the current level. For level 1, this returns 0. For higher levels, it uses the experience growth formula.

```
public long CurrentLevelTotalExperience()
```

Returns

long

The total experience required for the current level.

NextLevelTotalExperience()

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

```
public long NextLevelTotalExperience()
```

Returns

long

The total experience required for the next level, or current level experience if at max level.

SetTotalCurrentExp(long)

Sets the total current experience and automatically updates the level to match. The level will be calculated based on the experience growth formula.

```
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 the experience doesn't match the level, it will be reset to the base experience for the current level. This method is useful for debugging and ensuring data consistency.

```
public void ValidateExperience()
```

Operators

implicit operator int(EntityLevel)

Implicitly converts an EntityLevel to its integer level value. Allows EntityLevel objects to be used directly as integers in calculations.

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

Parameters

`entityLevel` EntityLevel

The EntityLevel instance to convert.

Returns

int

The current level as an integer.

Class ExpSource

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A MonoBehaviour component that provides a source of experience points. Implements IExpSource and can only be harvested once. Once harvested, subsequent calls to Exp will return 0.

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

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← ExpSource

Implements

IExpSource

Properties

Exp

Gets the experience points from this source. The first call will return the configured experience and mark the source as harvested. Subsequent calls will return 0.

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

Property Value

long

Harvested

Gets or sets whether this experience source has been harvested. Once set to true, the Exp property will return 0.

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

Property Value

bool

Class GrowthFormula

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.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/Soap RPG Framework/Growth Formula", false, 6)]
public static void CreateGrowthFormula()
```

Interface IAttributes

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for entities that have an attributes system. Provides access to the entity's attributes component.

```
public interface IAttributes
```

Properties

Attributes

Gets the entity's attributes component.

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

Property Value

EntityAttributes

Interface IClassSource

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that provide a source of a class. In RPG games, a class typically defines the role or profession of an entity, such as a warrior, mage, or rogue.

```
public interface IClassSource
```

Properties

Class

Gets the class.

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

Property Value

Class

Interface IExpSource

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that can provide experience points. Defines the contract for experience sources that can be harvested.

```
public interface IExpSource
```

Properties

Exp

Gets the experience points available from this source. The behavior may vary based on implementation and harvest state.

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

Property Value

long

Harvested

Gets or sets whether this experience source has been harvested. Used to track if the experience has already been collected.

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

Property Value

bool

Interface ILevel

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for entities that have a level system. Provides access to the entity's level management component.

```
public interface ILevel
```

Properties

Level

Gets the entity's level management component.

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

Property Value

EntityLevel

Interface ILevelable

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for entities that have a leveling system based on experience points. Provides the core functionality for level progression and experience management.

```
public interface ILevelable
```

Properties

CurrentTotalExperience

Gets the current total experience of the entity. This represents all experience gained since creation.

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

Property Value

long

Level

Gets or sets the current level of the entity. Setting the level may trigger level-up events and related effects.

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

Property Value

int

Methods

AddExp(long)

Adds experience points to the entity. May cause the entity to level up if enough experience is gained.

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

Parameters

`amount` long

The amount of experience to add.

CurrentLevelTotalExperience()

Gets the total experience required to reach the current level.

```
long CurrentLevelTotalExperience()
```

Returns

long

The experience threshold for the current level.

NextLevelTotalExperience()

Gets the total experience required to reach the next level.

```
long NextLevelTotalExperience()
```

Returns

long

The experience threshold for the next level.

SetTotalCurrentExp(long)

Sets the total current experience and updates the level accordingly. The level will be recalculated based on the new experience total.

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

Parameters

`totalCurrentExperience` long

The total experience to set.

ValidateExperience()

Validates that the current experience matches the current level. Used for debugging and ensuring data consistency.

```
void ValidateExperience()
```

Interface IStatSetSource

Namespace: ElectricDrill.SoapRpgFramework

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that provide a source of stat sets. Defines the contract for accessing a StatSet that contains stat definitions.

```
public interface IStatSetSource
```

Properties

StatSet

Gets the stat set that defines the available stats.

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

Property Value

StatSet

Namespace ElectricDrill.SoapRpgFramework.Attributes

Classes

Attribute

Represents a character attribute that extends BoundedValue functionality. An attribute is a measurable characteristic such as strength, dexterity, luck, and so on. Can have a min and a max value.

AttributeMenuItems

Provides Unity Editor menu items for creating attributes.

AttributePointsTracker

Manages the allocation and tracking of attribute points for character development. Handles spending, refunding, and validation of attribute point investments.

AttributeSet

A ScriptableObject that defines a collection of attributes for use in character systems. Attribute sets are usually used to group attributes that can be shared across different characters or classes. For example, any character of any class could have the same attribute set, including attributes like strength, dexterity, constitution, intelligence, and luck.

AttributeSetInstance

Represents a runtime instance of an AttributeSet with actual values for each attribute. Provides methods to manipulate and access attribute values during gameplay.

AttributeSetInstanceExtensions

Extension methods for converting SerializableDictionary to AttributeSetInstance.

AttributeSetMenuItems

Provides Unity Editor menu items for creating attribute sets.

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.SoapRpgFramework.Attributes.EntityAttributes.EntityCore component on the same GameObject.

Structs

AttributeChangeInfo

Immutable structure that contains information about an attribute value change. Used to track and communicate attribute modifications across the system.

Interfaces

IAttributeContainer

Interface for objects that can contain and query attributes. Provides a standardized way to check for attribute containment across different attribute containers.

Class Attribute

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Represents a character attribute that extends BoundedValue functionality. An attribute is a measurable characteristic such as strength, dexterity, luck, and so on. Can have a min and a max value.

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

Inheritance

object ← Object ← ScriptableObject ← BoundedValue ← Attribute

Inherited Members

BoundedValue.HasMaxValue , BoundedValue.MaxValue , BoundedValue.HasMinValue , BoundedValue.MinValue

Struct AttributeChangeInfo

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Immutable structure that contains information about an attribute value change. Used to track and communicate attribute modifications across the system.

```
public readonly struct AttributeChangeInfo
```

Constructors

AttributeChangeInfo(EntityAttributes, Attribute, long, long)

Initializes a new instance of the AttributeChangeInfo structure.

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

Parameters

`source` EntityAttributes

The EntityAttributes instance that owns the changed attribute

`attribute` Attribute

The attribute that was changed

`oldValue` long

The previous value before the change

`newValue` long

The new value after the change

Properties

Attribute

Gets the attribute that was changed.

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

Property Value

Attribute

NewValue

Gets the new value of the attribute after the change.

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

Property Value

long

OldValue

Gets the previous value of the attribute before the change.

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

Property Value

long

Source

Gets the EntityAttributes instance that owns the changed attribute.

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

Property Value

EntityAttributes

Class AttributeMenuItems

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Provides Unity Editor menu items for creating attributes.

```
public static class AttributeMenuItems
```

Inheritance

object ← AttributeMenuItems

Methods

CreateAttribute()

Creates a new Attribute asset in the project window. Can be accessed via Assets/Create/Soap RPG Framework/Attribute menu or Ctrl+Alt+A shortcut.

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

Class AttributePointsTracker

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Manages the allocation and tracking of attribute points for character development. Handles spending, refunding, and validation of attribute point investments.

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

Inheritance

object ← AttributePointsTracker

Properties

Available

Gets the number of attribute points currently available to spend.

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

Property Value

int

SpentAttributesKeys

Gets the collection of attributes that have had points spent on them.

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

Property Value

Dictionary<Attribute, int>.KeyCollection

Methods

AddPoints(int)

Adds the specified amount of points to the available point pool.

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

Parameters

`amount` int

The number of points to add

GetSpentOn(Attribute)

Gets the number of points spent on the specified attribute.

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

Parameters

`attribute` Attribute

The attribute to query

Returns

long

The number of points spent on the attribute

Refund(Attribute)

Refunds all points spent on the specified attribute.

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

Parameters

`attribute` Attribute

The attribute to fully refund

Refund(Attribute, int)

Refunds the specified amount of points from the given attribute.

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

Parameters

`attribute` Attribute

The attribute to refund points from

`amount` int

The number of points to refund

RefundAll()

Refunds all spent points from all attributes, returning them to the available pool.

```
public void RefundAll()
```

SpendOn(Attribute, int)

Spends the specified amount of points on the given attribute.

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

Parameters

`attribute` Attribute

The attribute to invest points in

`amount` int

The number of points to spend

Class AttributeSet

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A ScriptableObject that defines a collection of attributes for use in character systems. Attribute sets are usually used to group attributes that can be shared across different characters or classes. For example, any character of any class could have the same attribute set, including attributes like strength, dexterity, constitution, intelligence, and luck.

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

Inheritance

object ← Object ← ScriptableObject ← AttributeSet

Implements

IAttributeContainer

Properties

Attributes

Gets a read-only list of all attributes contained in this set.

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

Property Value

IReadOnlyList<Attribute>

Methods

Contains(Attribute)

Checks whether the specified attribute is contained in this set.

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

Parameters

`attribute` Attribute

The attribute to check for

Returns

bool

True if the attribute is in the set, false otherwise

Class AttributeSetInstance

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Represents a runtime instance of an AttributeSet with actual values for each attribute. Provides methods to manipulate and access attribute values during gameplay.

```
public class AttributeSetInstance : IAttributeContainer
```

Inheritance

object ← AttributeSetInstance

Implements

IAttributeContainer

Constructors

AttributeSetInstance(AttributeSet)

Initializes a new instance of AttributeSetInstance based on an AttributeSet. All attributes are initialized with a value of 0.

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

Parameters

`attrSet` AttributeSet

The AttributeSet to base this instance on

Properties

Attributes

Gets the internal dictionary containing all attribute-value pairs.

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

Property Value

Dictionary<Attribute, long>

this[Attribute]

Gets or sets the value of an attribute using indexer syntax.

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

Parameters

`attribute` Attribute

The attribute to access

Property Value

long

The current value when getting, or sets the value when setting

Methods

AddValue(Attribute, long)

Adds a value to the specified attribute. If the attribute doesn't exist, it will be created.

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

Parameters

`attribute` Attribute

The attribute to modify

`value` long

The value to add. Can be negative to decrease the value.

Clone()

Creates a deep copy of this AttributeSetInstance.

```
public AttributeSetInstance Clone()
```

Returns

AttributeSetInstance

A new AttributeSetInstance with the same attribute values

Contains(Attribute)

Checks whether the specified attribute is contained in this instance.

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

Parameters

`stat` Attribute

The attribute to check for

Returns

bool

True if the attribute exists, false otherwise

Get(Attribute)

Gets the current value of the specified attribute.

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

Parameters

`attribute` Attribute

The attribute to retrieve

Returns

long

The current value of the attribute

GetAsPercentage(Attribute)

Gets the value of an attribute as a percentage.

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

Parameters

`stat` Attribute

The attribute to convert to percentage

Returns

Percentage

A Percentage representation of the attribute value

GetEnumerator()

Returns an enumerator that iterates through the attribute-value pairs.

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

Returns

IEnumerator<KeyValuePair<Attribute, long>>

An enumerator for the attribute-value pairs

Operators

operator +(AttributeSetInstance, AttributeSetInstance)

Adds two AttributeSetInstance objects together, combining their attribute values.

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

Parameters

a AttributeSetInstance

The first AttributeSetInstance

b AttributeSetInstance

The second AttributeSetInstance

Returns

AttributeSetInstance

A new AttributeSetInstance with combined values

explicit operator AttributeSetInstance(SerializableDictionary<Attribute, long>)

Explicitly converts a SerializableDictionary to an AttributeSetInstance.

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

Parameters

`dictionary` SerializableDictionary<Attribute, long>

The dictionary to convert

Returns

AttributeSetInstance

A new AttributeSetInstance based on the dictionary

Class AttributeSetInstanceExtensions

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Extension methods for converting SerializableDictionary to AttributeSetInstance.

```
public static class AttributeSetInstanceExtensions
```

Inheritance

object ← AttributeSetInstanceExtensions

Methods

ToAttributeSetInstance(SerializableDictionary<Attribute, long>, AttributeSet)

Converts a SerializableDictionary of attributes to an AttributeSetInstance. Validates that all attributes from the AttributeSet are present in the dictionary.

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

Parameters

`dictionary` SerializableDictionary<Attribute, long>

The dictionary to convert

`attributeSet` AttributeSet

The AttributeSet to validate against

Returns

AttributeSetInstance

A new AttributeSetInstance with values from the dictionary

Class AttributeSetMenuItems

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Provides Unity Editor menu items for creating attribute sets.

```
public static class AttributeSetMenuItems
```

Inheritance

object ← AttributeSetMenuItems

Methods

CreateAttributeSet()

Creates a new AttributeSet asset in the project window. Can be accessed via Assets/Create/Soap RPG Framework/Attribute Set menu.

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

Class EntityAttributes

Namespace: ElectricDrill.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework.Attributes.EntityAttributes.EntityCore component on the same GameObject.

```
[DisallowMultipleComponent]
[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.SoapRpgFramework.Attributes.EntityAttributes.useClassBaseAttributes is true. If true, it returns the attribute set from the EntityClass. Otherwise, it returns the ElectricDrill.SoapRpgFramework.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.SoapRpgFramework.Attributes.EntityAttributes.useCache flag.

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

Property Value

Cache<Attribute, long>

AvailableAttributePoints

Gets the number of attribute points currently available to spend.

```
public int AvailableAttributePoints { get; }
```

Property Value

int

EntityClass

Gets the class for this entity. The class is used to determine base attributes when ElectricDrill.SoapRpg Framework.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.SoapRpgFramework.Attributes

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that can contain and query attributes. Provides a standardized way to check for attribute containment across different attribute containers.

```
public interface IAttributeContainer
```

Methods

Contains(Attribute)

Determines whether the container includes the specified attribute.

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

Parameters

`attribute` Attribute

The attribute to locate in the container

Returns

bool

True if the attribute is found in the container; otherwise, false

Namespace ElectricDrill.SoapRpgFramework.Events

Classes

AttributeChangedGameEvent

Game event that is raised when an attribute changes. Contains information about the attribute that changed, including its previous and new values.

AttributeChangedGameEventListener

Event listener that responds to AttributeChangedGameEvent events. Receives information about the attribute that changed, including its previous and new values.

EntityCoreGameEvent

Game event that carries EntityCore data. Used for events that need to pass information about an entity's core component.

EntityCoreGameEventListener

Event listener that responds to EntityCoreGameEvent events. Receives EntityCore data when the associated game event is raised.

GameEvent

A ScriptableObject-based event system that allows decoupled communication between game systems. GameEvents can be raised to notify all registered listeners without requiring direct references.

GameEventGenerator

A ScriptableObject tool for automatically generating GameEvent and GameEventListener classes. Allows developers to define event schemas and generate strongly-typed event classes at design time.

GameEventGenerator.EventParameter

Represents a parameter that can be passed to a generated game event. Supports both native C# types and custom MonoScript-based types.

GameEventGenerator.GameEventDefinition

Defines a complete game event with its name, parameters, and metadata. Contains all information needed to generate the corresponding C# classes.

GameEventGeneric1<T>

A generic ScriptableObject-based event system that can carry one parameter of type T. Supports both MonoBehaviour-based listeners and code-based Action listeners.

GameEventGeneric2<T, U>

A generic ScriptableObject-based event system that can carry two parameters of types T and U. Supports both MonoBehaviour-based listeners and code-based Action listeners.

GameEventGeneric3<T, U, W>

A generic ScriptableObject-based event system that can carry three parameters of types T, U, and W. Supports both MonoBehaviour-based listeners and code-based Action listeners.

GameEventGeneric4<T, U, W, K>

A generic ScriptableObject-based event system that can carry four parameters of types T, U, W, and K. Supports both MonoBehaviour-based listeners and code-based Action listeners.

GameEventListener

MonoBehaviour component that listens to GameEvent objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

GameEventListenerGeneric1<T>

MonoBehaviour component that listens to GameEventGeneric1 objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

GameEventListenerGeneric2<T, U>

MonoBehaviour component that listens to GameEventGeneric2 objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

GameEventListenerGeneric3<T, U, W>

MonoBehaviour component that listens to GameEventGeneric3 objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

GameEventListenerGeneric4<T, U, W, K>

MonoBehaviour component that listens to GameEventGeneric4 objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

IntGameEvent

Game event that carries integer data. A general-purpose event for passing integer values between game systems.

IntGameEventListener

Event listener that responds to IntGameEvent events. Receives integer data when the associated game event is raised.

StatChangedGameEvent

Game event that is raised when a stat changes. Contains information about the stat that changed, including its previous and new values.

StatChangedGameEventListener

Event listener that responds to StatChangedGameEvent events. Receives information about the stat that changed, including its previous and new values.

Interfaces

IRaisable<T>

Interface for objects that can be raised with one parameter of type T. Defines the contract for events that carry a single piece of data.

IRaisable<T, U>

Interface for objects that can be raised with two parameters of types T and U. Defines the contract for events that carry two pieces of data.

IRaisable<T, U, V>

Interface for objects that can be raised with three parameters of types T, U, and V. Defines the contract for events that carry three pieces of data.

IRaisable<T, U, V, W>

Interface for objects that can be raised with four parameters of types T, U, V, and W. Defines the contract for events that carry four pieces of data.

Enums

GameEventGenerator.EventParameter.NativeType

Enumeration of supported native C# types for event parameters.

GameEventGenerator.EventParameter.ParameterType

Enumeration defining whether the parameter is a native type or a custom MonoScript type.

Class AttributeChangedGameEvent

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Game event that is raised when an attribute changes. Contains information about the attribute that changed, including its previous and new values.

```
[CreateAssetMenu(fileName = "AttributeChanged Game Event", menuName = "Soap RPG
Framework/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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Event listener that responds to AttributeChangedGameEvent events. Receives information about the attribute that changed, including its previous and new values.

```
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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Game event that carries EntityCore data. Used for events that need to pass information about an entity's core component.

```
[CreateAssetMenu(fileName = "EntityCore Game Event", menuName = "Soap RPG
Framework/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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Event listener that responds to EntityCoreGameEvent events. Receives EntityCore data when the associated game event is raised.

```
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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A ScriptableObject-based event system that allows decoupled communication between game systems. GameEvents can be raised to notify all registered listeners without requiring direct references.

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

Inheritance

object ← Object ← ScriptableObject ← GameEvent

Methods

Raise()

Raises the event, notifying all registered listeners. Listeners are processed in reverse order to handle potential modifications during iteration.

```
public void Raise()
```

RegisterListener(GameEventListener)

Registers a listener to be notified when this event is raised. Duplicate listeners are automatically prevented.

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

Parameters

`listener` GameEventListener

The GameEventListener to register.

UnregisterListener(GameEventListener)

Unregisters a listener so it will no longer be notified when this event is raised.

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

Parameters

`listener` GameEventListener

The GameEventListener to unregister.

Class GameEventGenerator

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A ScriptableObject tool for automatically generating GameEvent and GameEventListener classes. Allows developers to define event schemas and generate strongly-typed event classes at design time.

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

Inheritance

object ← Object ← ScriptableObject ← GameEventGenerator

Fields

baseSaveLocation

Gets or sets the base directory where generated files will be saved.

```
public string baseSaveLocation
```

Field Value

string

eventsToGenerate

Gets or sets the list of event definitions to generate classes for.

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

Field Value

List<GameEventGenerator.GameEventDefinition>

menuBasePath

Gets or sets the menu path prefix for CreateAssetMenu attributes on generated events.

```
public string menuBasePath
```

Field Value

string

rootNamespace

Gets or sets the root namespace for generated event classes.

```
public string rootNamespace
```

Field Value

string

Methods

GenerateGameEvents()

Generates C# class files for all defined game events and their corresponding listeners. Creates the necessary directory structure and writes the generated code to disk.

```
public void GenerateGameEvents()
```

RemoveGeneratedEventFiles(string, int)

Removes the generated files for a specific event from the file system.

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

Parameters

`eventName` string

The name of the event whose files should be removed

`parameterCount` int

The number of parameters the event had

Class GameEventGenerator.EventParameter

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Represents a parameter that can be passed to a generated game event. Supports both native C# types and custom MonoScript-based types.

```
[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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Enumeration of supported native C# types for event parameters.

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

Fields

bool = 3

float = 2

int = 0

long = 1

Enum GameEventGenerator.EventParameter.ParameterType

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Enumeration defining whether the parameter is a native type or a custom MonoScript type.

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

Fields

`MonoScript = 1`

`Native = 0`

Class GameEventGenerator.GameEventDefinition

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Defines a complete game event with its name, parameters, and metadata. Contains all information needed to generate the corresponding C# classes.

```
[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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A generic ScriptableObject-based event system that can carry one parameter of type T. Supports both MonoBehaviour-based listeners and code-based Action listeners.

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

Type Parameters

`T`

The type of data this event carries.

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric1<T>

Implements

IRaisable<T>

Derived

AttributeChangedGameEvent, EntityCoreGameEvent, IntGameEvent, StatChangedGameEvent

Methods

Raise(T)

Raises the event with the specified context, notifying all registered listeners. Both MonoBehaviour listeners and Action listeners are notified.

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

Parameters

`context` T

The data to pass to all listeners.

RegisterListener(GameEventListenerGeneric1<T>)

Registers a MonoBehaviour-based listener to be notified when this event is raised. Duplicate listeners are automatically prevented.

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

Parameters

`listener` GameEventListenerGeneric1<T>

The GameEventListenerGeneric1 to register.

UnregisterListener(GameEventListenerGeneric1<T>)

Unregisters a MonoBehaviour-based listener so it will no longer be notified when this event is raised.

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

Parameters

`listener` GameEventListenerGeneric1<T>

The GameEventListenerGeneric1 to unregister.

Events

OnEventRaised

Event that can be subscribed to from code using standard C# event syntax. Automatically handles registration and unregistration of Action listeners.

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

Event Type

Action<T>

Class GameEventGeneric2<T, U>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A generic ScriptableObject-based event system that can carry two parameters of types T and U. Supports both MonoBehaviour-based listeners and code-based Action listeners.

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

Type Parameters

`T`

The type of the first parameter this event carries.

`U`

The type of the second parameter this event carries.

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric2<T, U>

Implements

IRaisable<T, U>

Methods

Raise(T, U)

Raises the event with the specified contexts, notifying all registered listeners. Both MonoBehaviour listeners and Action listeners are notified.

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

Parameters

`context1` T

The first parameter to pass to all listeners.

`context2` U

The second parameter to pass to all listeners.

RegisterListener(GameEventListenerGeneric2<T, U>)

Registers a MonoBehaviour-based listener to be notified when this event is raised. Duplicate listeners are automatically prevented.

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

Parameters

`listener` GameEventListenerGeneric2<T, U>

The GameEventListenerGeneric2 to register.

UnregisterListener(GameEventListenerGeneric2<T, U>)

Unregisters a MonoBehaviour-based listener so it will no longer be notified when this event is raised.

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

Parameters

`listener` GameEventListenerGeneric2<T, U>

The GameEventListenerGeneric2 to unregister.

Events

OnEventRaised

Event that can be subscribed to from code using standard C# event syntax. Automatically handles registration and unregistration of Action listeners.

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

Event Type

Action<T, U>

Class GameEventGeneric3<T, U, W>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A generic ScriptableObject-based event system that can carry three parameters of types T, U, and W. Supports both MonoBehaviour-based listeners and code-based Action listeners.

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

Type Parameters

`T`

The type of the first parameter this event carries.

`U`

The type of the second parameter this event carries.

`W`

The type of the third parameter this event carries.

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric3<T, U, W>

Implements

IRaisable<T, U, W>

Methods

Raise(T, U, W)

Raises the event with the specified contexts, notifying all registered listeners. Both MonoBehaviour listeners and Action listeners are notified.

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

Parameters

`contextT` T

The first parameter to pass to all listeners.

`contextU` U

The second parameter to pass to all listeners.

`contextW` W

The third parameter to pass to all listeners.

RegisterListener(GameEventListenerGeneric3<T, U, W>)

Registers a MonoBehaviour-based listener to be notified when this event is raised. Duplicate listeners are automatically prevented.

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

Parameters

`listener` GameEventListenerGeneric3<T, U, W>

The GameEventListenerGeneric3 to register.

UnregisterListener(GameEventListenerGeneric3<T, U, W>)

Unregisters a MonoBehaviour-based listener so it will no longer be notified when this event is raised.

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

Parameters

`listener` GameEventListenerGeneric3<T, U, W>

The GameEventListenerGeneric3 to unregister.

Events

OnEventRaised

Event that can be subscribed to from code using standard C# event syntax. Automatically handles registration and unregistration of Action listeners.

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

Event Type

Action<T, U, W>

Class GameEventGeneric4<T, U, W, K>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A generic ScriptableObject-based event system that can carry four parameters of types T, U, W, and K. Supports both MonoBehaviour-based listeners and code-based Action listeners.

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

Type Parameters

T

The type of the first parameter this event carries.

U

The type of the second parameter this event carries.

W

The type of the third parameter this event carries.

K

The type of the fourth parameter this event carries.

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric4<T, U, W, K>

Implements

IRaisable<T, U, W, K>

Methods

Raise(T, U, W, K)

Raises the event with the specified contexts, notifying all registered listeners. Both MonoBehaviour listeners and Action listeners are notified.

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

Parameters

`contextT` T

The first parameter to pass to all listeners.

`contextU` U

The second parameter to pass to all listeners.

`contextW` W

The third parameter to pass to all listeners.

`contextK` K

The fourth parameter to pass to all listeners.

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

Registers a MonoBehaviour-based listener to be notified when this event is raised. Duplicate listeners are automatically prevented.

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

Parameters

`listener` GameEventListenerGeneric4<T, U, W, K>

The GameEventListenerGeneric4 to register.

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

Unregisters a MonoBehaviour-based listener so it will no longer be notified when this event is raised.

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

Parameters

`listener` GameEventListenerGeneric4<T, U, W, K>

The GameEventListenerGeneric4 to unregister.

Events

OnEventRaised

Event that can be subscribed to from code using standard C# event syntax. Automatically handles registration and unregistration of Action listeners.

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

Event Type

Action<T, U, W, K>

Class GameEventListener

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

MonoBehaviour component that listens to GameEvent objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

```
public class GameEventListener : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← GameEventListener

Methods

OnEventRaised()

Called by the GameEvent when it is raised. Invokes the configured UnityEvent response.

```
public void OnEventRaised()
```

Class GameEventListenerGeneric1<T>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

MonoBehaviour component that listens to GameEventGeneric1 objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

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

Type Parameters

T

The type of parameter the listened event carries.

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

UnityEvent<T>

Methods

OnEventRaised(T)

Called by the GameEvent when it is raised. Invokes the configured UnityEvent response with the provided context.

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

Parameters

`context` T

The data passed from the event.

Class GameEventListenerGeneric2<T, U>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

MonoBehaviour component that listens to GameEventGeneric2 objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

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

Type Parameters

T

The type of the first parameter the listened event carries.

U

The type of the second parameter the listened event carries.

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)

Called by the GameEvent when it is raised. Invokes the configured UnityEvent response with the provided contexts.

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

Parameters

`contextT` T

The first parameter passed from the event.

`contextU` U

The second parameter passed from the event.

Class GameEventListenerGeneric3<T, U, W>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

MonoBehaviour component that listens to GameEventGeneric3 objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

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

Type Parameters

T

The type of the first parameter the listened event carries.

U

The type of the second parameter the listened event carries.

W

The type of the third parameter the listened event carries.

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

UnityEvent<T, U, W>

Methods

OnEventRaised(T, U, W)

Called by the GameEvent when it is raised. Invokes the configured UnityEvent response with the provided contexts.

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

Parameters

`contextT` T

The first parameter passed from the event.

`contextU` U

The second parameter passed from the event.

`contextW` W

The third parameter passed from the event.

Class GameEventListenerGeneric4<T, U, W, K>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

MonoBehaviour component that listens to GameEventGeneric4 objects and responds with UnityEvents. Automatically registers and unregisters itself when enabled/disabled.

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

Type Parameters

T

The type of the first parameter the listened event carries.

U

The type of the second parameter the listened event carries.

W

The type of the third parameter the listened event carries.

K

The type of the fourth parameter the listened event carries.

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)

Called by the GameEvent when it is raised. Invokes the configured UnityEvent response with the provided contexts.

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

Parameters

`contextT` T

The first parameter passed from the event.

`contextU` U

The second parameter passed from the event.

`contextW` W

The third parameter passed from the event.

`contextK` K

The fourth parameter passed from the event.

Interface IRaisable<T>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that can be raised with one parameter of type T. Defines the contract for events that carry a single piece of data.

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

Type Parameters

`T`

The type of data this raisable object carries.

Methods

Raise(T)

Raises the event with the specified context.

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

Parameters

`context` T

The data to pass when raising the event.

Interface IRaisable<T, U>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that can be raised with two parameters of types T and U. Defines the contract for events that carry two pieces of data.

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

Type Parameters

`T`

The type of the first parameter.

`U`

The type of the second parameter.

Methods

Raise(T, U)

Raises the event with the specified contexts.

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

Parameters

`context1` T

The first parameter to pass when raising the event.

`context2` U

The second parameter to pass when raising the event.

Interface IRaisable<T, U, V>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that can be raised with three parameters of types T, U, and V. Defines the contract for events that carry three pieces of data.

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

Type Parameters

`T`

The type of the first parameter.

`U`

The type of the second parameter.

`V`

The type of the third parameter.

Methods

Raise(T, U, V)

Raises the event with the specified contexts.

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

Parameters

`context1` T

The first parameter to pass when raising the event.

`context2` U

The second parameter to pass when raising the event.

`context3` V

The third parameter to pass when raising the event.

Interface IRaisable<T, U, V, W>

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that can be raised with four parameters of types T, U, V, and W. Defines the contract for events that carry four pieces of data.

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

Type Parameters

`T`

The type of the first parameter.

`U`

The type of the second parameter.

`V`

The type of the third parameter.

`W`

The type of the fourth parameter.

Methods

Raise(T, U, V, W)

Raises the event with the specified contexts.

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

Parameters

`context1` T

The first parameter to pass when raising the event.

`context2` U

The second parameter to pass when raising the event.

`context3` V

The third parameter to pass when raising the event.

`context4` W

The fourth parameter to pass when raising the event.

Class IntGameEvent

Namespace: ElectricDrill.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Game event that carries integer data. A general-purpose event for passing integer values between game systems.

```
[CreateAssetMenu(fileName = "Int Game Event", menuName = "Soap RPG
Framework/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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Event listener that responds to IntGameEvent events. Receives integer data when the associated game event is raised.

```
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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Game event that is raised when a stat changes. Contains information about the stat that changed, including its previous and new values.

```
[CreateAssetMenu(fileName = "StatChanged Game Event", menuName = "Soap RPG
Framework/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.SoapRpgFramework.Events

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Event listener that responds to StatChangedGameEvent events. Receives information about the stat that changed, including its previous and new values.

```
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.SoapRpgFramework.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>

Abstract base class for scaling components that work with ScriptableObject-based sets.

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.SoapRpgFramework.Scaling

Assembly: com.electricdrill.soap-rpg-framework.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/Soap RPG Framework/Scaling/Attribute Scaling Component", false, 1)]
public static void CreateAttributeScalingComponent()
```

Class AttributesScalingComponent

Namespace: ElectricDrill.SoapRpgFramework.Scaling

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework.Scaling

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework.Scaling

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework.Scaling

Assembly: com.electricdrill.soap-rpg-framework.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/Soap RPG Framework/Scaling/Scaling Formula &#S", false, 0)]
public static void CreateScalingFormula()
```

Class SoSetScalingComponentBase<SetType, KeyType>

Namespace: ElectricDrill.SoapRpgFramework.Scaling

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Abstract base class for scaling components that work with ScriptableObject-based sets.

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

Type Parameters

`SetType`

The type of ScriptableObject set that contains the items used for defining the scaling value

`KeyType`

The type of items within the set that can be used to calculate the scaling value

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 final scaled value based on the entity's values and the configured scaling multipliers.

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

Parameters

`entity` EntityCore

The entity to calculate the scaled value for

Returns

long

The calculated scaled value as a long integer

GetEntitySet(EntityCore)

Gets the set associated with the specified entity. This method must be implemented by derived classes to define how to retrieve the entity's set.

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

Parameters

`entity` EntityCore

The entity to get the set for

Returns

SetType

The set associated with the entity

GetEntityValue(EntityCore, KeyType)

Gets the current value of a specific key for the given entity. This method must be implemented by derived classes to define how to retrieve entity values.

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

Parameters

`entity` EntityCore

The entity to get the value from

`key` KeyType

The specific key to get the value for

Returns

long

The current value of the specified key for the entity

GetSetItems()

Gets all items from the current set that can be used for scaling. This method must be implemented by derived classes to define which items are available for scaling.

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

Returns

IEnumerable<KeyType>

An enumerable collection of all scalable items in the set

OnValidate()

Validates and refreshes the scaling attribute values based on the current set. Called automatically when the component is validated in the editor.

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

Class StatScalingComponentMenuItems

Namespace: ElectricDrill.SoapRpgFramework.Scaling

Assembly: com.electricdrill.soap-rpg-framework.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/Soap RPG Framework/Scaling/Stat Scaling Component", false, 2)]
public static void CreateStatScalingComponent()
```

Class StatsScalingComponent

Namespace: ElectricDrill.SoapRpgFramework.Scaling

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework.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

Represents a stat in the RPG system that extends BoundedValue with attribute scaling capabilities.
Stats can be scaled based on entity attributes and provide equality comparison based on name.

StatMenuItems

Provides Unity Editor menu items for creating Stat assets.

StatSet

A ScriptableObject that defines a collection of stats that can be used by entities. Implements IStatContainer to provide stat containment functionality.

StatSetInstance

Represents an instance of a StatSet with actual values for each stat. Implements IEnumerable and IStatContainer to provide collection functionality.

StatSetMenuItems

Provides Unity Editor menu items for creating StatSet assets.

StatToStatModifier

A ScriptableObject that defines how one stat can scale upon another stat basing on a percentage.
Used to create relationships between different stats in the RPG system.

StatToStatModifierMenuItems

Provides Unity Editor menu items for creating StatToStatModifier assets.

Interfaces

IStatContainer

Interface for objects that can contain and check for the presence of stats. Provides basic functionality for stat container validation.

Class EntityStats

Namespace: ElectricDrill.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.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.

```
[DisallowMultipleComponent]
[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 AddStatToStatModifer(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.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Interface for objects that can contain and check for the presence of stats. Provides basic functionality for stat container validation.

```
public interface IStatContainer
```

Methods

Contains(Stat)

Determines whether the container contains the specified stat.

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

Parameters

`stat` Stat

The stat to check for.

Returns

bool

true if the container contains the stat; otherwise, false.

Class Stat

Namespace: ElectricDrill.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Represents a stat in the RPG system that extends BoundedValue with attribute scaling capabilities. Stats can be scaled based on entity attributes and provide equality comparison based on name.

```
public class Stat : BoundedValue
```

Inheritance

object ← Object ← ScriptableObject ← BoundedValue ← Stat

Inherited Members

BoundedValue.HasMaxValue , BoundedValue.MaxValue , BoundedValue.HasMinValue , BoundedValue.MinValue

Properties

AttributesScaling

Gets or sets the attributes scaling component that determines how this stat scales with entity attributes. Can be null if the stat doesn't scale with attributes.

```
[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)

Determines whether two Stat objects are equal based on their names.

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

Parameters

`a` Stat

The first stat to compare.

`b` Stat

The second stat to compare.

Returns

bool

true if the stats are equal; otherwise, false.

operator !=(Stat, Stat)

Determines whether two Stat objects are not equal.

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

Parameters

a Stat

The first stat to compare.

b Stat

The second stat to compare.

Returns

bool

true if the stats are not equal; otherwise, false.

Class StatMenuItems

Namespace: ElectricDrill.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Provides Unity Editor menu items for creating Stat assets.

```
public static class StatMenuItems
```

Inheritance

object ← StatMenuItems

Methods

CreateStat()

Creates a new Stat asset in the project window.

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

Class StatSet

Namespace: ElectricDrill.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A ScriptableObject that defines a collection of stats that can be used by entities. Implements IStatContainer to provide stat containment functionality.

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

Inheritance

object ← Object ← ScriptableObject ← StatSet

Implements

IStatContainer

Properties

Stats

Gets a read-only list of all stats in this set.

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

Property Value

IReadOnlyList<Stat>

Methods

Contains(Stat)

Determines whether this stat set contains the specified stat.

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

Parameters

`stat` Stat

The stat to check for.

Returns

bool

true if the stat set contains the stat; otherwise, false.

Get(Stat)

Gets the stat from this set that matches the specified stat.

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

Parameters

`stat` Stat

The stat to find.

Returns

Stat

The matching stat from this set.

Class StatSetInstance

Namespace: ElectricDrill.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Represents an instance of a StatSet with actual values for each stat. Implements IEnumerable and IStatContainer to provide collection functionality.

```
public class StatSetInstance : IStatContainer
```

Inheritance

object ← StatSetInstance

Implements

IStatContainer

Constructors

StatSetInstance(StatSet)

Initializes a new StatSetInstance based on the provided StatSet. All stats from the StatSet are initialized with a value of 0.

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

Parameters

`statSet` StatSet

The StatSet to base this instance on.

Properties

this[Stat]

Gets or sets the value of the specified stat using indexer syntax.

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

Parameters

`stat` Stat

The stat to get or set.

Property Value

long

The current value of the stat.

Methods

AddValue(Stat, long)

Adds value to the specified stat. If the stat doesn't exist, it will be created with the specified value. Use negative values to subtract from the stat.

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

Parameters

`stat` Stat

The stat to modify.

`value` long

The value to add (can be negative).

Clone()

Creates a deep copy of this StatSetInstance.

```
public StatSetInstance Clone()
```

Returns

StatSetInstance

A new StatSetInstance with the same stats and values.

Contains(Stat)

Determines whether this instance contains the specified stat.

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

Parameters

`stat` Stat

The stat to check for.

Returns

bool

true if the instance contains the stat; otherwise, false.

Get(Stat)

Gets the current value of the specified stat.

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

Parameters

`stat` Stat

The stat to retrieve.

Returns

long

The current value of the stat.

GetAsPercentage(Stat)

Gets the value of the specified stat as a Percentage.

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

Parameters

`stat` Stat

The stat to get as a percentage.

Returns

Percentage

The stat value wrapped in a Percentage object.

GetEnumerator()

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

Returns

IEnumerator<KeyValuePair<Stat, long>>

Operators

operator +(StatSetInstance, StatSetInstance)

Adds the values of corresponding stats from two StatSetInstances. All stats present in the first instance must also be present in the second instance.

```
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 corresponding stat values.

Exceptions

KeyNotFoundException

Thrown when a stat from instance 'a' is not found in instance 'b'.

Class StatSetMenuItems

Namespace: ElectricDrill.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Provides Unity Editor menu items for creating StatSet assets.

```
public static class StatSetMenuItems
```

Inheritance

object ← StatSetMenuItems

Methods

CreateStatSet()

Creates a new StatSet asset in the project window.

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

Class StatToStatModifier

Namespace: ElectricDrill.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A ScriptableObject that defines how one stat can scale upon another stat basing on a percentage. Used to create relationships between different stats in the RPG system.

```
public class StatToStatModifier : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← StatToStatModifier

Properties

Percentage

Gets the percentage of the source stat value that will be applied to the target stat.

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

Property Value

Percentage

SourceStat

Gets the stat that provides the value for the modification calculation.

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

Property Value

Stat

TargetStat

Gets the stat that will be modified by the source stat.

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

Property Value

Stat

Class StatToStatModifierMenuItems

Namespace: ElectricDrill.SoapRpgFramework.Stats

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Provides Unity Editor menu items for creating StatToStatModifier assets.

```
public static class StatToStatModifierMenuItems
```

Inheritance

object ← StatToStatModifierMenuItems

Methods

CreateStatToStatModifier()

Creates a new StatToStatModifier asset in the project window.

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

Namespace ElectricDrill.SoapRpgFramework.Utils

Classes

Cache<KType, VType>
: A generic cache class that provides basic caching functionality.

InitializationUtils
: Provides utility methods for initializing and refreshing collections during object setup. Contains helper methods commonly used during component initialization and validation.

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>
: A serializable dictionary implementation that can be displayed and edited in the Unity Inspector. Provides all standard dictionary functionality while maintaining Unity serialization compatibility.

SerializableHashSet<T>
: A serializable hash set implementation that can be displayed and edited in the Unity Inspector. Provides all standard hash set functionality while maintaining Unity serialization compatibility.

Structs

SerKeyValPair<T, U>

A serializable key-value pair structure that can be used in Unity's inspector. Provides implicit conversion to and from the standard KeyValuePair.

Class Cache<KType, VType>

Namespace: ElectricDrill.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.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.

Class InitializationUtils

Namespace: ElectricDrill.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

Provides utility methods for initializing and refreshing collections during object setup. Contains helper methods commonly used during component initialization and validation.

```
public static class InitializationUtils
```

Inheritance

object ← InitializationUtils

Methods

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

Refreshes a list of key-value pairs to match a provided set of keys, preserving existing values where possible. This method ensures that the inspector-reserved values list contains exactly the keys specified, maintaining any previously assigned values and setting defaults for new keys.

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

Parameters

`inspectorReservedValues` List<SerKeyValPair<TKey, TValue>>

The list to refresh (passed by reference)

`keys` IEnumerable<TKey>

The collection of keys that should be present in the list

Type Parameters

`TKey`

The type of the keys

`TValue`

The type of the values

Class IntRef

Namespace: ElectricDrill.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

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

```
[CreateAssetMenu(fileName = "Int Var", menuName = "Soap RPG Framework/Utils/Int Var")]
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.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.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.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

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

```
[CreateAssetMenu(fileName = "Long Var", menuName = "Soap RPG Framework/Utils/Long Var")]
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.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.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 SerKeyValPair<T, U>

Namespace: ElectricDrill.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A serializable key-value pair structure that can be used in Unity's inspector. Provides implicit conversion to and from the standard KeyValuePair.

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

Type Parameters

`T`

The type of the key.

`U`

The type of the value.

Constructors

SerKeyValPair(T, U)

Initializes a new instance of the SerKeyValPair structure with the specified key and value.

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

Parameters

`key` T

The key of the key-value pair.

`value` U

The value of the key-value pair.

Fields

Key

The key of the key-value pair.

```
public T Key
```

Field Value

T

Value

The value of the key-value pair.

```
public U Value
```

Field Value

U

Operators

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

Implicitly converts a SerKeyValPair to a KeyValuePair.

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

Parameters

`serKeyValPair` SerKeyValPair<T, U>

The SerKeyValPair to convert.

Returns

KeyValuePair<T, U>

A KeyValuePair with the same key and value.

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

Implicitly converts a KeyValuePair to a SerKeyValPair.

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

Parameters

`keyValuePair` KeyValuePair<T, U>

The KeyValuePair to convert.

Returns

SerKeyValPair<T, U>

A SerKeyValPair with the same key and value.

Class SerializableDictionary<TKey, TValue>

Namespace: ElectricDrill.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A serializable dictionary implementation that can be displayed and edited in the Unity Inspector. Provides all standard dictionary functionality while maintaining Unity serialization compatibility.

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

Type Parameters

`TKey`

The type of the dictionary keys

`TValue`

The type of the dictionary values

Inheritance

object ← SerializableDictionary<TKey, TValue>

Properties

this[TKey]

Gets or sets the value associated with the specified key.

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

Parameters

`key` TKey

The key of the value to get or set

Property Value

TValue

The value associated with the specified key

Keys

Gets a collection containing the keys in the dictionary.

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

Property Value

Dictionary<TKey, TValue>.KeyCollection

Values

Gets a collection containing the values in the dictionary.

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

Property Value

Dictionary<TKey, TValue>.ValueCollection

Methods

Clear()

Removes all keys and values from the dictionary.

```
public void Clear()
```

ContainsKey(TKey)

Determines whether the dictionary contains the specified key.

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

Parameters

`key` TKey

The key to locate

Returns

bool

True if the dictionary contains the key; otherwise, false

GetEnumerator()

Returns an enumerator that iterates through the dictionary.

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

Returns

IEnumerator<KeyValuePair<TKey, TValue>>

An enumerator for the dictionary

OnAfterDeserialize()

Reconstructs the internal dictionary from the serialized list format. Called automatically by Unity after deserialization occurs.

```
public void OnAfterDeserialize()
```

OnBeforeSerialize()

Converts the internal dictionary to a serialized list format for Unity serialization. Called automatically by Unity before serialization occurs.

```
public void OnBeforeSerialize()
```

TryGetValue(TKey, out TValue)

Attempts to get the value associated with the specified key.

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

Parameters

`key` TKey

The key to locate

`value` TValue

The value associated with the key, if found

Returns

bool

True if the key was found; otherwise, false

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

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.SoapRpgFramework.Utils

Assembly: com.electricdrill.soap-rpg-framework.Runtime.dll

A serializable hash set implementation that can be displayed and edited in the Unity Inspector. Provides all standard hash set functionality while maintaining Unity serialization compatibility.

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

Type Parameters

T

The type of elements in the hash set

Inheritance

object ← SerializableHashSet<T>

Properties

Count

Gets the number of elements contained in the hash set.

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

Property Value

int

IsReadOnly

Gets a value indicating whether the hash set is read-only.

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

Property Value

bool

Methods

Add(T)

Adds the specified element to the hash set.

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

Parameters

`item` T

The element to add

Clear()

Removes all elements from the hash set.

```
public void Clear()
```

Contains(T)

Determines whether the hash set contains the specified element.

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

Parameters

`item` T

The element to locate

Returns

bool

True if the hash set contains the element; otherwise, false

CopyTo(T[], int)

Copies the elements of the hash set to an array, starting at the specified array index.

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

Parameters

`array` T[]

The destination array

`arrayIndex` int

The zero-based index in array at which copying begins

GetEnumerator()

Returns an enumerator that iterates through the hash set.

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

Returns

IEnumerator<T>

An enumerator for the hash set

GetObjectData(SerializationInfo, StreamingContext)

Populates a SerializationInfo with the data needed to serialize the hash set.

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

Parameters

`info` SerializationInfo

The SerializationInfo to populate

`context` StreamingContext

The destination for this serialization

OnAfterDeserialize()

Reconstructs the internal hash set from the serialized list format. Called automatically by Unity after deserialization occurs.

```
public void OnAfterDeserialize()
```

OnBeforeSerialize()

Converts the internal hash set to a serialized list format for Unity serialization. Called automatically by Unity before serialization occurs.

```
public void OnBeforeSerialize()
```

Remove(T)

Removes the specified element from the hash set.

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

Parameters

`item` T

The element to remove

Returns

bool

True if the element was successfully removed; otherwise, false

RemoveWhere(Predicate<T>)

Removes all elements that match the conditions defined by the specified predicate.

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

Parameters

`match` Predicate<T>

The predicate that defines the conditions of the elements to remove

Returns

int

The number of elements that were removed