

Namespace ElectricDrill.AstraRpgHealth

Classes

[EntityHealth](#)

Manages the health, damage, and healing mechanics for an entity. Handles damage calculation, health regeneration, barriers (temporary HP), and death events.

Enums

[EntityHealth.HpBehaviourOnMaxHpDecrease](#)

[EntityHealth.HpBehaviourOnMaxHpIncrease](#)

Class EntityHealth

Namespace: [ElectricDrill.AstraRpgHealth](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Manages the health, damage, and healing mechanics for an entity. Handles damage calculation, health regeneration, barriers (temporary HP), and death events.

```
[RequireComponent(typeof(EntityCore))]  
public class EntityHealth : MonoBehaviour, IDamageable, IHealable, IResurrectable
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← EntityHealth

Implements

[IDamageable](#), [IHealable](#), [IResurrectable](#)

Properties

Barrier

Gets the current barrier points (temporary HP) of the entity.

```
public long Barrier { get; }
```

Property Value

long

Class

Gets or sets the EntityClass component associated with this entity.

```
public EntityClass Class { get; set; }
```

Property Value

EntityClass

EntityAttributes

Gets or sets the EntityAttributes component associated with this entity.

```
public EntityAttributes EntityAttributes { get; set; }
```

Property Value

EntityAttributes

EntityCore

Gets or sets the EntityCore component associated with this entity.

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

Property Value

EntityCore

EntityStats

Gets or sets the EntityStats component associated with this entity.

```
public EntityStats EntityStats { get; set; }
```

Property Value

EntityStats

HealthCanBeNegative

Gets or sets whether the entity's health can go below zero. When set to true, a death threshold must be defined.

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

Property Value

bool

Hp

Gets the current health points of the entity.

```
public long Hp { get; }
```

Property Value

long

IsImmune

Gets or sets whether the entity is immune to all damage.

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

Property Value

bool

MaxHp

Gets the maximum health points of the entity.

```
public long MaxHp { get; }
```

Property Value

long

OverrideOnDeathStrategy

Gets or sets the override strategy for handling entity death. This strategy takes precedence over the default strategy defined in the config. If null, the default strategy from the config will be used.

```
public OnDeathStrategy OverrideOnDeathStrategy { get; set; }
```

Property Value

[OnDeathStrategy](#)

Methods

AddMaxHpFlatModifier(long, HpBehaviourOnMaxHpIncrease, HpBehaviourOnMaxHpDecrease)

Adds a flat modifier to the entity's maximum health.

```
public void AddMaxHpFlatModifier(long amount, EntityHealth.HpBehaviourOnMaxHpIncrease  
onMaxHpIncrease = HpBehaviourOnMaxHpIncrease.None, EntityHealth.HpBehaviourOnMaxHpDecrease  
onMaxHpDecrease = HpBehaviourOnMaxHpDecrease.RemoveHealthUpToMaxHp)
```

Parameters

amount long

The amount to add to the flat modifier. Can be negative.

onMaxHpIncrease [EntityHealth.HpBehaviourOnMaxHpIncrease](#)

Behavior to apply when max HP increases.

onMaxHpDecrease [EntityHealth.HpBehaviourOnMaxHpDecrease](#)

Behavior to apply when max HP decreases.

AddMaxHpPercentageModifier(Percentage, HpBehaviourOnMaxHpIncrease, HpBehaviourOnMaxHpDecrease)

Adds a percentage modifier to the entity's maximum health.

```
public void AddMaxHpPercentageModifier(Percentage amount,
EntityHealth.HpBehaviourOnMaxHpIncrease onMaxHpIncrease = HpBehaviourOnMaxHpIncrease.None,
EntityHealth.HpBehaviourOnMaxHpDecrease onMaxHpDecrease =
HpBehaviourOnMaxHpDecrease.RemoveHealthUpToMaxHp)
```

Parameters

amount Percentage

The percentage to add to the modifier.

onMaxHpIncrease [EntityHealth.HpBehaviourOnMaxHpIncrease](#)

Behavior to apply when max HP increases.

onMaxHpDecrease [EntityHealth.HpBehaviourOnMaxHpDecrease](#)

Behavior to apply when max HP decreases.

AddMaxHpScaling(StatsScalingComponent, HpBehaviourOnMaxHpIncrease, HpBehaviourOnMaxHpDecrease)

Adds a stats-based scaling component that dynamically affects maximum health. Subscribes to stat changes if this is the first scaling added.

```
public void AddMaxHpScaling(StatsScalingComponent scaling,
EntityHealth.HpBehaviourOnMaxHpIncrease onMaxHpIncrease = HpBehaviourOnMaxHpIncrease.None,
EntityHealth.HpBehaviourOnMaxHpDecrease onMaxHpDecrease =
HpBehaviourOnMaxHpDecrease.RemoveHealthUpToMaxHp)
```

Parameters

scaling StatsScalingComponent

The scaling component to add.

`onMaxHpIncrease` [EntityHealth.HpBehaviourOnMaxHpIncrease](#)

Behavior to apply when max HP increases because of the scaling.

`onMaxHpDecrease` [EntityHealth.HpBehaviourOnMaxHpDecrease](#)

Behavior to apply when max HP decreases because of the scaling.

`CalculateReducedDmg(long, long, DefenseReductionFn, long, Stat, DamageReductionFn, bool)`

Calculates reduced damage based on piercing and defensive stats using reduction functions.

```
public static long CalculateReducedDmg(long amount, long piercingStatValue,
DefenseReductionFn defenseReductionFn, long defensiveStatValue, Stat defensiveStat,
DamageReductionFn damageReductionFn, bool applyClamp = true)
```

Parameters

`amount` long

The base damage amount.

`piercingStatValue` long

The attacker's piercing stat value.

`defenseReductionFn` [DefenseReductionFn](#)

Function to calculate defense reduction (can be null).

`defensiveStatValue` long

The defender's defensive stat value.

`defensiveStat` Stat

The defensive stat object (can be null if defenseReductionFn is null).

`damageReductionFn` [DamageReductionFn](#)

Function to calculate damage reduction (can be null).

applyClamp bool

Whether to apply stat clamping in defense reduction. Default true for gameplay; false for analysis.

Returns

long

The reduced damage amount after applying all reduction calculations.

ClearMaxHpScalings(HpBehaviourOnMaxHpIncrease, HpBehaviourOnMaxHpDecrease)

Removes all stats-based scaling components from maximum health calculations.

```
public void ClearMaxHpScalings(EntityHealth.HpBehaviourOnMaxHpIncrease onMaxHpIncrease =  
    HpBehaviourOnMaxHpIncrease.None, EntityHealth.HpBehaviourOnMaxHpDecrease onMaxHpDecrease =  
    HpBehaviourOnMaxHpDecrease.RemoveHealthUpToMaxHp)
```

Parameters

onMaxHpIncrease [EntityHealth.HpBehaviourOnMaxHpIncrease](#)

onMaxHpDecrease [EntityHealth.HpBehaviourOnMaxHpDecrease](#)

Heal(PreHealInfo)

Heals the entity by the specified amount, applying critical multipliers and heal modifiers. The actual health gained may be less than the heal amount if the entity is at or near maximum health. Throws DeadEntityException if the entity is dead.

```
public ReceivedHealInfo Heal(PreHealInfo info)
```

Parameters

info [PreHealInfo](#)

The pre-heal information including amount, source, and healer.

Returns

[ReceivedHealInfo](#)

Information about the heal received including the actual health gained.

Exceptions

[DeadEntityException](#)

Thrown when attempting to heal a dead entity.

IsAlive()

Checks if the entity's current health is above the death threshold.

```
public bool IsAlive()
```

Returns

bool

true if current health is > death threshold, false otherwise

IsDead()

Checks if the entity's current health is at or below the death threshold. Returns the cached death state for performance.

```
public bool IsDead()
```

Returns

bool

true if current health is <= death threshold, false otherwise

ManualHealthRegenerationTick()

Method to be called to trigger manual health regeneration (e.g., if you are using a turn-based system, call this method at the start/end of each turn).

```
public void ManualHealthRegenerationTick()
```

RemoveMaxHpScaling(StatsScalingComponent, HpBehaviourOnMaxHpIncrease, HpBehaviourOnMaxHpDecrease)

Removes a previously added stats-based scaling component from maximum health calculations. Unsubscribes from stat changes if no scalings remain.

```
public void RemoveMaxHpScaling(StatsScalingComponent scaling,  
EntityHealth.HpBehaviourOnMaxHpIncrease onMaxHpIncrease = HpBehaviourOnMaxHpIncrease.None,  
EntityHealth.HpBehaviourOnMaxHpDecrease onMaxHpDecrease =  
HpBehaviourOnMaxHpDecrease.RemoveHealthUpToMaxHp)
```

Parameters

scaling StatsScalingComponent

The scaling component to remove.

onMaxHpIncrease [EntityHealth.HpBehaviourOnMaxHpIncrease](#)

Behavior to apply when max HP increases because of the scaling removal.

onMaxHpDecrease [EntityHealth.HpBehaviourOnMaxHpDecrease](#)

Behavior to apply when max HP decreases because of the scaling removal.

Resurrect(Percentage)

Resurrects the entity with a percentage of maximum health using the default resurrection source from config. Throws InvalidOperationException if the entity is already alive.

```
public void Resurrect(Percentage withHpPercent)
```

Parameters

withHpPercent Percentage

The percentage of maximum health to restore on resurrection.

Exceptions

InvalidOperationException

Thrown when attempting to resurrect an entity that is already alive.

Resurrect(Percentage, HealSource)

Resurrects the entity with a percentage of maximum health. Throws InvalidOperationException if the entity is already alive.

```
public void Resurrect(Percentage withHpPercent, HealSource healSource)
```

Parameters

withHpPercent Percentage

The percentage of maximum health to restore on resurrection.

healSource [HealSource](#)

The heal source associated with the resurrection.

Exceptions

InvalidOperationException

Thrown when attempting to resurrect an entity that is already alive.

Resurrect(long)

Resurrects the entity with a specific amount of health using the default resurrection source from config. Throws InvalidOperationException if the entity is already alive.

```
public void Resurrect(long withHp)
```

Parameters

withHp long

The amount of health to restore on resurrection.

Exceptions

InvalidOperationException

Thrown when attempting to resurrect an entity that is already alive.

Resurrect(long, HealSource)

Resurrects the entity with a specific amount of health. Throws InvalidOperationException if the entity is already alive.

```
public void Resurrect(long withHp, HealSource healSource)
```

Parameters

withHp long

The amount of health to restore on resurrection.

healSource [HealSource](#)

The heal source associated with the resurrection.

Exceptions

InvalidOperationException

Thrown when attempting to resurrect an entity that is already alive.

SetHpToMax()

Sets the entity's current health to its maximum health value. Throws `DeadEntityException` if the entity is dead.

```
public void SetHpToMax()
```

Exceptions

[DeadEntityException](#)

Thrown when attempting to set HP to max on a dead entity.

SetupMaxHp(HpBehaviourOnMaxHpIncrease, HpBehaviourOnMaxHpDecrease)

Recalculates and updates the entity's maximum health based on base value, modifiers, and scaling. Raises the max health changed event if the value differs from the previous maximum. If the entity is dead, only recalculates max HP and raises events without adjusting current HP.

Behavior to apply when max HP increases. Behavior to apply when max HP decreases.

```
public void SetupMaxHp(EntityHealth.HpBehaviourOnMaxHpIncrease onMaxHpIncrease,  
EntityHealth.HpBehaviourOnMaxHpDecrease onMaxHpDecrease =  
HpBehaviourOnMaxHpDecrease.RemoveHealthUpToMaxHp)
```

Parameters

`onMaxHpIncrease` [EntityHealth.HpBehaviourOnMaxHpIncrease](#)

`onMaxHpDecrease` [EntityHealth.HpBehaviourOnMaxHpDecrease](#)

TakeDamage(PreDamageInfo)

Applies damage to the entity through the damage calculation pipeline. This is the primary method for dealing damage in the Astra RPG Health system.

Execution Order:

- 1. Death State Check:** If the entity is already dead, the damage is marked as prevented with `EntityDead` reason and returned immediately. Dead entities cannot take damage.

2. **Pre-Damage Event:** Raises the PreDamageGameEvent to notify listeners of incoming damage. This allows systems to react before any damage calculations occur.
3. **DamageInfo Creation:** Converts the PreDamageInfo into a DamageInfo object that will track the damage through the calculation pipeline.
4. **Immunity Check:** If the entity is immune (IsImmune is true), the damage is marked as prevented with EntityImmune reason. This happens before the pipeline to short-circuit processing.
5. **Damage Calculation Pipeline:** If damage is not already prevented, the configured DamageCalculationStrategy processes the damage. The pipeline can apply:
 - o Damage reduction based on defensive stats
 - o Barrier consumption (temporary HP absorption)
 - o Critical hit calculations
 - o General damage and/or damage type and damage source modifiers
 - o Custom pipeline stages defined in the strategy
 The pipeline can also mark damage as prevented for various reasons.
6. **Prevention Check:** If the damage was prevented (either by immunity or during the pipeline):
 - o Creates a DamageResolution.Prevented result with all prevention reasons
 - o Raises the DamageResolutionGameEvent
 - o Returns immediately without affecting health
7. **Health Reduction:** If damage was not prevented, the final calculated damage amount is subtracted from the entity's current health via RemoveHealth(). This may trigger the LostHealthGameEvent.
8. **Damage Resolution Event:** Raises the DamageResolutionGameEvent with the applied damage information. This allows systems to react to successful damage (e.g., lifesteal, on-hit effects).
9. **Death Check:** If the entity's health is now at or below the death threshold:
 - o Raises the EntityDiedGameEvent with this EntityHealth and the DamageResolution
 - o Executes the OnDeathStrategy (either the entity's override or the config default)
 - o The death strategy handles the actual death behavior (e.g., destroy, return to the object pool, disable, ragdoll)
10. **Return:** Returns a DamageResolution.Applied result containing all damage information for the caller to process if needed.

Strategy Selection:

The method uses the first available DamageCalculationStrategy in this priority order:

1. OverrideDamageCalculationStrategy (if set on this EntityHealth)
2. CustomDamageCalculationStrategy (if set on this EntityHealth)
3. DefaultDamageCalculationStrategy (from the AstraRpgHealthConfig)

```
public DamageResolution TakeDamage(PreDamageInfo preDamage)
```

Parameters

`preDamage` [`PreDamageInfo`](#)

The pre-damage information including base amount, damage type, dealer entity, and additional context. This is used by the damage calculation pipeline to compute the final damage to apply.

Returns

[`DamageResolution`](#)

A `DamageResolution` indicating whether damage was applied or prevented. If prevented, includes the reasons for prevention. If applied, includes the final damage amounts and calculation details.

Enum EntityHealth.HpBehaviourOnMaxHpDecrease

Namespace: [ElectricDrill.AstraRpgHealth](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
public enum EntityHealth.HpBehaviourOnMaxHpDecrease
```

Fields

RemoveHealthAnyway = 3

RemoveHealthUpTo1 = 2

RemoveHealthUpToMaxHp = 0

RemoveHealthUpToMaxHpAndConvertRemovedToBarrier = 1

Enum EntityHealth.HpBehaviourOnMaxHpIncrease

Namespace: [ElectricDrill.AstraRpgHealth](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
public enum EntityHealth.HpBehaviourOnMaxHpIncrease
```

Fields

AddHealthAndConvertExcessToBarrier = 2

AddHealthUpToMaxHp = 1

None = 0

Namespace ElectricDrill.AstraRpgHealth.Config

Classes

[AstraRpgHealthConfig](#)

[AstraRpgHealthConfigProvider](#)

[AstraRpgHealthGlobalSettings](#)

Interfaces

[IAstraRpgHealthConfig](#)

Interface for health system configuration. This allows for dependency injection and easier testing.

Class AstraRpgHealthConfig

Namespace: [ElectricDrill.AstraRpgHealth.Config](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
public class AstraRpgHealthConfig : ScriptableObject, IAstraRpgHealthConfig
```

Inheritance

object ← Object ← ScriptableObject ← AstraRpgHealthConfig

Implements

[IAstraRpgHealthConfig](#)

Fields

damageSourceModifications

```
public SerializableDictionary<DamageSource, Stat> damageSourceModifications
```

Field Value

SerializableDictionary<[DamageSource](#), Stat>

damageTypeModifications

```
public SerializableDictionary<DamageType, Stat> damageTypeModifications
```

Field Value

SerializableDictionary<[DamageType](#), Stat>

Properties

DamageSourceModifications

```
public SerializableDictionary<DamageSource, Stat> DamageSourceModifications { get; set; }
```

Property Value

SerializableDictionary<[DamageSource](#), Stat>

DamageTypeModifications

```
public SerializableDictionary<DamageType, Stat> DamageTypeModifications { get; set; }
```

Property Value

SerializableDictionary<[DamageType](#), Stat>

DefaultDamageCalculationCalculationStrategy

```
public DamageCalculationStrategy DefaultDamageCalculationCalculationStrategy { get; set; }
```

Property Value

[DamageCalculationStrategy](#)

DefaultOnDeathStrategy

```
public OnDeathStrategy DefaultOnDeathStrategy { get; set; }
```

Property Value

[OnDeathStrategy](#)

DefaultOnResurrectionStrategy

```
public OnResurrectionStrategy DefaultOnResurrectionStrategy { get; set; }
```

Property Value

[OnResurrectionStrategy](#)

DefaultResurrectionSource

```
public HealSource DefaultResurrectionSource { get; set; }
```

Property Value

[HealSource](#)

GenericDamageModificationStat

```
public Stat GenericDamageModificationStat { get; set; }
```

Property Value

Stat

HealAmountModifierStat

```
public Stat HealAmountModifierStat { get; set; }
```

Property Value

Stat

HealthAttributesScaling

```
public AttributesScalingComponent HealthAttributesScaling { get; set; }
```

Property Value

AttributesScalingComponent

LifestealConfig

```
public LifestealConfig LifestealConfig { get; set; }
```

Property Value

[LifestealConfig](#)

ManualHealthRegenerationStat

```
public Stat ManualHealthRegenerationStat { get; set; }
```

Property Value

Stat

PassiveHealthRegenerationInterval

```
public float PassiveHealthRegenerationInterval { get; set; }
```

Property Value

float

PassiveHealthRegenerationSource

```
public HealSource PassiveHealthRegenerationSource { get; set; }
```

Property Value

[HealSource](#)

PassiveHealthRegenerationStat

```
public Stat PassiveHealthRegenerationStat { get; set; }
```

Property Value

Stat

Class AstraRpgHealthConfigProvider

Namespace: [ElectricDrill.AstraRpgHealth.Config](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
public static class AstraRpgHealthConfigProvider
```

Inheritance

object ← AstraRpgHealthConfigProvider

Properties

Instance

```
public static IAstraRpgHealthConfig Instance { get; set; }
```

Property Value

[IAstraRpgHealthConfig](#)

Methods

Reset()

```
public static void Reset()
```

WarmUp()

```
public static void WarmUp()
```

Class AstraRpgHealthGlobalSettings

Namespace: [ElectricDrill.AstraRpgHealth.Config](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
public class AstraRpgHealthGlobalSettings : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← AstraRpgHealthGlobalSettings

Properties

ActiveConfig

```
public AstraRpgHealthConfig ActiveConfig { get; set; }
```

Property Value

[AstraRpgHealthConfig](#)

Interface IAstraRpgHealthConfig

Namespace: [ElectricDrill.AstraRpgHealth.Config](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Interface for health system configuration. This allows for dependency injection and easier testing.

```
public interface IAstraRpgHealthConfig
```

Properties

DamageSourceModifications

```
SerializableDictionary<DamageSource, Stat> DamageSourceModifications { get; set; }
```

Property Value

```
SerializableDictionary<DamageSource, Stat>
```

DamageTypeModifications

```
SerializableDictionary<DamageType, Stat> DamageTypeModifications { get; set; }
```

Property Value

```
SerializableDictionary<DamageType, Stat>
```

DefaultDamageCalculationCalculationStrategy

```
DamageCalculationStrategy DefaultDamageCalculationCalculationStrategy { get; set; }
```

Property Value

[DamageCalculationStrategy](#)

DefaultOnDeathStrategy

```
OnDeathStrategy DefaultOnDeathStrategy { get; set; }
```

Property Value

[OnDeathStrategy](#)

DefaultOnResurrectionStrategy

```
OnResurrectionStrategy DefaultOnResurrectionStrategy { get; set; }
```

Property Value

[OnResurrectionStrategy](#)

DefaultResurrectionSource

```
HealSource DefaultResurrectionSource { get; set; }
```

Property Value

[HealSource](#)

GenericDamageModificationStat

```
Stat GenericDamageModificationStat { get; set; }
```

Property Value

Stat

HealAmountModifierStat

```
Stat HealAmountModifierStat { get; set; }
```

Property Value

Stat

HealthAttributesScaling

```
AttributesScalingComponent HealthAttributesScaling { get; set; }
```

Property Value

AttributesScalingComponent

LifestealConfig

```
LifestealConfig LifestealConfig { get; set; }
```

Property Value

[LifestealConfig](#)

ManualHealthRegenerationStat

```
Stat ManualHealthRegenerationStat { get; set; }
```

Property Value

Stat

PassiveHealthRegenerationInterval

```
float PassiveHealthRegenerationInterval { get; set; }
```

Property Value

float

PassiveHealthRegenerationSource

```
HealSource PassiveHealthRegenerationSource { get; set; }
```

Property Value

[HealSource](#)

PassiveHealthRegenerationStat

```
Stat PassiveHealthRegenerationStat { get; set; }
```

Property Value

Stat

Namespace ElectricDrill.AstraRpgHealth.

Damage

Classes

[DamageAmountInfo](#)

Holds numeric details related to an attempt to apply damage, allowing intermediate values to be recorded for each phase of the calculation pipeline (e.g. reductions, resistances, absorptions).

[DamagePreventionReasonExtensions](#)

Extension helpers for [DamagePreventionReason](#).

[DamageResolution](#)

The result returned by damage application attempts. Encapsulates whether the damage was applied or prevented, the reasons for prevention, and the concrete damage information that was ultimately applied (if any).

[DamageSource](#)

Defines a damage source asset that identifies how damage was produced (for example: a spell, trap, environmental hazard, system event, etc.). Create instances via Assets -> Create -> Astra RPG Health / Damage Source.

[DamageType](#)

Scriptable asset that describes a damage category (for example: Physical, Fire, Poison). A DamageType declares which stat reduces this damage, which reduction functions to use, whether a stat pierces its defensive stats and whether it ignores barriers. Create instances via Assets -> Create -> Astra RPG Health / DamageType.

[PreDamageInfo](#)

Immutable container representing damage information before it is applied. Use [Builder](#) to construct instances using the fluent step builder.

[PreDamageInfo.PreDamageInfoStepBuilder](#)

Concrete step builder implementing the fluent interfaces. After setting required fields, optional flags (critical, multiplier) can be configured before calling [Build\(\)](#).

Structs

[DamageAmountInfo.StepAmountRecord](#)

Immutable record representing the damage value before and after a single pipeline phase (identified by the phase type).

Interfaces

[IDamageable](#)

Implemented by entities that can receive damage. The [TakeDamage\(PreDamageInfo\)](#) method accepts a [PreDamageInfo](#) record describing a pending damage attempt and returns a [DamageResolution](#) describing whether the damage was applied or prevented and additional details.

[PreDamageInfo.DamageInfoAmount](#)

Fluent builder step: specify the damage amount.

[PreDamageInfo.DamageInfoDealer](#)

Fluent builder step: specify the dealer entity.

[PreDamageInfo.DamageInfoSource](#)

Fluent builder step: specify the damage source.

[PreDamageInfo.DamageInfoTarget](#)

Fluent builder step: specify the target entity.

[PreDamageInfo.DamageInfoType](#)

Fluent builder step: specify the damage type.

Enums

[DamageOutcome](#)

The overall result of attempting to apply damage to a target. Use this in [DamageResolution](#) to indicate whether any damage was applied or if the attempt was prevented.

[DamagePreventionReason](#)

Flags that explain why a damage attempt was prevented. Multiple reasons may be combined. These flags are used in [Reasons](#) when an attempt is prevented.

Class DamageAmountInfo

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Holds numeric details related to an attempt to apply damage, allowing intermediate values to be recorded for each phase of the calculation pipeline (e.g. reductions, resistances, absorptions).

```
public class DamageAmountInfo
```

Inheritance

object ← DamageAmountInfo

Remarks

This class is mutable for [Current](#) and keeps a list of [DamageAmountInfo.StepAmountRecord](#) entries that describe the value before and after each step. It is used to trace how the initial raw value is transformed into the final damage applied.

Constructors

DamageAmountInfo(long)

Creates a new instance of [DamageAmountInfo](#) with the provided initial value.

```
public DamageAmountInfo(long initialAmount)
```

Parameters

initialAmount long

Raw starting damage value.

Properties

Current

The current damage value during calculation; updated by the various phases. The value is clamped to a minimum of 0.

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

Property Value

long

InitialAmount

The raw initial value provided to the damage calculation pipeline.

```
public long InitialAmount { get; }
```

Property Value

long

Records

Read-only list of the recorded phase entries in order.

```
public IReadOnlyList<DamageAmountInfo.StepAmountRecord> Records { get; }
```

Property Value

IReadOnlyList<[DamageAmountInfo.StepAmountRecord](#)>

Methods

FromRaw(long)

Convenience factory that builds a [DamageAmountInfo](#) from a raw value; equivalent to using the public constructor.

```
public static DamageAmountInfo FromRaw(long raw)
```

Parameters

raw long

Raw damage value.

Returns

[DamageAmountInfo](#)

A new [DamageAmountInfo](#) initialized to **raw**.

GetStepAmount(Type)

Retrieves the record associated with a given phase identified by its type.

```
public DamageAmountInfo.StepAmountRecord GetStepAmount(Type stepType)
```

Parameters

stepType Type

The phase type to look up.

Returns

[DamageAmountInfo.StepAmountRecord](#)

The corresponding [DamageAmountInfo.StepAmountRecord](#) or the default if not found.

Remarks

If no matching record exists the default value of [DamageAmountInfo.StepAmountRecord](#) is returned (with [StepType](#) possibly [null](#)).

Struct DamageAmountInfo.StepAmountRecord

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Immutable record representing the damage value before and after a single pipeline phase (identified by the phase type).

```
public readonly struct DamageAmountInfo.StepAmountRecord
```

Constructors

StepAmountRecord(Type, long, long)

Creates a new record for a phase with the associated before/after values.

```
public StepAmountRecord(Type stepType, long pre, long post)
```

Parameters

stepType Type

Phase type.

pre long

Value before the phase.

post long

Value after the phase.

Properties

Post

Damage value immediately after the phase executes.

```
public long Post { get; }
```

Property Value

long

Pre

Damage value immediately before the phase executes.

```
public long Pre { get; }
```

Property Value

long

StepType

The type that represents the phase (for example, a reduction class). May be `null` for uninitialized or default records.

```
public Type StepType { get; }
```

Property Value

Type

Methods

ToString()

Compact textual representation of the record, useful for logging.

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

Returns

string

String in the format "{TypeName}: {Pre} -> {Post}".

Enum DamageOutcome

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The overall result of attempting to apply damage to a target. Use this in [DamageResolution](#) to indicate whether any damage was applied or if the attempt was prevented.

```
public enum DamageOutcome
```

Fields

Applied = 0

The damage was applied to the target's health/defenses. See [FinalDamageInfo](#) for details.

Prevented = 1

The damage attempt was prevented. See [Reasons](#) for prevention reasons.

Enum DamagePreventionReason

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Flags that explain why a damage attempt was prevented. Multiple reasons may be combined. These flags are used in [Reasons](#) when an attempt is prevented.

```
[Flags]  
public enum DamagePreventionReason
```

Extension Methods

[DamagePreventionReasonExtensions.IsTerminal\(DamagePreventionReason\)](#)

Fields

AllDamageImmune = 2

The target is immune to all damage because of its generic damage reduction stat value.

BarrierAbsorbed = 16

The target's barrier fully absorbed the damage.

DamageSourceImmune = 8

The target is immune to the specific [DamageSource](#) that originated the damage.

DamageTypeImmune = 4

The target is immune to this specific [DamageType](#).

DefenseAbsorbed = 32

The target's stat defense for the damage type fully absorbed the damage.

EntityDead = 512

The target was already dead when the damage would have been applied.

EntityImmune = 1

The target entity is immune to all damage (global immunity).

None = 0

No prevention; damage can proceed.

PipelineReducedToZero = 256

After reductions, a pipeline step reduced the damage to zero and nothing was applied.

PrePhaseIgnored = 64

The pre-processing phase explicitly ignored the damage (for example by a passive ability that intercepted the PreDmgGameEvent and instructed the PreDamageInfo to ignore it).

PrePhaseZeroAmount = 128

The damage amount was zero in the pre-phase and therefore not applied.

Class DamagePreventionReasonExtensions

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Extension helpers for [DamagePreventionReason](#).

```
public static class DamagePreventionReasonExtensions
```

Inheritance

object ← DamagePreventionReasonExtensions

Methods

IsTerminal(DamagePreventionReason)

Returns true when the provided set of reasons indicates a terminal prevention state (i.e. any prevention reason is present). False when [None](#).

```
public static bool IsTerminal(this DamagePreventionReason reasons)
```

Parameters

reasons [DamagePreventionReason](#)

Returns

bool

Class DamageResolution

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The result returned by damage application attempts. Encapsulates whether the damage was applied or prevented, the reasons for prevention, and the concrete damage information that was ultimately applied (if any).

```
public class DamageResolution
```

Inheritance

object ← DamageResolution

Properties

FinalDamageInfo

The final computed damage information that was applied to the target. This is non-null only when [Outcome](#) is [Applied](#). The contained [DamageInfo](#) includes applied amount, barrier/health split, termination step type and any additional diagnostics produced by the pipeline.

```
public DamageInfo FinalDamageInfo { get; }
```

Property Value

[DamageInfo](#)

Outcome

The high-level outcome of the attempt: either [Applied](#) or [Prevented](#).

```
public DamageOutcome Outcome { get; }
```

Property Value

[DamageOutcome](#)

PreInfo

The original [PreDamageInfo](#) that triggered this resolution. Useful for logging and diagnostic correlation.

```
public PreDamageInfo PreInfo { get; }
```

Property Value

[PreDamageInfo](#)

Reasons

When [Outcome](#) is [Prevented](#), this contains the combined [DamagePreventionReason](#) flags explaining why. It will be [None](#) when [Outcome](#) is [Applied](#).

```
public DamagePreventionReason Reasons { get; }
```

Property Value

[DamagePreventionReason](#)

TerminationStepType

If the pipeline execution terminated early, this will contain the System.Type of the pipeline termination step that decided to stop processing. May be null when the pipeline completed normally.

```
public Type TerminationStepType { get; }
```

Property Value

Type

Methods

Applied(DamageInfo, PreDamageInfo)

Create a [DamageResolution](#) representing an applied damage attempt. The `info` parameter contains the concrete damage values applied.

```
public static DamageResolution Applied(DamageInfo info, PreDamageInfo pre)
```

Parameters

`info` [DamageInfo](#)

Final damage info that was applied.

`pre` [PreDamageInfo](#)

Original pre-damage request.

Returns

[DamageResolution](#)

A configured [DamageResolution](#) with Outcome == Applied.

Prevented(DamagePreventionReason, PreDamageInfo, DamageInfo)

Create a [DamageResolution](#) representing a prevented damage attempt. `reasons` should explain why the attempt didn't apply. Optionally the `info` parameter may contain a partial [DamageInfo](#) produced before prevention.

```
public static DamageResolution Prevented(DamagePreventionReason reasons, PreDamageInfo pre,  
DamageInfo info = null)
```

Parameters

`reasons` [DamagePreventionReason](#)

Flags explaining the prevention.

`pre` [PreDamageInfo](#)

Original pre-damage request.

info [DamageInfo](#)

Optional partial damage info produced before prevention.

Returns

[DamageResolution](#)

A configured [DamageResolution](#) with Outcome == Prevented.

Class DamageSource

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Defines a damage source asset that identifies how damage was produced (for example: a spell, trap, environmental hazard, system event, etc.). Create instances via Assets -> Create -> Astra RPG Health / Damage Source.

```
[CreateAssetMenu(fileName = "New Damage Source", menuName = "Astra RPG  
Health/Damage Source")]  
public class DamageSource : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← DamageSource

Methods

ToString()

Returns the asset name of this damage source.

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

Returns

string

Class DamageType

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Scriptable asset that describes a damage category (for example: Physical, Fire, Poison). A DamageType declares which stat reduces this damage, which reduction functions to use, whether a stat pierces its defensive stats and whether it ignores barriers. Create instances via Assets -> Create -> Astra RPG Health / DamageType.

```
[CreateAssetMenu(fileName = "New Damage Type", menuName = "Astra RPG Health/DamageType")]
public class DamageType : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← DamageType

Properties

DamageReductionFn

The reduction function used to compute how the damage amount is modified by the reducing stat (see [ReducedBy](#)). This may implement flat, percentage, logarithmic, or custom reduction behaviour.

```
public DamageReductionFn DamageReductionFn { get; }
```

Property Value

[DamageReductionFn](#)

DefenseReductionFn

The reduction function used to compute how the piercing stat modifies the defensive stat that reduces damage for this [DamageType](#). This may implement flat, percentage, logarithmic, or custom reduction behaviour.

```
public DefenseReductionFn DefenseReductionFn { get; }
```

Property Value

[DefenseReductionFn](#)

DefensiveStatPiercedBy

The stat that pierces the defensive stat for this damage type (for example an "Armor Penetration" stat pierces the "Armor" stat). May be null.

```
public Stat DefensiveStatPiercedBy { get; }
```

Property Value

Stat

IgnoresBarrier

If true, this damage type bypasses barrier mechanic and applies directly to underlying health. Typical use: certain elemental or pure damage types.

```
public bool IgnoresBarrier { get; protected set; }
```

Property Value

bool

ReducedBy

The stat that reduces the raw damage amount for this [DamageType](#). For example Physical damage may be reduced by an "Armor" stat.

```
public Stat ReducedBy { get; }
```

Property Value

Stat

Methods

ToString()

Returns the asset name of this damage type.

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

Returns

string

Interface IDamageable

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Implemented by entities that can receive damage. The [TakeDamage\(PreDamageInfo\)](#) method accepts a [PreDamageInfo](#) record describing a pending damage attempt and returns a [DamageResolution](#) describing whether the damage was applied or prevented and additional details.

```
public interface IDamageable
```

Methods

TakeDamage(PreDamageInfo)

Process an incoming damage request and return a resolution object.

Implementations should run the damage processing pipeline (pre-phase, reduction, defensive checks, barriers, health application) or delegate to the centralized pipeline host used in the project. The provided `preDamage` describes the intent (amount, type, source, critical flags, target/dealer).

The returned [DamageResolution](#) must reflect the final outcome:

- If damage was applied, an instance created with [Applied\(DamageInfo, PreDamageInfo\)](#) is expected and [FinalDamageInfo](#) will contain the concrete damage values that were actually applied to the target.
- If the damage was prevented (for example due to immunity, zero amount after reductions, or an explicit ignore), an instance created with [Prevented\(DamagePreventionReason, PreDamageInfo, DamageInfo\)](#) is expected.

Implementations SHOULD NOT throw for normal prevention cases; use the prevention reasons and the returned resolution to communicate why the damage didn't apply.

```
DamageResolution TakeDamage(PreDamageInfo preDamage)
```

Parameters

`preDamage` [PreDamageInfo](#)

Immutable description of the pending damage attempt.

Returns

[DamageResolution](#)

A [DamageResolution](#) describing the final outcome and details.

Class PreDamageInfo

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Immutable container representing damage information before it is applied. Use [Builder](#) to construct instances using the fluent step builder.

```
public class PreDamageInfo
```

Inheritance

object ← PreDamageInfo

Properties

Amount

The raw damage amount (before modifiers).

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

Property Value

long

Builder

Entry point for the fluent builder. Example:

```
var pre = PreDamageInfo.Builder
    .WithAmount(10)
    .WithType(DamageType.Physical)
    .WithSource(DamageSource.Weapon)
    .WithTarget(targetEntity)
    .WithDealer(dealerEntity)
    .WithIsCritical(false)
    .Build();
```

```
public static PreDamageInfo.DamageInfoAmount Builder { get; }
```

Property Value

[PreDamageInfo.DamageInfoAmount](#)

CriticalMultiplier

Multiplier applied when this is critical damage. A value of 1.0 means no change. Values <= 0 are normalized to 1.0 by the constructor.

```
public double CriticalMultiplier { get; set; }
```

Property Value

double

DamageSource

The source or reason for the damage (for example an ability or environmental effect).

```
public DamageSource DamageSource { get; }
```

Property Value

[DamageSource](#)

Dealer

The entity that deals the damage (may be null for environmental damage).

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

Property Value

Ignore

If true, the damage will be ignored and not applied to the target. Use this in cases where the damage should be negated (e.g., passive abilities).

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

Property Value

bool

IsCritical

Whether this damage instance was flagged as a critical hit.

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

Property Value

bool

Target

The entity that will receive the damage.

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

Property Value

EntityCore

Type

The type/category of the damage (e.g. physical, magical).

```
public DamageType Type { get; set; }
```

Property Value

[DamageType](#)

Interface PreDamageInfo.DamageInfoAmount

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Fluent builder step: specify the damage amount.

```
public interface PreDamageInfo.DamageInfoAmount
```

Methods

WithAmount(long)

Set the damage amount and advance to the next step.

```
PreDamageInfo.DamageInfoType WithAmount(long amount)
```

Parameters

amount long

Returns

[PreDamageInfo.DamageInfoType](#)

Interface PreDamageInfo.DamageInfoDealer

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Fluent builder step: specify the dealer entity.

```
public interface PreDamageInfo.DamageInfoDealer
```

Methods

WithDealer(EntityCore)

Set the damage dealer and obtain the concrete builder for optional extra flags.

```
PreDamageInfo.PreDamageInfoStepBuilder WithDealer(EntityCore dealer)
```

Parameters

dealer EntityCore

Returns

[PreDamageInfo.PreDamageInfoStepBuilder](#)

Interface PreDamageInfo.DamageInfoSource

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Fluent builder step: specify the damage source.

```
public interface PreDamageInfo.DamageInfoSource
```

Methods

WithSource(DamageSource)

Set the damage source and advance to the next step.

```
PreDamageInfo.DamageInfoTarget WithSource(DamageSource damageSource)
```

Parameters

damageSource [DamageSource](#)

Returns

[PreDamageInfo.DamageInfoTarget](#)

Interface PreDamageInfo.DamageInfoTarget

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Fluent builder step: specify the target entity.

```
public interface PreDamageInfo.DamageInfoTarget
```

Methods

WithTarget(EntityCore)

Set the damage target and advance to the next step.

```
PreDamageInfo.DamageInfoDealer WithTarget(EntityCore target)
```

Parameters

target EntityCore

Returns

[PreDamageInfo.DamageInfoDealer](#)

Interface PreDamageInfo.DamageInfoType

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Fluent builder step: specify the damage type.

```
public interface PreDamageInfo.DamageInfoType
```

Methods

WithType(DamageType)

Set the damage type and advance to the next step.

```
PreDamageInfo.DamageInfoSource WithType(DamageType type)
```

Parameters

type [DamageType](#)

Returns

[PreDamageInfo.DamageInfoSource](#)

Class

PreDamageInfo.PreDamageInfoStepBuilder

Namespace: [ElectricDrill.AstraRpgHealth.Damage](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Concrete step builder implementing the fluent interfaces. After setting required fields, optional flags (critical, multiplier) can be configured before calling [Build\(\)](#).

```
public sealed class PreDamageInfo.PreDamageInfoStepBuilder : PreDamageInfo.DamageInfoAmount,  
    PreDamageInfo.DamageInfoType, PreDamageInfo.DamageInfoSource,  
    PreDamageInfo.DamageInfoTarget, PreDamageInfo.DamageInfoDealer
```

Inheritance

object ← PreDamageInfo.PreDamageInfoStepBuilder

Implements

[PreDamageInfo.DamageInfoAmount](#), [PreDamageInfo.DamageInfoType](#),
[PreDamageInfo.DamageInfoSource](#), [PreDamageInfo.DamageInfoTarget](#),
[PreDamageInfo.DamageInfoDealer](#)

Methods

Build()

Build the [PreDamageInfo](#) instance with the previously configured values.

```
public PreDamageInfo Build()
```

Returns

[PreDamageInfo](#)

WithAmount(long)

Set the damage amount.

```
public PreDamageInfo.DamageInfoType WithAmount(long amount)
```

Parameters

amount long

Returns

[PreDamageInfo.DamageInfoType](#)

WithCriticalMultiplier(double)

Set the critical damage multiplier. Values <= 0 will be normalized to 1.0 by the constructor.

```
public PreDamageInfo.PreDamageInfoStepBuilder WithCriticalMultiplier(double multiplier)
```

Parameters

multiplier double

Returns

[PreDamageInfo.PreDamageInfoStepBuilder](#)

WithDealer(EntityCore)

Set the dealer entity (the entity causing the damage).

```
public PreDamageInfo.PreDamageInfoStepBuilder WithDealer(EntityCore dealer)
```

Parameters

dealer EntityCore

Returns

[PreDamageInfo.PreDamageInfoStepBuilder](#)

WithIsCritical(bool)

Mark this damage as critical or not.

```
public PreDamageInfo.PreDamageInfoStepBuilder WithIsCritical(bool isCritical)
```

Parameters

`isCritical` bool

Returns

[PreDamageInfo.PreDamageInfoStepBuilder](#)

WithSource(DamageSource)

Set the damage source.

```
public PreDamageInfo.DamageInfoTarget WithSource(DamageSource damageSource)
```

Parameters

`damageSource` [DamageSource](#)

Returns

[PreDamageInfo.DamageInfoTarget](#)

WithTarget(EntityCore)

Set the target entity that will receive the damage.

```
public PreDamageInfo.DamageInfoDealer WithTarget(EntityCore target)
```

Parameters

target EntityCore

Returns

[PreDamageInfo.DamageInfoDealer](#)

WithType(DamageType)

Set the damage type.

```
public PreDamageInfo.DamageInfoSource WithType(DamageType type)
```

Parameters

type [DamageType](#)

Returns

[PreDamageInfo.DamageInfoSource](#)

Namespace ElectricDrill.AstraRpgHealth. Damage.CalculationPipeline

Classes

[ApplyBarrierStep](#)

Damage step that consumes target barrier (temporary shields). It doesn't remove health.

[ApplyCriticalMultiplierStep](#)

Damage step that applies critical hit multipliers when the current damage is flagged as critical.

[ApplyDefenseStep](#)

Damage step that applies defensive calculations based on the damage type's defensive and piercing stats and configured reduction functions.

[ApplyDmgModifiersStep](#)

Damage step that applies generic and configured damage source/type modifiers (for example weaknesses and resistances) to the current damage amount.

[DamageCalculationStrategy](#)

ScriptableObject that defines a configurable damage calculation pipeline. The pipeline is composed of ordered [DamageStep](#) instances executed sequentially.

[DamageInfo](#)

Container for damage pipeline state while a damage calculation is executed. Holds the amounts being transformed, the source/type metadata and accumulated prevention reasons.

[DamageStep](#)

Base class for a single step in the damage calculation pipeline. A [DamageStep](#) performs a focused transformation on a [DamageInfo](#) instance (for example applying defenses, barriers or modifiers).

Class ApplyBarrierStep

Namespace: [ElectricDrill.AstraRpgHealth.Damage.CalculationPipeline](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Damage step that consumes target barrier (temporary shields). It doesn't remove health.

```
public class ApplyBarrierStep : DamageStep
```

Inheritance

object ← [DamageStep](#) ← ApplyBarrierStep

Inherited Members

[DamageStep.Process\(DamageInfo\)](#)

Properties

DisplayName

Human readable name shown in pipeline editors.

```
public override string DisplayName { get; }
```

Property Value

string

Methods

ProcessStep(DamageInfo)

Applies the target's barrier to reduce the current damage amount. If the damage type ignores barrier, or the target has no barrier, the method returns the input unchanged. When barrier is consumed the target's barrier value is reduced accordingly. If the barrier completely absorbs the damage, [Pipeline ReducedToZero](#) is added.

```
public override DamageInfo ProcessStep(DamageInfo data)
```

Parameters

data [DamageInfo](#)

Current damage pipeline data.

Returns

[DamageInfo](#)

The same **data** instance updated with reduced amounts and any barrier consumption.

Class ApplyCriticalMultiplierStep

Namespace: [ElectricDrill.AstraRpgHealth.Damage.CalculationPipeline](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Damage step that applies critical hit multipliers when the current damage is flagged as critical.

```
public class ApplyCriticalMultiplierStep : DamageStep
```

Inheritance

object ← [DamageStep](#) ← ApplyCriticalMultiplierStep

Inherited Members

[DamageStep.Process\(DamageInfo\)](#)

Properties

DisplayName

Human readable name shown in pipeline editors.

```
public override string DisplayName { get; }
```

Property Value

string

Methods

ProcessStep(DamageInfo)

Multiplies the current damage by the critical multiplier from [DamageInfo](#) when the hit is marked critical. If the multiplier is 1 or invalid, no change occurs.

```
public override DamageInfo ProcessStep(DamageInfo data)
```

Parameters

data [DamagelInfo](#)

Current damage pipeline data.

Returns

[DamagelInfo](#)

The same **data** instance updated with the multiplied amount when applicable.

Class ApplyDefenseStep

Namespace: [ElectricDrill.AstraRpgHealth.Damage.CalculationPipeline](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Damage step that applies defensive calculations based on the damage type's defensive and piercing stats and configured reduction functions.

```
public class ApplyDefenseStep : DamageStep
```

Inheritance

object ← [DamageStep](#) ← ApplyDefenseStep

Inherited Members

[DamageStep.Process\(DamageInfo\)](#)

Properties

DisplayName

Human readable name shown in pipeline editors.

```
public override string DisplayName { get; }
```

Property Value

string

Methods

ProcessStep(DamageInfo)

Applies defense and damage reduction logic using the damage type's configuration. If the damage type is inconsistently configured the method logs a warning and skips the corresponding reduction stage.

```
public override DamageInfo ProcessStep(DamageInfo data)
```

Parameters

data [DamagelInfo](#)

Current damage pipeline data.

Returns

[DamagelInfo](#)

The same **data** instance updated with the reduced amount.

Class ApplyDmgModifiersStep

Namespace: [ElectricDrill.AstraRpgHealth.Damage.CalculationPipeline](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Damage step that applies generic and configured damage source/type modifiers (for example weaknesses and resistances) to the current damage amount.

```
public class ApplyDmgModifiersStep : DamageStep
```

Inheritance

object ← [DamageStep](#) ← ApplyDmgModifiersStep

Inherited Members

[DamageStep.Process\(DamageInfo\)](#).

Properties

DisplayName

Human readable name shown in pipeline editors.

```
public override string DisplayName { get; }
```

Property Value

string

Methods

ProcessStep(DamageInfo)

Applies configured percentage-based modifiers to [Amounts](#). If the configured adjustments lead to immunity or a terminal prevention condition the method will mark the [DamageInfo](#) with the appropriate [DamagePreventionReason](#).

```
public override DamageInfo ProcessStep(DamageInfo data)
```

Parameters

data [DamageInfo](#)

Current damage pipeline data.

Returns

[DamageInfo](#)

The same **data** instance updated with the modified amounts and any prevention reasons.

Class DamageCalculationStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Damage.CalculationPipeline](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

ScriptableObject that defines a configurable damage calculation pipeline. The pipeline is composed of ordered [DamageStep](#) instances executed sequentially.

```
[CreateAssetMenu(fileName = "New Configurable Strategy", menuName = "Astra RPG Health/Damage Calculation Pipeline/Configurable Strategy")]
public class DamageCalculationStrategy : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← DamageCalculationStrategy

Fields

steps

Ordered list of steps composing this calculation strategy. Steps are executed in sequence.

```
[SerializeField]
public List<DamageStep> steps
```

Field Value

List<[DamageStep](#)>

Methods

CalculateDamage(DamageInfo)

Executes the entire damage calculation pipeline defined in this asset.

```
public virtual DamageInfo CalculateDamage(DamageInfo data)
```

Parameters

data [DamagelInfo](#)

The initial damage information.

Returns

[DamagelInfo](#)

The damage information after all steps have been applied.

Class DamageInfo

Namespace: [ElectricDrill.AstraRpgHealth.Damage.CalculationPipeline](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Container for damage pipeline state while a damage calculation is executed. Holds the amounts being transformed, the source/type metadata and accumulated prevention reasons.

```
public class DamageInfo
```

Inheritance

object ← DamageInfo

Constructors

DamageInfo(PreDamageInfo)

Creates a new [DamageInfo](#) from the provided [PreDamageInfo](#). The constructor initializes amounts, metadata and sets pre-phase prevention reasons if applicable.

```
public DamageInfo(PreDamageInfo pre)
```

Parameters

`pre` [PreDamageInfo](#)

The immutable pre-damage description used to initialize this instance.

Properties

Amounts

Numeric amounts and intermediate records for the damage being processed.

```
public DamageAmountInfo Amounts { get; set; }
```

Property Value

[DamageAmountInfo](#)

CriticalMultiplier

Critical multiplier applied when [IsCritical](#) is true (1.0 means no change).

```
public double CriticalMultiplier { get; }
```

Property Value

double

DamageSource

The origin/source of the damage (spell, environmental, trap, etc.).

```
public DamageSource DamageSource { get; }
```

Property Value

[DamageSource](#)

Dealer

Entity that deals the damage (may be the same as target for self damage or null for environmental damage).

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

Property Value

EntityCore

IsCritical

Indicates whether the incoming hit was marked as critical.

```
public bool IsCritical { get; }
```

Property Value

bool

IsPrevented

Returns true when accumulated reasons include a terminal prevention condition.

```
public bool IsPrevented { get; }
```

Property Value

bool

Reasons

Reasons accumulated through the pipeline that caused damage to be prevented. This is a flag enum that can contain multiple reasons.

```
public DamagePreventionReason Reasons { get; }
```

Property Value

[DamagePreventionReason](#)

Target

Entity that is the target of this damage calculation.

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

Property Value

EntityCore

TerminationStepType

[DamageStep](#) that caused termination of the pipeline.

```
public Type TerminationStepType { get; }
```

Property Value

Type

Type

Configured damage type describing how the damage behaves (defenses, true, etc.).

```
public DamageType Type { get; }
```

Property Value

[DamageType](#)

Methods

AddReason(DamagePreventionReason, Type, bool)

Adds a prevention reason to the accumulated flags and optionally marks the termination step.

```
public void AddReason(DamagePreventionReason reason, Type stepType, bool terminate = true)
```

Parameters

reason [DamagePreventionReason](#)

Reason to add.

stepType Type

Type of the step that added the reason (can be null for pre-phase).

terminate bool

If true and the termination step is not already set, records **stepType** as the termination cause.

Class DamageStep

Namespace: [ElectricDrill.AstraRpgHealth.Damage.CalculationPipeline](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Base class for a single step in the damage calculation pipeline. A [DamageStep](#) performs a focused transformation on a [DamageInfo](#) instance (for example applying defenses, barriers or modifiers).

```
[Serializable]  
public abstract class DamageStep
```

Inheritance

object ← DamageStep

Derived

[ApplyBarrierStep](#), [ApplyCriticalMultiplierStep](#), [ApplyDefenseStep](#), [ApplyDmgModifiersStep](#)

Properties

DisplayName

Human-readable name shown in editors and logs for this step. Implementations should return a short descriptive label.

```
public abstract string DisplayName { get; }
```

Property Value

string

Methods

Process(DamageInfo)

Entry point executed by the pipeline runner. This method enforces common preconditions (skipping when already prevented or when the current amount is <= 0), invokes [ProcessStep\(DamageInfo\)](#), records

the before/after amounts for tracing and sets the [PipelineReducedToZero](#) reason when a step reduces a positive amount to zero or less.

```
public DamageInfo Process(DamageInfo data)
```

Parameters

data [DamageInfo](#)

The pipeline state being processed.

Returns

[DamageInfo](#)

The same **data** instance, potentially modified by the step.

ProcessStep(DamageInfo)

Implement this method in derived classes to perform the actual transformation of the pipeline state.

```
public abstract DamageInfo ProcessStep(DamageInfo data)
```

Parameters

data [DamageInfo](#)

Pipeline state to process. Implementations should mutate this instance to reflect changes.

Returns

[DamageInfo](#)

The modified **data** instance.

Namespace ElectricDrill.AstraRpgHealth. DamageReductionFunctions

Classes

[DamageReductionFn](#)

Base type for damage reduction functions. Implementations compute how an incoming damage amount is reduced using a defensive statistic value.

Class DamageReductionFn

Namespace: [ElectricDrill.AstraRpgHealth.DamageReductionFunctions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Base type for damage reduction functions. Implementations compute how an incoming damage amount is reduced using a defensive statistic value.

```
public abstract class DamageReductionFn : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← DamageReductionFn

Derived

[FlatDamageReductionFn](#), [LogDamageReductionFn](#), [PercentageDamageReductionFn](#)

Remarks

Implementations return a non-negative integer amount of damage. Implementations are responsible for any rounding or clamping behavior (for example, ensuring the returned value is not negative).

Methods

ReducedDmg(long, double)

Reduces the specified **amount** of damage using the provided defensive statistic value and returns the resulting damage amount.

```
public abstract long ReducedDmg(long amount, double defensiveStatValue)
```

Parameters

amount long

The original incoming damage amount to be reduced.

defensiveStatValue double

The value of the defensive statistic used to reduce damage (semantics depend on the implementation: absolute value, percentage, etc.).

Returns

long

The reduced damage amount as a non-negative long. Implementations should ensure the returned value is ≥ 0 and apply any rounding semantics they require.

Namespace ElectricDrill.AstraRpgHealth. DamageReductionFunctions.DamageReduction Functions

Classes

[FlatDamageReductionFn](#)

[LogDamageReductionFn](#)

[PercentageDamageReductionFn](#)

Class FlatDamageReductionFn

Namespace: [ElectricDrill.AstraRpgHealth.DamageReductionFunctions.DamageReductionFunctions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
[CreateAssetMenu(fileName = "Flat Damage Reduction Fn", menuName = "Astra RPG Health/Dmg Reduction Functions/Flat Dmg Reduction")]
public class FlatDamageReductionFn : DamageReductionFn
```

Inheritance

object ← Object ← ScriptableObject ← [DamageReductionFn](#) ← FlatDamageReductionFn

Methods

ReducedDmg(long, double)

Subtracts `defensiveStatValue * _factor` from the incoming `amount`. The result is clamped to a minimum of 0 and rounded to the nearest whole number.

```
public override long ReducedDmg(long amount, double defensiveStatValue)
```

Parameters

`amount` long

Incoming damage amount.

`defensiveStatValue` double

Value of the defensive statistic used to reduce damage. This value is multiplied by ElectricDrill.AstraRpgHealth.DamageReductionFunctions.DamageReductionFunctions.FlatDamageReductionFn._factor before being subtracted.

Returns

long

The reduced damage as a non-negative long (rounded).

Class LogDamageReductionFn

Namespace: [ElectricDrill.AstraRpgHealth.DamageReductionFunctions.DamageReductionFunctions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
[CreateAssetMenu(fileName = "Log Dmg Reduction Fn", menuName = "Astra RPG Health/Dmg Reduction Functions/Log Dmg Reduction")]
public class LogDamageReductionFn : DamageReductionFn
```

Inheritance

object ← Object ← ScriptableObject ← [DamageReductionFn](#) ← LogDamageReductionFn

Methods

ReducedDmg(long, double)

Reduces damage using a logarithmic formula that produces diminishing returns as the defensive stat increases.

```
public override long ReducedDmg(long amount, double defensiveStatValue)
```

Parameters

amount long

Incoming damage amount.

defensiveStatValue double

Defensive statistic value used to compute the reduction multiplier.

Returns

long

The reduced damage as a non-negative long (rounded).

Remarks

Formula: Reduced Damage = Damage * (BaseValue / (BaseValue + Log(1 + DefensiveStat * ScaleFactor)))

If `defensiveStatValue` is zero or negative the original damage is returned. The final result is clamped to be non-negative and rounded to the nearest integer.

Class PercentageDamageReductionFn

Namespace: [ElectricDrill.AstraRpgHealth.DamageReductionFunctions.DamageReductionFunctions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
[CreateAssetMenu(fileName = "Percentage Dmg Reduction Fn", menuName = "Astra RPG Health/Dmg Reduction Functions/Percentage Dmg Reduction")]
public class PercentageDamageReductionFn : DamageReductionFn
```

Inheritance

object ← Object ← ScriptableObject ← [DamageReductionFn](#) ← PercentageDamageReductionFn

Methods

ReducedDmg(long, double)

Reduces damage by a percentage determined by `defensiveStatValue`. Example: `defensiveStatValue == 25` -> damage reduced by 25%.

```
public override long ReducedDmg(long amount, double defensiveStatValue)
```

Parameters

`amount` long

Incoming damage amount.

`defensiveStatValue` double

Percentage (0..100+). Treated as a percent reduction of the incoming damage.

Returns

long

The reduced damage as a non-negative long (rounded).

Remarks

The computation is: $\text{reducedAmount} = \text{amount} - \text{amount} * \text{defensiveStatValue} / 100.0$ The result is clamped to a minimum of 0 and rounded to the nearest integer.

Namespace ElectricDrill.AstraRpgHealth.Death Classes

[DestroyOnDeathStrategy](#)

A death strategy that destroys the entity's UnityEngine.GameObject when it dies. This is provided as a ScriptableObject to be reused across entities. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Destroy.

[DisableOnDeathStrategy](#)

A death strategy that disables the entity's UnityEngine.GameObject when it dies. This is useful for entities that should be hidden but not destroyed, allowing for resurrection. Pairs well with EnableOnResurrectionStrategy to re-enable the GameObject on resurrection. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Disable.

[DoNothingOnDeathStrategy](#)

A no-op death strategy. Use this when no action is required upon an entity's death. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Do Nothing.

[MultipleOnDeathStrategy](#)

A death strategy that executes multiple death strategies in order. This allows combining multiple death behaviors without creating new strategy classes. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Multiple.

[OnDeathStrategy](#)

Base strategy type for handling an entity's death. Derive from this class and implement [Die\(Entity, Health\)](#) to define custom death behavior.

Class DestroyOnDeathStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Death](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

A death strategy that destroys the entity's UnityEngine.GameObject when it dies. This is provided as a ScriptableObject to be reused across entities. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Destroy.

```
[CreateAssetMenu(fileName = "Destroy On Death Strategy", menuName = "Astra RPG
Health/Death strategies/Destroy")]
public class DestroyOnDeathStrategy : OnDeathStrategy
```

Inheritance

object ← Object ← ScriptableObject ← [OnDeathStrategy](#) ← DestroyOnDeathStrategy

Methods

Die(EntityHealth)

Destroys the UnityEngine.GameObject associated with `entityHealth`.

```
public override void Die(EntityHealth entityHealth)
```

Parameters

`entityHealth` [EntityHealth](#)

The [EntityHealth](#) instance to destroy.

Class DisableOnDeathStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Death](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

A death strategy that disables the entity's UnityEngine.GameObject when it dies. This is useful for entities that should be hidden but not destroyed, allowing for resurrection. Pairs well with EnableOnResurrectionStrategy to re-enable the GameObject on resurrection. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Disable.

```
[CreateAssetMenu(fileName = "Disable On Death Strategy", menuName = "Astra RPG
Health/Death strategies/Disable")]
public class DisableOnDeathStrategy : OnDeathStrategy
```

Inheritance

object ← Object ← ScriptableObject ← [OnDeathStrategy](#) ← DisableOnDeathStrategy

Methods

Die(EntityHealth)

Disables the UnityEngine.GameObject associated with `entityHealth`.

```
public override void Die(EntityHealth entityHealth)
```

Parameters

`entityHealth` [EntityHealth](#)

The [EntityHealth](#) instance that died.

Class DoNothingOnDeathStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Death](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

A no-op death strategy. Use this when no action is required upon an entity's death. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Do Nothing.

```
[CreateAssetMenu(fileName = "Do Nothing On Death Strategy", menuName = "Astra RPG
Health/Death strategies/Do Nothing")]
public class DoNothingOnDeathStrategy : OnDeathStrategy
```

Inheritance

object ← Object ← ScriptableObject ← [OnDeathStrategy](#) ← DoNothingOnDeathStrategy

Methods

Die(EntityHealth)

Intentionally performs no action when an entity dies.

```
public override void Die(EntityHealth entityHealth)
```

Parameters

entityHealth [EntityHealth](#)

The [EntityHealth](#) instance that died (unused).

Class MultipleOnDeathStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Death](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

A death strategy that executes multiple death strategies in order. This allows combining multiple death behaviors without creating new strategy classes. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Multiple.

```
[CreateAssetMenu(fileName = "Multiple On Death Strategy", menuName = "Astra RPG
Health/Death strategies/Multiple")]
public class MultipleOnDeathStrategy : OnDeathStrategy
```

Inheritance

object ← Object ← ScriptableObject ← [OnDeathStrategy](#) ← MultipleOnDeathStrategy

Methods

Die(EntityHealth)

Executes all configured death strategies in order. Null strategies in the list are skipped.

```
public override void Die(EntityHealth entityHealth)
```

Parameters

entityHealth [EntityHealth](#)

The [EntityHealth](#) instance that died.

Class OnDeathStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Death](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Base strategy type for handling an entity's death. Derive from this class and implement [Die\(EntityHealth\)](#) to define custom death behavior.

```
public abstract class OnDeathStrategy : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← OnDeathStrategy

Derived

[DestroyOnDeathStrategy](#), [DisableOnDeathStrategy](#), [DoNothingOnDeathStrategy](#),
[MultipleOnDeathStrategy](#)

Methods

Die(EntityHealth)

Invoked when an entity should die. Implementations should perform any cleanup, state changes, or destruction logic here.

```
public abstract void Die(EntityHealth entityHealth)
```

Parameters

entityHealth [EntityHealth](#)

The [EntityHealth](#) instance representing the entity that died.

Namespace ElectricDrill.AstraRpgHealth. DefenseReductionFunctions

Classes

[DefenseReductionFn](#)

Base type for defense reduction functions. Implementations compute a reduced defense value based on a piercing stat and the pierced defense stat.

Class DefenseReductionFn

Namespace: [ElectricDrill.AstraRpgHealth.DefenseReductionFunctions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Base type for defense reduction functions. Implementations compute a reduced defense value based on a piercing stat and the pierced defense stat.

```
public abstract class DefenseReductionFn : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← DefenseReductionFn

Derived

[FlatDefenseReductionFn](#), [LogDefenseReductionFn](#), [PercentageDefenseReductionFn](#)

Methods

ReducedDef(long, long, Stat, bool)

Compute the reduced defense value after applying a piercing stat.

```
public abstract double ReducedDef(long piercingStatValue, long defensiveStatValue, Stat  
defensiveStat, bool applyClamp = true)
```

Parameters

piercingStatValue long

The attacker's piercing stat value used to reduce defense.

defensiveStatValue long

The defender's original defense stat value to be reduced.

defensiveStat Stat

The defensive stat object used to clamp the result within its min/max bounds.

applyClamp bool

Whether to apply stat clamping to the result. Default true for normal gameplay; false for analysis/graphs.

Returns

double

The reduced defense value (clamped if applyClamp is true).

Namespace ElectricDrill.AstraRpgHealth. DefenseReductionFunctions.DefenseReduction Functions

Classes

[FlatDefenseReductionFn](#)

Reduces defense by subtracting a scaled amount of the piercing stat from the pierced defense. The result is clamped to the stat's min/max bounds. Create instances via Assets -> Create -> Astra RPG Health / Def Reduction Functions / Flat Def Reduction.

[LogDefenseReductionFn](#)

Applies a logarithmic divisive reduction to defense to provide diminishing returns as piercing increases. Create instances via Assets -> Create -> Astra RPG Health / Def Reduction Functions / Log Def Reduction.

[PercentageDefenseReductionFn](#)

Reduces defense by subtracting a percentage of the pierced defense equal to the piercing stat percentage. Example: piercingStatValue = 20 -> reduces defense by 20%. Create instances via Assets -> Create -> Astra RPG Health / Def Reduction Functions / Log Def Reduction.

Class FlatDefenseReductionFn

Namespace: [ElectricDrill.AstraRpgHealth.DefenseReductionFunctions.DefenseReductionFunctions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Reduces defense by subtracting a scaled amount of the piercing stat from the pierced defense. The result is clamped to the stat's min/max bounds. Create instances via Assets -> Create -> Astra RPG Health / Def Reduction Functions / Flat Def Reduction.

```
[CreateAssetMenu(fileName = "Flat Def Reduction Fn", menuName = "Astra RPG Health/Def Reduction Functions/Flat Def Reduction")]
public class FlatDefenseReductionFn : DefenseReductionFn
```

Inheritance

object ← Object ← ScriptableObject ← [DefenseReductionFn](#) ← FlatDefenseReductionFn

Methods

ReducedDef(long, long, Stat, bool)

Subtracts (piercingStatValue * factor) from the pierced defense and optionally clamps the result to the stat's bounds.

```
public override double ReducedDef(long piercingStatValue, long defensiveStatValue, Stat
defensiveStat, bool applyClamp = true)
```

Parameters

piercingStatValue long

The attacker's piercing stat value used to reduce defense.

defensiveStatValue long

The defender's original defense stat value to be reduced.

defensiveStat Stat

The defensive stat object used to clamp the result.

applyClamp bool

Whether to apply stat clamping to the result.

Returns

double

The reduced defense value (clamped to stat's min/max bounds if applyClamp is true).

Class LogDefenseReductionFn

Namespace: [ElectricDrill.AstraRpgHealth.DefenseReductionFunctions.DefenseReductionFunctions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Applies a logarithmic divisive reduction to defense to provide diminishing returns as piercing increases.
Create instances via Assets -> Create -> Astra RPG Health / Def Reduction Functions / Log Def Reduction.

```
[CreateAssetMenu(fileName = "Log Def Reduction Fn", menuName = "Astra RPG Health/Def Reduction Functions/Log Def Reduction")]
public class LogDefenseReductionFn : DefenseReductionFn
```

Inheritance

object ← Object ← ScriptableObject ← [DefenseReductionFn](#) ← LogDefenseReductionFn

Methods

ReducedDef(long, long, Stat, bool)

Reduces defense using a logarithmic divisive formula that provides diminishing returns. Formula:
Reduced Defense = Defense * (BaseValue / (BaseValue + Log(1 + PiercingStat * ScaleFactor)))

```
public override double ReducedDef(long piercingStatValue, long defensiveStatValue, Stat
defensiveStat, bool applyClamp = true)
```

Parameters

piercingStatValue long

The attacker's piercing stat value.

defensiveStatValue long

The defender's original defense stat value.

defensiveStat Stat

The defensive stat object used to clamp the result.

applyClamp bool

Whether to apply stat clamping to the result.

Returns

double

The reduced defense value (clamped to stat's min/max bounds if applyClamp is true)

Class PercentageDefenseReductionFn

Namespace: [ElectricDrill.AstraRpgHealth.DefenseReductionFunctions.DefenseReductionFunctions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Reduces defense by subtracting a percentage of the pierced defense equal to the piercing stat percentage. Example: piercingStatValue = 20 -> reduces defense by 20%. Create instances via Assets -> Create -> Astra RPG Health / Def Reduction Functions / Log Def Reduction.

```
[CreateAssetMenu(fileName = "Percentage Def Reduction Fn", menuName = "Astra RPG Health/Def Reduction Functions/Percentage Def Reduction")]
public class PercentageDefenseReductionFn : DefenseReductionFn
```

Inheritance

object ← Object ← ScriptableObject ← [DefenseReductionFn](#) ← PercentageDefenseReductionFn

Methods

ReducedDef(long, long, Stat, bool)

Apply percentage-based reduction to the pierced defense and optionally clamp the result to the stat's bounds.

```
public override double ReducedDef(long piercingStatValue, long defensiveStatValue, Stat
defensiveStat, bool applyClamp = true)
```

Parameters

piercingStatValue long

Percentage to reduce (e.g. 25 means 25%).

defensiveStatValue long

Original defense stat.

defensiveStat Stat

The defensive stat object used to clamp the result.

applyClamp bool

Whether to apply stat clamping to the result.

Returns

double

The reduced defense value (clamped to stat's min/max bounds if applyClamp is true).

Namespace ElectricDrill.AstraRpgHealth.Events

Classes

[DamageResolutionGameEvent](#)

Used to notify whether a damage was applied or prevented.

[DamageResolutionGameEventListener](#)

Used to notify whether a damage was applied or prevented.

[EntityDiedGameEvent](#)

The entity died.

[EntityDiedGameEventListener](#)

The entity died.

[EntityGainedHealthGameEvent](#)

The entity gained some health.

[EntityGainedHealthGameEventListener](#)

The entity gained some health.

[EntityHealedGameEvent](#)

The entity received an healing.

[EntityHealedGameEventListener](#)

The entity received an healing.

[EntityLostHealthGameEvent](#)

The entity lost some health.

[EntityLostHealthGameEventListener](#)

The entity lost some health.

[EntityMaxHealthChangedGameEvent](#)

The entity whose Max HP changed, the new Max HP, and the old Max HP.

[EntityMaxHealthChangedGameEventListener](#)

The entity whose Max HP changed, the new Max HP, and the old Max HP.

[EntityResurrectedGameEvent](#)

The entity resurrected with the specified amount of health.

[EntityResurrectedGameEventListener](#)

The entity resurrected with the specified amount of health.

[PreDmgGameEvent](#)

[PreDmgGameEventListener](#)

[PreHealGameEvent](#)

THe entity is about to be healed.

[PreHealGameEventListener](#)

THe entity is about to be healed.

Class DamageResolutionGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Used to notify whether a damage was applied or prevented.

```
[CreateAssetMenu(fileName = "DamageResolution Game Event", menuName = "Astra RPG
Health/Events/Generated (Damage)/DamageResolution")]
public class DamageResolutionGameEvent : GameEventGeneric1<DamageResolution>,
IRaisable<DamageResolution>
```

Inheritance

```
object ← Object ← ScriptableObject ← GameEventGeneric1<DamageResolution> ←
DamageResolutionGameEvent
```

Implements

```
IRaisable<DamageResolution>
```

Inherited Members

```
GameEventGeneric1<DamageResolution>.OnEventRaised ,
GameEventGeneric1<DamageResolution>.Raise(DamageResolution) ,
GameEventGeneric1<DamageResolution>.RegisterListener(GameEventListenerGeneric1<DamageResolut
ion>) ,
GameEventGeneric1<DamageResolution>.UnregisterListener(GameEventListenerGeneric1<DamageResolut
ion>)
```

Class DamageResolutionGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Used to notify whether a damage was applied or prevented.

```
public class DamageResolutionGameEventListener : GameEventListenerGeneric1<DamageResolution>
```

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<DamageResolution> ← DamageResolutionGameEventListener
```

Inherited Members

```
GameEventListenerGeneric1<DamageResolution>._event ,  
GameEventListenerGeneric1<DamageResolution>._response ,  
GameEventListenerGeneric1<DamageResolution>.OnEventRaised(DamageResolution)
```

Class EntityDiedGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity died.

```
[CreateAssetMenu(fileName = "EntityDied Game Event", menuName = "Astra RPG
Health/Events/Generated (Health)/EntityDied")]
public class EntityDiedGameEvent : GameEventGeneric2<EntityHealth, DamageResolution>,
IRaisable<EntityHealth, DamageResolution>
```

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric2<[EntityHealth](#), [DamageResolution](#)> ← EntityDiedGameEvent

Implements

IRaisable<[EntityHealth](#), [DamageResolution](#)>

Inherited Members

GameEventGeneric2<EntityHealth, DamageResolution>.OnEventRaised ,
GameEventGeneric2<EntityHealth, DamageResolution>.Raise(EntityHealth, DamageResolution) ,
GameEventGeneric2<EntityHealth,
DamageResolution>.RegisterListener(GameEventListenerGeneric2<EntityHealth, DamageResolution>) ,
GameEventGeneric2<EntityHealth,
DamageResolution>.UnregisterListener(GameEventListenerGeneric2<EntityHealth, DamageResolution>)

Class EntityDiedGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity died.

```
public class EntityDiedGameEventListener : GameEventListenerGeneric2<EntityHealth,  
DamageResolution>
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ←
GameEventListenerGeneric2<[EntityHealth](#), [DamageResolution](#)> ← EntityDiedGameEventListener

Inherited Members

GameEventListenerGeneric2<EntityHealth, DamageResolution>._event ,
GameEventListenerGeneric2<EntityHealth, DamageResolution>._response ,
GameEventListenerGeneric2<EntityHealth, DamageResolution>.OnEventRaised(EntityHealth,
DamageResolution)

Class EntityGainedHealthGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity gained some health.

```
[CreateAssetMenu(fileName = "EntityGainedHealth Game Event", menuName = "Astra RPG
Health/Events/Generated (Health)/EntityGainedHealth")]
public class EntityGainedHealthGameEvent : GameEventGeneric2<EntityHealth, long>,
IRaisable<EntityHealth, long>
```

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric2<[EntityHealth](#), long> ← EntityGainedHealthGameEvent

Implements

IRaisable<[EntityHealth](#), long>

Inherited Members

GameEventGeneric2<EntityHealth, long>.OnEventRaised ,
GameEventGeneric2<EntityHealth, long>.Raise(EntityHealth, long) ,
GameEventGeneric2<EntityHealth, long>.RegisterListener(GameEventListenerGeneric2<EntityHealth, long>) ,
GameEventGeneric2<EntityHealth, long>.UnregisterListener(GameEventListenerGeneric2<EntityHealth, long>)

Class EntityGainedHealthGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity gained some health.

```
public class EntityGainedHealthGameEventListener : GameEventListenerGeneric2<EntityHealth,  
long>
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ←
GameEventListenerGeneric2<[EntityHealth](#), long> ← EntityGainedHealthGameEventListener

Inherited Members

GameEventListenerGeneric2<EntityHealth, long>._event ,
GameEventListenerGeneric2<EntityHealth, long>._response ,
GameEventListenerGeneric2<EntityHealth, long>.OnEventRaised(EntityHealth, long)

Class EntityHealedGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity received an healing.

```
[CreateAssetMenu(fileName = "EntityHealed Game Event", menuName = "Astra RPG
Health/Events/Generated (Health)/EntityHealed")]
public class EntityHealedGameEvent : GameEventGeneric1<ReceivedHealInfo>,
IRaisable<ReceivedHealInfo>
```

Inheritance

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

Implements

IRaisable<[ReceivedHealInfo](#)>

Inherited Members

GameEventGeneric1<ReceivedHealInfo>.OnEventRaised ,
GameEventGeneric1<ReceivedHealInfo>.Raise(ReceivedHealInfo) ,
GameEventGeneric1<ReceivedHealInfo>.RegisterListener(GameEventListenerGeneric1<ReceivedHealInfo>) ,
GameEventGeneric1<ReceivedHealInfo>.UnregisterListener(GameEventListenerGeneric1<ReceivedHealInfo>)

Class EntityHealedGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity received an healing.

```
public class EntityHealedGameEventListener : GameEventListenerGeneric1<ReceivedHealInfo>
```

Inheritance

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

Inherited Members

GameEventListenerGeneric1<ReceivedHealInfo>._event ,
GameEventListenerGeneric1<ReceivedHealInfo>._response ,
GameEventListenerGeneric1<ReceivedHealInfo>.OnEventRaised(ReceivedHealInfo)

Class EntityLostHealthGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity lost some health.

```
[CreateAssetMenu(fileName = "EntityLostHealth Game Event", menuName = "Astra RPG
Health/Events/Generated (Health)/EntityLostHealth")]
public class EntityLostHealthGameEvent : GameEventGeneric2<EntityHealth, long>,
IRaisable<EntityHealth, long>
```

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric2<[EntityHealth](#), long> ← EntityLostHealthGameEvent

Implements

IRaisable<[EntityHealth](#), long>

Inherited Members

GameEventGeneric2<EntityHealth, long>.OnEventRaised ,
GameEventGeneric2<EntityHealth, long>.Raise(EntityHealth, long) ,
GameEventGeneric2<EntityHealth, long>.RegisterListener(GameEventListenerGeneric2<EntityHealth, long>) ,
GameEventGeneric2<EntityHealth, long>.UnregisterListener(GameEventListenerGeneric2<EntityHealth, long>)

Class EntityLostHealthGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity lost some health.

```
public class EntityLostHealthGameEventListener : GameEventListenerGeneric2<EntityHealth,  
long>
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ←
GameEventListenerGeneric2<[EntityHealth](#), long> ← EntityLostHealthGameEventListener

Inherited Members

GameEventListenerGeneric2<EntityHealth, long>._event ,
GameEventListenerGeneric2<EntityHealth, long>._response ,
GameEventListenerGeneric2<EntityHealth, long>.OnEventRaised(EntityHealth, long)

Class EntityMaxHealthChangedGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity whose Max HP changed, the new Max HP, and the old Max HP.

```
[CreateAssetMenu(fileName = "EntityMaxHealthChanged Game Event", menuName = "Astra RPG
Health/Events/Generated (Health)/EntityMaxHealthChanged")]
public class EntityMaxHealthChangedGameEvent : GameEventGeneric3<EntityHealth, long, long>,
IRaisable<EntityHealth, long, long>
```

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric3<[EntityHealth](#), long, long> ← EntityMaxHealthChangedGameEvent

Implements

IRaisable<[EntityHealth](#), long, long>

Inherited Members

GameEventGeneric3<EntityHealth, long, long>.OnEventRaised ,
GameEventGeneric3<EntityHealth, long, long>.Raise(EntityHealth, long, long) ,
GameEventGeneric3<EntityHealth, long,
long>.RegisterListener(GameEventListenerGeneric3<EntityHealth, long, long>) ,
GameEventGeneric3<EntityHealth, long,
long>.UnregisterListener(GameEventListenerGeneric3<EntityHealth, long, long>)

Class

EntityMaxHealthChangedGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity whose Max HP changed, the new Max HP, and the old Max HP.

```
public class EntityMaxHealthChangedGameEventListener :  
GameEventListenerGeneric3<EntityHealth, long, long>
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ←
GameEventListenerGeneric3<[EntityHealth](#), long, long> ← EntityMaxHealthChangedGameEventListener

Inherited Members

GameEventListenerGeneric3<EntityHealth, long, long>._event ,
GameEventListenerGeneric3<EntityHealth, long, long>._response ,
GameEventListenerGeneric3<EntityHealth, long, long>.OnEventRaised(EntityHealth, long, long)

Class EntityResurrectedGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity resurrected with the specified amount of health.

```
[CreateAssetMenu(fileName = "EntityResurrected Game Event", menuName = "Astra RPG
Health/Events/Generated (Health)/EntityResurrected")]
public class EntityResurrectedGameEvent : GameEventGeneric2<EntityHealth, ResurrectionInfo>,
IRaisable<EntityHealth, ResurrectionInfo>
```

Inheritance

```
object ← Object ← ScriptableObject ← GameEventGeneric2<EntityHealth, ResurrectionInfo> ←
EntityResurrectedGameEvent
```

Implements

```
IRaisable<EntityHealth, ResurrectionInfo>
```

Inherited Members

```
GameEventGeneric2<EntityHealth, ResurrectionInfo>.OnEventRaised ,
GameEventGeneric2<EntityHealth, ResurrectionInfo>.Raise(EntityHealth, ResurrectionInfo) ,
GameEventGeneric2<EntityHealth,
ResurrectionInfo>.RegisterListener(GameEventListenerGeneric2<EntityHealth, ResurrectionInfo>) ,
GameEventGeneric2<EntityHealth,
ResurrectionInfo>.UnregisterListener(GameEventListenerGeneric2<EntityHealth, ResurrectionInfo>)
```

Class EntityResurrectedGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity resurrected with the specified amount of health.

```
public class EntityResurrectedGameEventListener : GameEventListenerGeneric2<EntityHealth,  
ResurrectionInfo>
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ←
GameEventListenerGeneric2<[EntityHealth](#), [ResurrectionInfo](#)> ← EntityResurrectedGameEventListener

Inherited Members

GameEventListenerGeneric2<EntityHealth, ResurrectionInfo>._event ,
GameEventListenerGeneric2<EntityHealth, ResurrectionInfo>._response ,
GameEventListenerGeneric2<EntityHealth, ResurrectionInfo>.OnEventRaised(EntityHealth,
ResurrectionInfo)

Class PreDmgGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
[CreateAssetMenu(fileName = "PreDmg Game Event", menuName = "Astra RPG
Health/Events/Generated (Damage)/PreDmg")]
public class PreDmgGameEvent : GameEventGeneric1<PreDamageInfo>, IRaisable<PreDamageInfo>
```

Inheritance

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

Implements

IRaisable<[PreDamageInfo](#)>

Inherited Members

```
GameEventGeneric1<PreDamageInfo>.OnEventRaised ,
GameEventGeneric1<PreDamageInfo>.Raise(PreDamageInfo) ,
GameEventGeneric1<PreDamageInfo>.RegisterListener(GameEventListenerGeneric1<PreDamageInfo>) ,
GameEventGeneric1<PreDamageInfo>.UnregisterListener(GameEventListenerGeneric1<PreDamageInfo>
)
```

Class PreDmgGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
public class PreDmgGameEventListener : GameEventListenerGeneric1<PreDamageInfo>
```

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<PreDamageInfo> ← PreDmgGameEventListener
```

Inherited Members

```
GameEventListenerGeneric1<PreDamageInfo>._event ,  
GameEventListenerGeneric1<PreDamageInfo>._response ,  
GameEventListenerGeneric1<PreDamageInfo>.OnEventRaised(PreDamageInfo)
```

Class PreHealGameEvent

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity is about to be healed.

```
[CreateAssetMenu(fileName = "PreHeal Game Event", menuName = "Astra RPG
Health/Events/Generated (Health)/PreHeal")]
public class PreHealGameEvent : GameEventGeneric2<PreHealInfo, EntityCore>,
IRaisable<PreHealInfo, EntityCore>
```

Inheritance

object ← Object ← ScriptableObject ← GameEventGeneric2<[PreHealInfo](#), EntityCore> ←
PreHealGameEvent

Implements

IRaisable<[PreHealInfo](#), EntityCore>

Inherited Members

GameEventGeneric2<PreHealInfo, EntityCore>.OnEventRaised ,
GameEventGeneric2<PreHealInfo, EntityCore>.Raise(PreHealInfo, EntityCore) ,
GameEventGeneric2<PreHealInfo,
EntityCore>.RegisterListener(GameEventListenerGeneric2<PreHealInfo, EntityCore>) ,
GameEventGeneric2<PreHealInfo,
EntityCore>.UnregisterListener(GameEventListenerGeneric2<PreHealInfo, EntityCore>)

Class PreHealGameEventListener

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

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

The entity is about to be healed.

```
public class PreHealGameEventListener : GameEventListenerGeneric2<PreHealInfo, EntityCore>
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ←
GameEventListenerGeneric2<[PreHealInfo](#), EntityCore> ← PreHealGameEventListener

Inherited Members

GameEventListenerGeneric2<PreHealInfo, EntityCore>._event ,
GameEventListenerGeneric2<PreHealInfo, EntityCore>._response ,
GameEventListenerGeneric2<PreHealInfo, EntityCore>.OnEventRaised(PreHealInfo, EntityCore)

Namespace ElectricDrill.AstraRpgHealth. Exceptions

Classes

[DeadEntityException](#)

Exception thrown when attempting to perform health-modifying operations on a dead entity. This exception indicates a programming error where the caller attempted to heal, damage, or otherwise modify the health of an entity that has already died.

Class DeadEntityException

Namespace: [ElectricDrill.AstraRpgHealth.Exceptions](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Exception thrown when attempting to perform health-modifying operations on a dead entity. This exception indicates a programming error where the caller attempted to heal, damage, or otherwise modify the health of an entity that has already died.

```
public class DeadEntityException : InvalidOperationException
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← DeadEntityException

Constructors

DeadEntityException(string, long, long, string)

Initializes a new instance of the DeadEntityException class.

```
public DeadEntityException(string entityName, long currentHp, long deathThreshold,  
    string attemptedOperation)
```

Parameters

entityName string

The name of the dead entity.

currentHp long

The current health points of the entity.

deathThreshold long

The death threshold value.

attemptedOperation string

The operation that was attempted.

Properties

AttemptedOperation

Gets the operation that was attempted on the dead entity.

```
public string AttemptedOperation { get; }
```

Property Value

string

CurrentHp

Gets the current health points of the dead entity.

```
public long CurrentHp { get; }
```

Property Value

long

DeathThreshold

Gets the death threshold of the entity (health value at or below which the entity is considered dead).

```
public long DeathThreshold { get; }
```

Property Value

long

EntityName

Gets the name of the entity that was dead when the operation was attempted.

```
public string EntityName { get; }
```

Property Value

string

Namespace ElectricDrill.AstraRpgHealth. Experience

Classes

[ExpCollector](#)

Collects experience when this entity deals killing blows. Requires an [EntityDiedGameEventListener](#) on the GameObject.

Class ExpCollector

Namespace: [ElectricDrill.AstraRpgHealth.Experience](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Collects experience when this entity deals killing blows. Requires an [EntityDiedGameEventListener](#) on the GameObject.

```
[RequireComponent(typeof(EntityDiedGameEventListener))]  
public class ExpCollector : MonoBehaviour
```

Inheritance

object ← Object ← Component ← Behaviour ← MonoBehaviour ← ExpCollector

Methods

CheckCollectKillExp(EntityHealth, DamageResolution)

Check whether the provided damage info represents a kill performed by this entity and collect experience if the victim is an experience source.

```
public void CheckCollectKillExp(EntityHealth entityHealth, DamageResolution dmgInfo)
```

Parameters

entityHealth [EntityHealth](#)

The health component of the entity that died.

dmgInfo [DamageResolution](#)

Damage resolution information that produced the death.

CollectExp(IExpSource)

Adds the experience provided by the given ElectricDrill.AstraRpgFramework.Experience.IExpSource to this entity's level component.

```
public void CollectExp(IExpSource expSource)
```

Parameters

expSource IExpSource

The experience source that provides the amount to collect.

Namespace ElectricDrill.AstraRpgHealth.Heal

Classes

[HealAmountInfo](#)

Represents the different numeric stages of a heal amount as it is processed: the raw requested amount, the amount after applying a heal modifier, and the final net amount applied.

[HealSource](#)

Defines a heal source asset that identifies how a heal was produced (e.g. spell, potion, resurrection, system, ...). Create instances via Assets -> Create -> Astra RPG Health / Heal Source.

[LifestealAmountSelector](#)

Selects which damage amount will be used as the basis for lifesteal calculations.

[LifestealConfig](#)

ScriptableObject for configuring lifesteal mappings for different damage types.

[LifestealStatConfig](#)

Configuration that associates a lifesteal Stat and HealSource with an optional amount selector.

[PreHealInfo](#)

Immutable description of a pending heal operation that can be passed through heal processing logic before the actual health change is applied.

[PreHealInfo.PreHealInfoStepBuilder](#)

Concrete builder implementing the fluent step interfaces to construct a [PreHealInfo](#).

[ReceivedHealInfo](#)

Represents the result of a heal operation as actually received by the target. Contains the resolved heal amount, source, involved entities and whether it was critical.

[ReceivedHealInfo.ReceivedHealInfoStepBuilder](#)

Concrete fluent builder used to assemble a [ReceivedHealInfo](#). Use [Builder](#) to start.

Interfaces

[IHealable](#)

Interface for entities that can receive healing. Implementers apply healing based on the provided [PreHealInfo](#).

[PreHealInfo.HealInfoAmount](#)

Builder step interface for specifying the heal amount.

[PreHealInfo.HealInfoHealer](#)

Builder step interface for specifying the healer entity and optional extras.

[PreHealInfo.HealInfoSource](#)

Builder step interface for specifying the heal source.

[ReceivedHealInfo.HealInfoAmount](#)

Builder step: provide the heal amount.

[ReceivedHealInfo.HealInfoHealer](#)

Builder step: provide the healer entity.

[ReceivedHealInfo.HealInfoSource](#)

Builder step: provide the heal source.

[ReceivedHealInfo.HealInfoTarget](#)

Builder step: provide the target entity and finalize options.

Enums

[LifestealBasisMode](#)

Specifies which damage value should be used as the basis for lifesteal calculations.

[StepValuePoint](#)

Selects whether to use the pre-step or post-step value when [Step](#) is selected.

Class HealAmountInfo

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Represents the different numeric stages of a heal amount as it is processed: the raw requested amount, the amount after applying a heal modifier, and the final net amount applied.

```
public class HealAmountInfo
```

Inheritance

object ← HealAmountInfo

Constructors

HealAmountInfo(long, long, long)

Create a new instance describing the provided heal values.

```
public HealAmountInfo(long rawAmount, long afterModifierAmount, long netAmount)
```

Parameters

rawAmount long

The initial requested heal amount.

afterModifierAmount long

The amount after applying modifier/stat adjustments.

netAmount long

The final amount actually applied to the target's health.

Properties

AfterModifierAmount

Gets the healing amount after being modified by the associated heal modifier statistic.

```
public long AfterModifierAmount { get; }
```

Property Value

long

NetAmount

Gets the actual amount of health added to the entity, considering the entity's maximum health.

```
public long NetAmount { get; }
```

Property Value

long

RawAmount

Gets the initial healing amount derived directly from PreHealInfo.

```
public long RawAmount { get; }
```

Property Value

long

Class HealSource

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Defines a heal source asset that identifies how a heal was produced (e.g. spell, potion, resurrection, system, ...). Create instances via Assets -> Create -> Astra RPG Health / Heal Source.

```
[CreateAssetMenu(fileName = "New Heal Source", menuName = "Astra RPG Health/Heal Source")]
public class HealSource : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← HealSource

Methods

ToString()

Returns the asset name of this heal source.

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

Returns

string

Interface IHealable

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Interface for entities that can receive healing. Implementers apply healing based on the provided [PreHealInfo](#).

```
public interface IHealable
```

Methods

Heal(PreHealInfo)

Apply a heal described by `info` and return information about the received heal.

```
ReceivedHealInfo Heal(PreHealInfo info)
```

Parameters

`info` [PreHealInfo](#)

Pre-heal context used to compute the actual heal applied.

Returns

[ReceivedHealInfo](#)

A [ReceivedHealInfo](#) describing what was actually applied to the target.

Class LifestealAmountSelector

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Selects which damage amount will be used as the basis for lifesteal calculations.

```
[Serializable]  
public class LifestealAmountSelector
```

Inheritance

object ← LifestealAmountSelector

Constructors

LifestealAmountSelector()

Initializes a new instance of [LifestealAmountSelector](#) using default values.

```
public LifestealAmountSelector()
```

LifestealAmountSelector(LifestealBasisMode, string, StepValuePoint)

Initializes a new instance of [LifestealAmountSelector](#) with the specified selection mode, optional pipeline step type name and step value point.

```
public LifestealAmountSelector(LifestealBasisMode mode, string stepTypeName,  
StepValuePoint stepPoint)
```

Parameters

mode [LifestealBasisMode](#)

Which basis to use (Initial, Step, Final).

stepTypeName string

Assembly-qualified or simple type name of the pipeline step to sample when [Step](#) is used.

stepPoint [StepValuePoint](#)

Whether to use the pre-step or post-step value when sampling a step.

Properties

Mode

Gets or sets the basis mode used to select the damage amount for lifesteal.

```
public LifestealBasisMode Mode { get; set; }
```

Property Value

[LifestealBasisMode](#)

StepPoint

Gets or sets whether to use the pre-step or post-step recorded value when sampling a pipeline step.

```
public StepValuePoint StepPoint { get; set; }
```

Property Value

[StepValuePoint](#)

StepType

Gets or sets the System.Type of the pipeline step to sample when [Mode](#) is [Step](#). Setting this property stores the assembly-qualified name; getting the property resolves the stored name to a runtime System.Type.

```
public Type StepType { get; set; }
```

Property Value

Type

Methods

Evaluate(DamageAmountInfo)

Evaluates the damage amounts and returns the long value that should be used for lifesteal according to this selector.

```
public long Evaluate(DamageAmountInfo amounts)
```

Parameters

amounts [DamageAmountInfo](#)

DamageAmountInfo instance containing initial, current and per-step records.

Returns

long

The selected damage value to compute lifesteal from.

Enum LifestealBasisMode

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Specifies which damage value should be used as the basis for lifesteal calculations.

```
[Serializable]
public enum LifestealBasisMode
```

Fields

Final = 2

Use the final damage amount (after all pipeline steps).

Initial = 0

Use the initial damage amount (before any pipeline modifications).

Step = 1

Use the damage recorded at a specific pipeline step (requires a step type).

Class LifestealConfig

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

ScriptableObject for configuring lifesteal mappings for different damage types.

```
[CreateAssetMenu(fileName = "New Lifesteal Config", menuName = "Astra RPG
Health/Lifesteal Config")]
public class LifestealConfig : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← LifestealConfig

Properties

LifestealMappings

Read-only mapping from DamageType to LifestealStatConfig.

```
public IReadOnlyDictionary<DamageType, LifestealStatConfig> LifestealMappings { get; }
```

Property Value

IReadOnlyDictionary<[DamageType](#), [LifestealStatConfig](#)>

Remarks

The dictionary is guaranteed to be non-null when accessed; calling code can safely enumerate it.

Class LifestealStatConfig

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Configuration that associates a lifesteal Stat and HealSource with an optional amount selector.

```
[Serializable]  
public class LifestealStatConfig
```

Inheritance

object ← LifestealStatConfig

Properties

AmountSelector

Selector that determines which damage amount (initial/step/final) is used to compute the heal.

```
public LifestealAmountSelector AmountSelector { get; }
```

Property Value

[LifestealAmountSelector](#)

LifestealSource

The HealSource that will be used when applying lifesteal.

```
public HealSource LifestealSource { get; }
```

Property Value

[HealSource](#)

LifestealStat

The Stat that will receive lifesteal when this mapping is used.

```
public Stat LifestealStat { get; }
```

Property Value

Stat

Class PreHealInfo

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Immutable description of a pending heal operation that can be passed through heal processing logic before the actual health change is applied.

```
public class PreHealInfo
```

Inheritance

object ← PreHealInfo

Properties

Amount

The requested heal amount (base value before modifiers).

```
public long Amount { get; }
```

Property Value

long

Builder

Fluent builder entry point for creating a [PreHealInfo](#). Use the returned builder to set amount, source and healer.

```
public static PreHealInfo.HealInfoAmount Builder { get; }
```

Property Value

[PreHealInfo.HealInfoAmount](#)

CriticalMultiplier

The multiplicative factor applied when [IsCritical](#) is true.

```
public double CriticalMultiplier { get; }
```

Property Value

double

HealSource

The [HealSource](#) identifying the origin/type of the heal.

```
public HealSource HealSource { get; }
```

Property Value

[HealSource](#)

Healer

The entity that caused the heal (may be null for non-entity sources).

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

Property Value

EntityCore

IsCritical

Whether this heal is flagged as a critical heal.

```
public bool IsCritical { get; }
```

Property Value

bool

Interface PreHealInfo.HealInfoAmount

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Builder step interface for specifying the heal amount.

```
public interface PreHealInfo.HealInfoAmount
```

Methods

WithAmount(long)

Specify the base heal amount.

```
PreHealInfo.HealInfoSource WithAmount(long amount)
```

Parameters

amount long

Returns

[PreHealInfo.HealInfoSource](#)

Interface PreHealInfo.HealInfoHealer

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Builder step interface for specifying the healer entity and optional extras.

```
public interface PreHealInfo.HealInfoHealer
```

Methods

WithHealer(EntityCore)

Specify the entity that performed the heal.

```
PreHealInfo.PreHealInfoStepBuilder WithHealer(EntityCore healer)
```

Parameters

healer EntityCore

Returns

[PreHealInfo.PreHealInfoStepBuilder](#)

Interface PreHealInfo.HealInfoSource

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Builder step interface for specifying the heal source.

```
public interface PreHealInfo.HealInfoSource
```

Methods

WithSource(HealSource)

Specify the [HealSource](#) for the heal.

```
PreHealInfo.HealInfoHealer WithSource(HealSource healSource)
```

Parameters

healSource [HealSource](#)

Returns

[PreHealInfo.HealInfoHealer](#)

Class PreHealInfo.PreHealInfoStepBuilder

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Concrete builder implementing the fluent step interfaces to construct a [PreHealInfo](#).

```
public sealed class PreHealInfo.PreHealInfoStepBuilder : PreHealInfo.HealInfoAmount,  
    PreHealInfo.HealInfoSource, PreHealInfo.HealInfoHealer
```

Inheritance

object ← PreHealInfo.PreHealInfoStepBuilder

Implements

[PreHealInfo.HealInfoAmount](#), [PreHealInfo.HealInfoSource](#), [PreHealInfo.HealInfoHealer](#)

Methods

Build()

Build and return the configured [PreHealInfo](#).

```
public PreHealInfo Build()
```

Returns

[PreHealInfo](#)

WithAmount(long)

Set the base heal amount.

```
public PreHealInfo.HealInfoSource WithAmount(long amount)
```

Parameters

`amount` long

Returns

[PreHealInfo.HealInfoSource](#)

WithCriticalMultiplier(double)

Set the critical multiplier applied when [`IsCritical`](#) is true.

```
public PreHealInfo.PreHealInfoStepBuilder WithCriticalMultiplier(double multiplier)
```

Parameters

`multiplier` double

Returns

[PreHealInfo.PreHealInfoStepBuilder](#)

WithHealer(EntityCore)

Set the entity that performed the heal.

```
public PreHealInfo.PreHealInfoStepBuilder WithHealer(EntityCore healer)
```

Parameters

`healer` EntityCore

Returns

[PreHealInfo.PreHealInfoStepBuilder](#)

WithIsCritical(bool)

Mark the built [PreHealInfo](#) as critical or not.

```
public PreHealInfo.PreHealInfoStepBuilder WithIsCritical(bool isCritical)
```

Parameters

isCritical bool

Returns

[PreHealInfo.PreHealInfoStepBuilder](#)

WithSource(HealSource)

Set the heal source.

```
public PreHealInfo.HealInfoHealer WithSource(HealSource healSource)
```

Parameters

healSource [HealSource](#)

Returns

[PreHealInfo.HealInfoHealer](#)

Class ReceivedHealInfo

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Represents the result of a heal operation as actually received by the target. Contains the resolved heal amount, source, involved entities and whether it was critical.

```
public class ReceivedHealInfo
```

Inheritance

object ← ReceivedHealInfo

Constructors

ReceivedHealInfo(HealAmountInfo, PreHealInfo, EntityCore)

Create a ReceivedHealInfo from the computed heal amount and the pre-heal context.

```
public ReceivedHealInfo(HealAmountInfo healAmount, PreHealInfo preHealInfo,  
EntityCore target)
```

Parameters

healAmount [HealAmountInfo](#)

Resolved heal amount information.

preHealInfo [PreHealInfo](#)

Pre-heal context that produced metadata such as source and healer.

target EntityCore

Entity that will receive the heal.

Properties

Builder

Entry point to build a [ReceivedHealInfo](#) using a step builder.

```
public static ReceivedHealInfo.HealInfoAmount Builder { get; }
```

Property Value

[ReceivedHealInfo.HealInfoAmount](#)

HealAmount

The calculated heal amount that will be applied to the target.

```
public HealAmountInfo HealAmount { get; }
```

Property Value

[HealAmountInfo](#)

HealSource

The [HealSource](#) that produced this heal.

```
public HealSource HealSource { get; }
```

Property Value

[HealSource](#)

Healer

The entity that performed the heal (may be null for system heals).

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

Property Value

EntityCore

IsCritical

True if the heal was a critical heal.

```
public bool IsCritical { get; }
```

Property Value

bool

Target

The entity that receives the heal.

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

Property Value

EntityCore

Interface ReceivedHealInfo.HealInfoAmount

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Builder step: provide the heal amount.

```
public interface ReceivedHealInfo.HealInfoAmount
```

Methods

WithAmount(HealAmountInfo)

Continue building by specifying the resolved [HealAmountInfo](#).

```
ReceivedHealInfo.HealInfoSource WithAmount(HealAmountInfo healAmount)
```

Parameters

healAmount [HealAmountInfo](#)

Returns

[ReceivedHealInfo.HealInfoSource](#)

Interface ReceivedHealInfo.HealInfoHealer

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Builder step: provide the healer entity.

```
public interface ReceivedHealInfo.HealInfoHealer
```

Methods

WithHealer(EntityCore)

Continue building by specifying the healer ElectricDrill.AstraRpgFramework.EntityCore.

```
ReceivedHealInfo.HealInfoTarget WithHealer(EntityCore healer)
```

Parameters

healer EntityCore

Returns

[ReceivedHealInfo.HealInfoTarget](#)

Interface ReceivedHealInfo.HealInfoSource

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Builder step: provide the heal source.

```
public interface ReceivedHealInfo.HealInfoSource
```

Methods

WithSource(HealSource)

Continue building by specifying the [HealSource](#).

```
ReceivedHealInfo.HealInfoHealer WithSource(HealSource healSource)
```

Parameters

healSource [HealSource](#)

Returns

[ReceivedHealInfo.HealInfoHealer](#)

Interface ReceivedHealInfo.HealInfoTarget

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Builder step: provide the target entity and finalize options.

```
public interface ReceivedHealInfo.HealInfoTarget
```

Methods

WithTarget(EntityCore)

Final builder step: specify the target, which returns the step builder for optional flags and build.

```
ReceivedHealInfo.ReceivedHealInfoStepBuilder WithTarget(EntityCore target)
```

Parameters

target EntityCore

Returns

[ReceivedHealInfo.ReceivedHealInfoStepBuilder](#)

Class

ReceivedHealInfo.ReceivedHealInfoStepBuilder

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Concrete fluent builder used to assemble a [ReceivedHealInfo](#). Use [Builder](#) to start.

```
public sealed class ReceivedHealInfo.ReceivedHealInfoStepBuilder :  
    ReceivedHealInfo.HealInfoAmount, ReceivedHealInfo.HealInfoSource,  
    ReceivedHealInfo.HealInfoHealer, ReceivedHealInfo.HealInfoTarget
```

Inheritance

object ← ReceivedHealInfo.ReceivedHealInfoStepBuilder

Implements

[ReceivedHealInfo.HealInfoAmount](#), [ReceivedHealInfo.HealInfoSource](#), [ReceivedHealInfo.HealInfoHealer](#),
[ReceivedHealInfo.HealInfoTarget](#)

Properties

Builder

Starts a new builder for [ReceivedHealInfo](#).

```
public static ReceivedHealInfo.HealInfoAmount Builder { get; }
```

Property Value

[ReceivedHealInfo.HealInfoAmount](#)

Methods

Build()

Build the final [ReceivedHealInfo](#) instance.

```
public ReceivedHealInfo Build()
```

Returns

[ReceivedHealInfo](#)

WithAmount(HealAmountInfo)

Set the resolved [HealAmountInfo](#).

```
public ReceivedHealInfo.HealInfoSource WithAmount(HealAmountInfo healAmount)
```

Parameters

healAmount [HealAmountInfo](#)

Returns

[ReceivedHealInfo.HealInfoSource](#)

WithHealer(EntityCore)

Set the entity that performed the heal.

```
public ReceivedHealInfo.HealInfoTarget WithHealer(EntityCore healer)
```

Parameters

healer EntityCore

Returns

[ReceivedHealInfo.HealInfoTarget](#)

WithIsCritical(bool)

Optionally mark the heal as critical.

```
public ReceivedHealInfo.ReceivedHealInfoStepBuilder WithIsCritical(bool isCritical)
```

Parameters

isCritical bool

Returns

[ReceivedHealInfo.ReceivedHealInfoStepBuilder](#)

WithSource(HealSource)

Set the [HealSource](#) for the heal.

```
public ReceivedHealInfo.HealInfoHealer WithSource(HealSource healSource)
```

Parameters

healSource [HealSource](#)

Returns

[ReceivedHealInfo.HealInfoHealer](#)

WithTarget(EntityCore)

Set the entity that will receive the heal. Returns the builder for optional flags.

```
public ReceivedHealInfo.ReceivedHealInfoStepBuilder WithTarget(EntityCore target)
```

Parameters

target EntityCore

Returns

ReceivedHealInfo.ReceivedHealInfoStepBuilder

Enum StepValuePoint

Namespace: [ElectricDrill.AstraRpgHealth.Heal](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Selects whether to use the pre-step or post-step value when [Step](#) is selected.

```
[Serializable]  
public enum StepValuePoint
```

Fields

Post = 1

Use the value recorded after the pipeline step executes.

Pre = 0

Use the value recorded before the pipeline step executes.

Namespace ElectricDrill.AstraRpgHealth. Resurrection

Classes

[DoNothingOnResurrectionStrategy](#)

A no-op resurrection strategy. Use this when no action is required upon an entity's resurrection. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Resurrection strategies / Do Nothing.

[EnableOnResurrectionStrategy](#)

A resurrection strategy that enables the entity's UnityEngine.GameObject when it is resurrected. This is useful when using a death strategy that disables the GameObject (e.g., DisableOnDeathStrategy). Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Resurrection strategies / Enable.

[MultipleOnResurrectionStrategy](#)

A resurrection strategy that executes multiple resurrection strategies in order. This allows combining multiple resurrection behaviors without creating new strategy classes. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Resurrection strategies / Multiple.

[OnResurrectionStrategy](#)

Base strategy type for handling an entity's resurrection. Derive from this class and implement [Resurrect\(EntityHealth\)](#) to define custom resurrection behavior.

[ResurrectionInfo](#)

Contains information about an entity resurrection event.

Interfaces

[IResurrectable](#)

Class DoNothingOnResurrectionStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Resurrection](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

A no-op resurrection strategy. Use this when no action is required upon an entity's resurrection. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Resurrection strategies / Do Nothing.

```
[CreateAssetMenu(fileName = "Do Nothing On Resurrection Strategy", menuName = "Astra RPG Health/Resurrection strategies/Do Nothing")]
public class DoNothingOnResurrectionStrategy : OnResurrectionStrategy
```

Inheritance

object ← Object ← ScriptableObject ← [OnResurrectionStrategy](#) ← DoNothingOnResurrectionStrategy

Methods

Resurrect(EntityHealth)

Intentionally performs no action when an entity is resurrected.

```
public override void Resurrect(EntityHealth entityHealth)
```

Parameters

`entityHealth` [EntityHealth](#)

The [EntityHealth](#) instance that is being resurrected (unused).

Class EnableOnResurrectionStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Resurrection](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

A resurrection strategy that enables the entity's UnityEngine.GameObject when it is resurrected. This is useful when using a death strategy that disables the GameObject (e.g., DisableOnDeathStrategy). Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Resurrection strategies / Enable.

```
[CreateAssetMenu(fileName = "Enable On Resurrection Strategy", menuName = "Astra RPG Health/Resurrection strategies/Enable")]
public class EnableOnResurrectionStrategy : OnResurrectionStrategy
```

Inheritance

object ← Object ← ScriptableObject ← [OnResurrectionStrategy](#) ← EnableOnResurrectionStrategy

Methods

Resurrect(EntityHealth)

Enables the UnityEngine.GameObject associated with `entityHealth`.

```
public override void Resurrect(EntityHealth entityHealth)
```

Parameters

`entityHealth` [EntityHealth](#)

The [EntityHealth](#) instance that is being resurrected.

Interface IResurrectable

Namespace: [ElectricDrill.AstraRpgHealth.Resurrection](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

```
public interface IResurrectable
```

Methods

Resurrect(Percentage, HealSource)

Resurrects the entity with a percentage of its maximum HP.

```
void Resurrect(Percentage withHpPercent, HealSource healSource)
```

Parameters

withHpPercent Percentage

The percentage of maximum HP to resurrect with.

healSource [HealSource](#)

The heal source associated with the resurrection.

Resurrect(long, HealSource)

Resurrects the entity with a fixed amount of HP.

```
void Resurrect(long withHp, HealSource healSource)
```

Parameters

withHp long

The fixed amount of HP to resurrect with.

healSource [HealSource](#)

The heal source associated with the resurrection.

Class MultipleOnResurrectionStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Resurrection](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

A resurrection strategy that executes multiple resurrection strategies in order. This allows combining multiple resurrection behaviors without creating new strategy classes. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Resurrection strategies / Multiple.

```
[CreateAssetMenu(fileName = "Multiple On Resurrection Strategy", menuName = "Astra RPG Health/Resurrection strategies/Multiple")]
public class MultipleOnResurrectionStrategy : OnResurrectionStrategy
```

Inheritance

object ← Object ← ScriptableObject ← [OnResurrectionStrategy](#) ← MultipleOnResurrectionStrategy

Methods

Resurrect(EntityHealth)

Executes all configured resurrection strategies in order. Null strategies in the list are skipped.

```
public override void Resurrect(EntityHealth entityHealth)
```

Parameters

entityHealth [EntityHealth](#)

The [EntityHealth](#) instance that is being resurrected.

Class OnResurrectionStrategy

Namespace: [ElectricDrill.AstraRpgHealth.Resurrection](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Base strategy type for handling an entity's resurrection. Derive from this class and implement [Resurrect\(EntityHealth\)](#) to define custom resurrection behavior.

```
public abstract class OnResurrectionStrategy : ScriptableObject
```

Inheritance

object ← Object ← ScriptableObject ← OnResurrectionStrategy

Derived

[DoNothingOnResurrectionStrategy](#), [EnableOnResurrectionStrategy](#), [MultipleOnResurrectionStrategy](#)

Methods

Resurrect(EntityHealth)

Invoked when an entity should be resurrected. Implementations should perform any setup, state changes, or initialization logic here.

```
public abstract void Resurrect(EntityHealth entityHealth)
```

Parameters

entityHealth [EntityHealth](#)

The [EntityHealth](#) instance representing the entity that is being resurrected.

Class ResurrectionInfo

Namespace: [ElectricDrill.AstraRpgHealth.Resurrection](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

Contains information about an entity resurrection event.

```
public class ResurrectionInfo
```

Inheritance

object ← ResurrectionInfo

Constructors

ResurrectionInfo(long, long, HealSource)

Initializes a new instance of the ResurrectionInfo class.

```
public ResurrectionInfo(long previousHp, long newHp, HealSource healSource)
```

Parameters

previousHp long

The HP before resurrection.

newHp long

The HP after resurrection.

healSource [HealSource](#)

The heal source associated with the resurrection.

Properties

HealSource

Gets the heal source associated with the resurrection. This can be used to categorize different types of resurrections (spell, item, phoenix down, etc.).

```
public HealSource HealSource { get; }
```

Property Value

[HealSource](#)

NewHp

Gets the health points the entity was resurrected with.

```
public long NewHp { get; }
```

Property Value

long

PreviousHp

Gets the health points the entity had before resurrection (typically at or below death threshold).

```
public long PreviousHp { get; }
```

Property Value

long

Namespace ElectricDrill.AstraRpgHealth. Scaling.Components

Classes

[HealthScalingComponent](#)

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

[HealthScalingComponentMenuItems](#)

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

Class HealthScalingComponent

Namespace: [ElectricDrill.AstraRpgHealth.Scaling.Components](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

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

```
public class HealthScalingComponent : ScalingComponent
```

Inheritance

object ← Object ← ScriptableObject ← ScalingComponent ← HealthScalingComponent

Methods

CalculateValue(EntityCore)

Calculates the scaling value based on the entity's health component.

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

Parameters

entity EntityCore

The entity to calculate the value for.

Returns

long

The calculated scaling value.

Class HealthScalingComponentMenuItems

Namespace: [ElectricDrill.AstraRpgHealth.Scaling.ScalingComponents](#)

Assembly: com.electricdrill.astra-rpg-health.Runtime.dll

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

```
public static class HealthScalingComponentMenuItems
```

Inheritance

object ← HealthScalingComponentMenuItems