

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

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

OverrideOnDeathGameAction

Gets or sets the override ElectricDrill.AstraRpgFramework.GameActions.GameAction<TContext> for handling entity death. This game action takes precedence over the default game action defined in the config. If null, the default game action from the [AstraRpgHealthConfigSO](#) configuration will be used.

```
public GameAction<Component> OverrideOnDeathGameAction { get; set; }
```

Property Value

GameAction<[Component](#)>

PassiveHealthRegeneration

Gets or sets whether passive health regeneration is enabled for this entity. When enabled, the entity will automatically regenerate health over time based on the passive health regeneration stat and the regeneration interval defined in the configuration.

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

Property Value

bool

Methods

AddExtraDamageResolutionEvent(DamageResolutionGameEvent)

Adds an extra DamageResolutionGameEvent to be raised when damage is resolved for this entity.

```
public void AddExtraDamageResolutionEvent(DamageResolutionGameEvent evt)
```

Parameters

evt [DamageResolutionGameEvent](#)

AddExtraEntityDiedEvent(EntityDiedGameEvent)

Adds an extra EntityDiedGameEvent to be raised when this entity dies.

```
public void AddExtraEntityDiedEvent(EntityDiedGameEvent evt)
```

Parameters

evt [EntityDiedGameEvent](#)

AddExtraEntityHealedEvent(EntityHealedGameEvent)

Adds an extra EntityHealedGameEvent to be raised when this entity is healed.

```
public void AddExtraEntityHealedEvent(EntityHealedGameEvent evt)
```

Parameters

evt [EntityHealedGameEvent](#)

AddExtraEntityResurrectedEvent(EntityResurrectedGameEvent)

Adds an extra EntityResurrectedGameEvent to be raised when this entity is resurrected.

```
public void AddExtraEntityResurrectedEvent(EntityResurrectedGameEvent evt)
```

Parameters

evt [EntityResurrectedGameEvent](#)

AddExtraGainedHealthEvent(EntityGainedHealthGameEvent)

Adds an extra EntityGainedHealthGameEvent to be raised when this entity gains health.

```
public void AddExtraGainedHealthEvent(EntityGainedHealthGameEvent evt)
```

Parameters

evt [EntityGainedHealthGameEvent](#)

AddExtraLostHealthEvent(EntityLostHealthGameEvent)

Adds an extra EntityLostHealthGameEvent to be raised when this entity loses health.

```
public void AddExtraLostHealthEvent(EntityLostHealthGameEvent evt)
```

Parameters

evt [EntityLostHealthGameEvent](#)

AddExtraMaxHealthChangedEvent(EntityMaxHealthChangedGameEvent)

Adds an extra EntityMaxHealthChangedGameEvent to be raised when this entity's max health changes.

```
public void AddExtraMaxHealthChangedEvent(EntityMaxHealthChangedGameEvent evt)
```

Parameters

evt [EntityMaxHealthChangedGameEvent](#)

AddExtraPreDamageInfoEvent(PreDamageGameEvent)

Adds an extra PreDamageGameEvent to be raised when this entity is about to take damage.

```
public void AddExtraPreDamageInfoEvent(PreDamageGameEvent evt)
```

Parameters

evt [PreDamageGameEvent](#)

AddExtraPreHealEvent(PreHealGameEvent)

Adds an extra PreHealGameEvent to be raised when this entity is about to be healed.

```
public void AddExtraPreHealEvent(PreHealGameEvent evt)
```

Parameters

evt [PreHealGameEvent](#)

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, DefenseReductionFnSO, long, Stat, DamageReductionFnSO, bool)`

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

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

Parameters

`amount` long

The base damage amount.

`piercingStatValue` long

The attacker's piercing stat value.

`defenseReductionFn` [DefenseReductionFnSO](#)

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` [DamageReductionFnSO](#)

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](#)

GetHpPortion(double)

Returns the amount of current HP represented by the given portion.

```
public long GetHpPortion(double portion)
```

Parameters

portion double

The portion of current HP (e.g., 0.2 means 20%).

Returns

long

GetMaxHpPortion(double)

Returns the amount of max HP represented by the given portion.

```
public long GetMaxHpPortion(double portion)
```

Parameters

portion double

The portion of max HP (e.g., 0.2 means 20%).

Returns

long

GetMissingHpPortion(double)

Returns the amount of missing HP represented by the given portion.

```
public long GetMissingHpPortion(double portion)
```

Parameters

portion double

The portion of missing HP (e.g., 0.2 means 20%).

Returns

long

Heal(PreHealContext)

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.

Heal Calculation:

1. Apply critical multiplier if the heal is flagged as critical.
2. Apply generic heal modifier stat (if configured).
3. Apply heal source-specific modifier stat (if configured and heal source matches).
4. Add the final calculated heal amount to current health (capped at max HP).

Heal modifiers stack additively (e.g., +25% generic heal + +10% source-specific = +35% total).

```
public ReceivedHealContext Heal(PreHealContext context)
```

Parameters

`context` [PreHealContext](#)

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

Returns

[ReceivedHealContext](#)

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 \leq 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, HealSourceSO)

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

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

Parameters

`withHpPercent` Percentage

The percentage of maximum health to restore on resurrection.

`healSource` [HealSourceSO](#)

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, HealSourceSO)`

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

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

Parameters

`withHp` long

The amount of health to restore on resurrection.

healSource [HealSourceSO](#)

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(PreDamageContext)

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 PreDamageContext 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 DamageResolutionContext.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 DamageResolutionContext
 - o Executes the on-death Game Action (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 DamageResolutionContext.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 DamageResolutionContext TakeDamage(PreDamageContext preDamage)
```

Parameters

preDamage [PreDamageContext](#)

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

[DamageResolutionContext](#)

A DamageResolutionContext 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

[AstraRpgHealthConfigProvider](#)

[AstraRpgHealthConfigSO](#)

[AstraRpgHealthGlobalSettingsSO](#)

Interfaces

[IAstraRpgHealthConfig](#)

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

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 AstraRpgHealthConfigSO

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

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

```
public class AstraRpgHealthConfigSO : ScriptableObject, IAstraRpgHealthConfig
```

Inheritance

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

Implements

[IAstraRpgHealthConfig](#)

Properties

DefaultDamageCalculationCalculationStrategy

Gets or sets the default strategy used for calculating net damage when none is explicitly specified in the target entity.

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

Property Value

[DamageCalculationStrategySO](#)

DefaultExpCollectionStrategy

Gets or sets the default strategy used for determining when and how experience is collected from kills. Individual ExpCollector components can override this with a custom strategy.

```
public ExpCollectionStrategySO DefaultExpCollectionStrategy { get; set; }
```

Property Value

[ExpCollectionStrategySO](#)

DefaultOnDeathGameAction

Gets or sets the default ElectricDrill.AstraRpgFramework.GameActions.GameAction<TContext> to execute when an entity dies, if no specific strategy is defined in the dying entity.

```
public GameAction<Component> DefaultOnDeathGameAction { get; set; }
```

Property Value

GameAction<[Component](#)>

DefaultOnResurrectionGameAction

Gets or sets the default ElectricDrill.AstraRpgFramework.GameActions.GameAction<TContext> to execute when an entity is resurrected, if no specific strategy is defined in the resurrected entity.

```
public GameAction<Component> DefaultOnResurrectionGameAction { get; set; }
```

Property Value

GameAction<[Component](#)>

DefaultResurrectionSource

Gets or sets the heal source used for resurrections heals.

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

Property Value

[HealSourceSO](#)

GenericFlatDamageModificationStat

Gets or sets the stat that provides flat damage modification, regardless of type or source. Applied after percentage modifications. Stacks additively with other flat modifications.

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

Property Value

Stat

GenericHealAmountModifierStat

Gets or sets the stat that modifies the amount of health healed. The stat value must be a percentage (e.g., 25 for +25% healing).

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

Property Value

Stat

GenericPercentageDamageModificationStat

Gets or sets the stat that modifies any damage received with percentage, regardless of type or source. Stacks additively with other percentage damage modifications.

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

Property Value

Stat

HealSourceModifications

Gets or sets the dictionary mapping heal sources to their corresponding heal modifier stats. Stacks additively with other heal modifications.

```
public SerializableDictionary<HealSourceSO, Stat> HealSourceModifications { get; set; }
```

Property Value

SerializableDictionary<[HealSourceSO](#), Stat>

HealthAttributesScaling

Gets or sets the attributes scaling component that determines how health scales based on attributes.

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

Property Value

AttributesScalingComponent

HealthRegenerationSource

Gets or sets the heal source used for health regeneration effects.

```
public HealSourceSO HealthRegenerationSource { get; set; }
```

Property Value

[HealSourceSO](#)

LifestealConfig

Gets or sets the configuration for lifesteal mechanics.

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

Property Value

[LifestealConfigSO](#)

ManualHealthRegenerationStat

Gets or sets the stat that determines manual health regeneration amount. Use the API to trigger manual regeneration ticks programmatically.

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

Property Value

Stat

PassiveHealthRegenerationInterval

Gets or sets the interval (in seconds) between passive health regeneration ticks. Regenerated health is calculated based on the Passive Health Regeneration Stat.

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

Property Value

float

PassiveHealthRegenerationStat

Gets or sets the stat that determines passive health regeneration amount. Stat value represents health regenerated per 10 seconds.

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

Property Value

Stat

SUPPRESSLIFESTEALEVENTS

Gets or sets whether to suppress PreHeal and ReceivedHeal events for lifesteal healing. When true, lifesteal will not raise heal events, improving performance when many entities deal repeatedly damage with lifesteal. Default is false (events are raised).

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

Property Value

bool

SuppressManualRegenerationEvents

Gets or sets whether to suppress PreHeal and ReceivedHeal events for manual health regeneration. When true, manual regeneration will not raise heal events, improving performance in case of frequent calls. Default is false (events are raised).

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

Property Value

bool

SuppressPassiveRegenerationEvents

Gets or sets whether to suppress PreHeal and ReceivedHeal events for passive health regeneration. When true, passive regeneration will not raise heal events, improving performance for frequent regeneration ticks. Default is false (events are raised).

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

Property Value

bool

Class AstraRpgHealthGlobalSettingsSO

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

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

```
public class AstraRpgHealthGlobalSettingsSO : ScriptableObject
```

Inheritance

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

Fields

AssetPath

Full asset path for the global settings asset (used in Editor)

```
public const string AssetPath = "Assets/Resources/AstraRpgHealthGlobalSettings.asset"
```

Field Value

string

DefaultConfigResourceName

Resource name for the default/fallback health config

```
public const string DefaultConfigResourceName = "Astra Rpg Health Config"
```

Field Value

string

ResourcePath

Resource path (without extension) for the global settings asset

```
public const string ResourcePath = "AstraRpgHealthGlobalSettings"
```

Field Value

string

Properties

ActiveConfig

Gets or sets the active Astra RPG Health configuration profile used by the system.

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

Property Value

[AstraRpgHealthConfigSO](#)

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

DefaultDamageCalculationCalculationStrategy

Gets or sets the default strategy used for calculating net damage when none is explicitly specified in the target entity.

```
DamageCalculationStrategySO DefaultDamageCalculationCalculationStrategy { get; set; }
```

Property Value

[DamageCalculationStrategySO](#)

DefaultExpCollectionStrategy

Gets or sets the default strategy used for determining when and how experience is collected from kills. Individual ExpCollector components can override this with a custom strategy.

```
ExpCollectionStrategySO DefaultExpCollectionStrategy { get; set; }
```

Property Value

[ExpCollectionStrategySO](#)

DefaultOnDeathGameAction

Gets or sets the default ElectricDrill.AstraRpgFramework.GameActions.GameAction<TContext> to execute when an entity dies, if no specific strategy is defined in the dying entity.

```
GameAction<Component> DefaultOnDeathGameAction { get; set; }
```

Property Value

GameAction<[Component](#)>

DefaultOnResurrectionGameAction

Gets or sets the default ElectricDrill.AstraRpgFramework.GameActions.GameAction<TContext> to execute when an entity is resurrected, if no specific strategy is defined in the resurrected entity.

```
GameAction<Component> DefaultOnResurrectionGameAction { get; set; }
```

Property Value

GameAction<[Component](#)>

DefaultResurrectionSource

Gets or sets the heal source used for resurrections heals.

```
HealSourceSO DefaultResurrectionSource { get; set; }
```

Property Value

[HealSourceSO](#)

GenericFlatDamageModificationStat

Gets or sets the stat that provides flat damage modification, regardless of type or source. Applied after percentage modifications. Stacks additively with other flat modifications.

```
Stat GenericFlatDamageModificationStat { get; set; }
```

Property Value

Stat

GenericHealAmountModifierStat

Gets or sets the stat that modifies the amount of health healed. The stat value must be a percentage (e.g., 25 for +25% healing).

```
Stat GenericHealAmountModifierStat { get; set; }
```

Property Value

Stat

GenericPercentageDamageModificationStat

Gets or sets the stat that modifies any damage received with percentage, regardless of type or source. Stacks additively with other percentage damage modifications.

```
Stat GenericPercentageDamageModificationStat { get; set; }
```

Property Value

Stat

HealSourceModifications

Gets or sets the dictionary mapping heal sources to their corresponding heal modifier stats. Stacks additively with other heal modifications.

```
SerializableDictionary<HealSourceSO, Stat> HealSourceModifications { get; set; }
```

Property Value

SerializableDictionary<[HealSourceSO](#), Stat>

HealthAttributesScaling

Gets or sets the attributes scaling component that determines how health scales based on attributes.

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

Property Value

AttributesScalingComponent

HealthRegenerationSource

Gets or sets the heal source used for health regeneration effects.

```
HealSourceSO HealthRegenerationSource { get; set; }
```

Property Value

[HealSourceSO](#)

LifestealConfig

Gets or sets the configuration for lifesteal mechanics.

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

Property Value

[LifestealConfigSO](#)

ManualHealthRegenerationStat

Gets or sets the stat that determines manual health regeneration amount. Use the API to trigger manual regeneration ticks programmatically. Useful for turn-based systems.

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

Property Value

Stat

PassiveHealthRegenerationInterval

Gets or sets the interval (in seconds) between passive health regeneration ticks. Regenerated health is calculated based on the Passive Health Regeneration Stat.

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

Property Value

float

PassiveHealthRegenerationStat

Gets or sets the stat that determines passive health regeneration amount. Stat value represents health regenerated per 10 seconds.

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

Property Value

Stat

SUPPRESSLIFESTEALEVENTS

Gets or sets whether to suppress PreHeal and ReceivedHeal events for lifesteal healing. When true, lifesteal will not raise heal events, improving performance when many entities deal repeatedly damage with lifesteal. Default is false (events are raised).

```
bool SuppressLifestealEvents { get; set; }
```

Property Value

bool

SuppressManualRegenerationEvents

Gets or sets whether to suppress PreHeal and ReceivedHeal events for manual health regeneration. When true, manual regeneration will not raise heal events, improving performance in case of frequent calls. Default is false (events are raised).

```
bool SuppressManualRegenerationEvents { get; set; }
```

Property Value

bool

SuppressPassiveRegenerationEvents

Gets or sets whether to suppress PreHeal and ReceivedHeal events for passive health regeneration. When true, passive regeneration will not raise heal events, improving performance for frequent regeneration ticks. Default is false (events are raised).

```
bool SuppressPassiveRegenerationEvents { get; set; }
```

Property Value

bool

Namespace ElectricDrill.AstraRpgHealth.

Damage

Classes

[DamageAmountContext](#)

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](#).

[DamageResolutionContext](#)

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).

[DamageSourceSO](#)

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.

[DamageTypeSO](#)

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.

[PreDamageContext](#)

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

[PreDamageContext.PreDamageInfoStepBuilder](#)

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

Structs

[DamageAmountContext.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\(PreDamageContext\)](#) method accepts a [PreDamageContext](#) record describing a pending damage attempt and returns a [DamageResolutionContext](#) describing whether the damage was applied or prevented and additional details.

[PreDamageContext.DamageInfoAmount](#)

Fluent builder step: specify the damage amount.

[PreDamageContext.DamageInfoDealer](#)

Fluent builder step: specify the dealer entity.

[PreDamageContext.DamageInfoSource](#)

Fluent builder step: specify the damage source.

[PreDamageContext.DamageInfoTarget](#)

Fluent builder step: specify the target entity.

[PreDamageContext.DamageInfoType](#)

Fluent builder step: specify the damage type.

Enums

[DamageOutcome](#)

The overall result of attempting to apply damage to a target. Use this in [DamageResolutionContext](#) 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 DamageAmountContext

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 DamageAmountContext
```

Inheritance

object ← DamageAmountContext

Remarks

This class is mutable for [Current](#) and keeps a list of [DamageAmountContext.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

DamageAmountContext(long)

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

```
public DamageAmountContext(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<DamageAmountContext.StepAmountRecord> Records { get; }
```

Property Value

IReadOnlyList<[DamageAmountContext.StepAmountRecord](#)>

Methods

FromRaw(long)

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

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

Parameters

raw long

Raw damage value.

Returns

[DamageAmountContext](#)

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

GetStepAmount(Type)

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

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

Parameters

stepType Type

The phase type to look up.

Returns

[DamageAmountContext.StepAmountRecord](#)

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

Remarks

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

Struct

DamageAmountContext.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 DamageAmountContext.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 [DamageResolutionContext](#) 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 [DamageSourceSO](#) that originated the damage.

DamageTypeImmune = 4

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

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 PreDamageContext 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 DamageResolutionContext

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 DamageResolutionContext
```

Inheritance

object ← DamageResolutionContext

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](#)

PreDamageContext

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

```
public PreDamageContext PreDamageContext { get; }
```

Property Value

[PreDamageContext](#)

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, PreDamageContext)

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

```
public static DamageResolutionContext Applied(DamageInfo info, PreDamageContext pre)
```

Parameters

`info` [DamageInfo](#)

Final damage info that was applied.

`pre` [PreDamageContext](#)

Original pre-damage request.

Returns

[DamageResolutionContext](#)

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

Prevented(DamagePreventionReason, PreDamageContext, DamageInfo)

Create a [DamageResolutionContext](#) 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 DamageResolutionContext Prevented(DamagePreventionReason reasons,  
PreDamageContext pre, DamageInfo info = null)
```

Parameters

`reasons` [DamagePreventionReason](#)

Flags explaining the prevention.

`pre` [PreDamageContext](#)

Original pre-damage request.

`info` [DamageInfo](#)

Optional partial damage info produced before prevention.

Returns

[DamageResolutionContext](#)

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

Class DamageSourceSO

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 DamageSourceSO : ScriptableObject
```

Inheritance

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

Properties

FlatDamageModificationStat

The stat representing the flat damage accumulator for this source. This stat serves as an accumulator for multiple flat modifiers (positive or negative) that affect damage from this source linearly.

```
public Stat FlatDamageModificationStat { get; }
```

Property Value

Stat

PercentageDamageModificationStat

The stat representing the percentage damage accumulator for this source. This stat serves as an accumulator for multiple percentage modifiers (positive or negative) that affect damage from this source.

```
public Stat PercentageDamageModificationStat { get; }
```

Property Value

Stat

Methods

ToString()

Returns the asset name of this damage source.

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

Returns

string

Class DamageTypeSO

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 DamageTypeSO : ScriptableObject
```

Inheritance

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

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 DamageReductionFnSO DamageReductionFn { get; }
```

Property Value

[DamageReductionFnSO](#)

DefenseReductionFn

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

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

Property Value

[DefenseReductionFnSO](#)

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

FlatDamageModificationStat

The stat representing the flat damage accumulator for this type. This stat serves as an accumulator for multiple flat modifiers (positive or negative) that affect damage of this type linearly.

```
public Stat FlatDamageModificationStat { get; }
```

Property Value

Stat

IgnoreGenericFlatDamageModifiers

If true, this damage type ignores generic flat damage modifiers. Useful for "true damage" mechanics that bypass flat reductions/increments.

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

Property Value

bool

IgnoreGenericPercentageDamageModifiers

If true, this damage type ignores generic percentage damage modifiers. Useful for "true damage" mechanics that bypass resistances/weaknesses.

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

Property Value

bool

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

PercentageDamageModificationStat

The stat representing the percentage damage accumulator for this type. This stat serves as an accumulator for multiple percentage modifiers (positive or negative) that affect damage of this type.

```
public Stat PercentageDamageModificationStat { get; }
```

Property Value

Stat

ReducedBy

The stat that reduces the raw damage amount for this [DamageTypeSO](#). 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\(PreDamageContext\)](#) method accepts a [PreDamageContext](#) record describing a pending damage attempt and returns a [DamageResolutionContext](#) describing whether the damage was applied or prevented and additional details.

```
public interface IDamageable
```

Methods

TakeDamage(PreDamageContext)

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 [DamageResolutionContext](#) must reflect the final outcome:

- If damage was applied, an instance created with [Applied\(DamageInfo, PreDamageContext\)](#) 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\(DamagePreventionReason, PreDamageContext, DamageInfo\)](#) 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.

```
DamageResolutionContext TakeDamage(PreDamageContext preDamage)
```

Parameters

`preDamage` [PreDamageContext](#)

Immutable description of the pending damage attempt.

Returns

[DamageResolutionContext](#)

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

Class PreDamageContext

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 PreDamageContext : IHasTarget, IHasSource
```

Inheritance

object ← PreDamageContext

Implements

IHasTarget, IHasSource

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 = PreDamageContext.Builder
    .WithAmount(10)
    .WithType(DamageType.Physical)
    .WithSource(DamageSource.Weapon)
    .WithTarget(targetEntity)
```

```
.WithDealer(dealerEntity)
.WithIsCritical(false)
.Build();

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

Property Value

[PreDamageContext.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 DamageSourceSO DamageSource { get; }
```

Property Value

[DamageSourceSO](#)

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

Source

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

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

Property Value

EntityCore

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 DamageTypeSO Type { get; set; }
```

Property Value

[DamageTypeSO](#)

Interface PreDamageContext.DamageInfoAmount

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

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

Fluent builder step: specify the damage amount.

```
public interface PreDamageContext.DamageInfoAmount
```

Methods

WithAmount(long)

Set the damage amount and advance to the next step.

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

Parameters

amount long

Returns

[PreDamageContext.DamageInfoType](#)

Interface PreDamageContext.DamageInfoDealer

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

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

Fluent builder step: specify the dealer entity.

```
public interface PreDamageContext.DamageInfoDealer
```

Methods

WithDealer(EntityCore)

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

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

Parameters

dealer EntityCore

Returns

[PreDamageContext.PreDamageInfoStepBuilder](#)

Interface PreDamageContext.DamageInfoSource

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

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

Fluent builder step: specify the damage source.

```
public interface PreDamageContext.DamageInfoSource
```

Methods

WithSource(DamageSourceSO)

Set the damage source and advance to the next step.

```
PreDamageContext.DamageInfoTarget WithSource(DamageSourceSO damageSource)
```

Parameters

damageSource [DamageSourceSO](#)

Returns

[PreDamageContext.DamageInfoTarget](#)

Interface PreDamageContext.DamageInfoTarget

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

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

Fluent builder step: specify the target entity.

```
public interface PreDamageContext.DamageInfoTarget
```

Methods

WithTarget(EntityCore)

Set the damage target and advance to the next step.

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

Parameters

target EntityCore

Returns

[PreDamageContext.DamageInfoDealer](#)

Interface PreDamageContext.DamageInfoType

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

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

Fluent builder step: specify the damage type.

```
public interface PreDamageContext.DamageInfoType
```

Methods

WithType(DamageTypeSO)

Set the damage type and advance to the next step.

```
PreDamageContext.DamageInfoSource WithType(DamageTypeSO type)
```

Parameters

type [DamageTypeSO](#)

Returns

[PreDamageContext.DamageInfoSource](#)

Class

PreDamageContext.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 PreDamageContext.PreDamageInfoStepBuilder :  
    PreDamageContext.DamageInfoAmount, PreDamageContext.DamageInfoType,  
    PreDamageContext.DamageInfoSource, PreDamageContext.DamageInfoTarget,  
    PreDamageContext.DamageInfoDealer
```

Inheritance

object ← PreDamageContext.PreDamageInfoStepBuilder

Implements

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

Methods

Build()

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

```
public PreDamageContext Build()
```

Returns

[PreDamageContext](#)

WithAmount(long)

Set the damage amount.

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

Parameters

amount long

Returns

[PreDamageContext.DamageInfoType](#)

WithCriticalMultiplier(double)

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

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

Parameters

multiplier double

Returns

[PreDamageContext.PreDamageInfoStepBuilder](#)

WithDealer(EntityCore)

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

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

Parameters

dealer EntityCore

Returns

[PreDamageContext.PreDamageInfoStepBuilder](#)

WithIsCritical(bool)

Mark this damage as critical or not.

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

Parameters

`isCritical` bool

Returns

[PreDamageContext.PreDamageInfoStepBuilder](#)

WithSource(DamageSourceSO)

Set the damage source.

```
public PreDamageContext.DamageInfoTarget WithSource(DamageSourceSO damageSource)
```

Parameters

`damageSource` [DamageSourceSO](#)

Returns

[PreDamageContext.DamageInfoTarget](#)

WithTarget(EntityCore)

Set the target entity that will receive the damage.

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

Parameters

target EntityCore

Returns

[PreDamageContext.DamageInfoDealer](#)

WithType(DamageTypeSO)

Set the damage type.

```
public PreDamageContext.DamageInfoSource WithType(DamageTypeSO type)
```

Parameters

type [DamageTypeSO](#)

Returns

[PreDamageContext.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.

[ApplyFlatDmgModifiersStep](#)

Damage step that applies flat damage modifiers (both positive increments and negative reductions) based on configured generic flat damage modifiers & source/type flat damage modifiers. This step should typically be applied AFTER percentage-based modifications.

[ApplyPercentageDmgModifiersStep](#)

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

[DamageCalculationStrategySO](#)

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.

```
[Serializable]  
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.

```
[Serializable]  
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.

```
[Serializable]  
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 [DamageInfo](#)

Current damage pipeline data.

Returns

[DamageInfo](#)

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

Class ApplyFlatDmgModifiersStep

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

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

Damage step that applies flat damage modifiers (both positive increments and negative reductions) based on configured generic flat damage modifiers & source/type flat damage modifiers. This step should typically be applied AFTER percentage-based modifications.

```
[Serializable]  
public class ApplyFlatDmgModifiersStep : DamageStep
```

Inheritance

object ← [DamageStep](#) ← ApplyFlatDmgModifiersStep

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 flat modifiers to [Amounts](#). Flat modifiers are summed additively and applied to the current damage amount. The final damage cannot go below 0 (negative results are clamped to 0).

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

Parameters

data [DamageInfo](#)

Current damage pipeline data.

Returns

[DamageInfo](#)

The same **data** instance updated with the modified amounts.

Class ApplyPercentageDmgModifiersStep

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.

```
[Serializable]  
public class ApplyPercentageDmgModifiersStep : DamageStep
```

Inheritance

object ← [DamageStep](#) ← ApplyPercentageDmgModifiersStep

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 DamageCalculationStrategySO

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 Damage Calculation Strategy", menuName = "Astra RPG Health/Damage Calculation Pipeline/Configurable Strategy")]
public class DamageCalculationStrategySO : ScriptableObject
```

Inheritance

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

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 : IHasSource, IHasTarget
```

Inheritance

object ← DamageInfo

Implements

IHasSource, IHasTarget

Constructors

DamageInfo(PreDamageContext)

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

```
public DamageInfo(PreDamageContext pre)
```

Parameters

`pre` [PreDamageContext](#)

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

Properties

Amounts

Numeric amounts and intermediate records for the damage being processed.

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

Property Value

[DamageAmountContext](#)

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 DamageSourceSO DamageSource { get; }
```

Property Value

[DamageSourceSO](#)

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](#)

Source

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

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

Property Value

EntityCore

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 DamageTypeSO Type { get; }
```

Property Value

[DamageTypeSO](#)

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](#), [ApplyFlatDmgModifiersStep](#),
[ApplyPercentageDmgModifiersStep](#)

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

[DamageReductionFnSO](#)

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

[FlatDamageReductionFnSO](#)

[LogDamageReductionFnSO](#)

[PercentageDamageReductionFnSO](#)

Class DamageReductionFnSO

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 DamageReductionFnSO : ScriptableObject
```

Inheritance

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

Derived

[FlatDamageReductionFnSO](#), [LogDamageReductionFnSO](#), [PercentageDamageReductionFnSO](#)

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.

Class FlatDamageReductionFnSO

Namespace: [ElectricDrill.AstraRpgHealth.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 FlatDamageReductionFnSO : DamageReductionFnSO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [DamageReductionFnSO](#) ← FlatDamageReductionFnSO

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.FlatDamageReductionFnSO._factor` before being subtracted.

Returns

long

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

Class LogDamageReductionFnSO

Namespace: [ElectricDrill.AstraRpgHealth.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 LogDamageReductionFnSO : DamageReductionFnSO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [DamageReductionFnSO](#) ← LogDamageReductionFnSO

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 PercentageDamageReductionFnSO

Namespace: [ElectricDrill.AstraRpgHealth.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 PercentageDamageReductionFnSO : DamageReductionFnSO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [DamageReductionFnSO](#) ← PercentageDamageReductionFnSO

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. DefenseReductionFunctions

Classes

[DefenseReductionFnSO](#)

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

[FlatDefenseReductionFnSO](#)

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.

[LogDefenseReductionFnSO](#)

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.

[PercentageDefenseReductionFnSO](#)

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 DefenseReductionFnSO

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 DefenseReductionFnSO : ScriptableObject
```

Inheritance

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

Derived

[FlatDefenseReductionFnSO](#), [LogDefenseReductionFnSO](#), [PercentageDefenseReductionFnSO](#)

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).

Class FlatDefenseReductionFnSO

Namespace: [ElectricDrill.AstraRpgHealth.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 FlatDefenseReductionFnSO : DefenseReductionFnSO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [DefenseReductionFnSO](#) ← FlatDefenseReductionFnSO

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 LogDefenseReductionFnSO

Namespace: [ElectricDrill.AstraRpgHealth.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 LogDefenseReductionFnSO : DefenseReductionFnSO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [DefenseReductionFnSO](#) ← LogDefenseReductionFnSO

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 PercentageDefenseReductionFnSO

Namespace: [ElectricDrill.AstraRpgHealth.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 PercentageDefenseReductionFnSO : DefenseReductionFnSO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [DefenseReductionFnSO](#) ← PercentageDefenseReductionFnSO

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](#)

Event raised to notify the resolution of a damage attempt (applied or prevented). Payload: [DamageResolutionContext](#).

[DamageResolutionGameEventListener](#)

Used to notify whether a damage was applied or prevented.

[EntityDiedContext](#)

[EntityDiedGameEvent](#)

Event raised when an entity dies. Payload: [EntityDiedContext](#).

[EntityDiedGameEventListener](#)

Event raised when an entity dies. Payload: [EntityDiedContext](#).

[EntityGainedHealthGameEvent](#)

Event raised when an entity gains health. Payload: [EntityHealthChangedContext](#).

[EntityGainedHealthGameEventListener](#)

Event raised when an entity gains health. Payload: [EntityHealthChangedContext](#).

[EntityHealedGameEvent](#)

Event raised when an entity is about to be healed. Payload: [PreHealContext](#).

[EntityHealedGameEventListener](#)

Event raised when an entity is about to be healed. Payload: [PreHealContext](#).

[EntityHealthChangedContext](#)

[EntityLostHealthGameEvent](#)

Event raised when an entity loses health. Payload: [EntityHealthChangedContext](#).

[EntityLostHealthGameEventListener](#)

Event raised when an entity loses health. Payload: [EntityHealthChangedContext](#).

[EntityMaxHealthChangedContext](#)

[EntityMaxHealthChangedGameEvent](#)

Event raised when an entity's maximum health changes. Payload: [EntityMaxHealthChangedContext](#).

[EntityMaxHealthChangedGameEventListener](#)

Event raised when an entity's maximum health changes. Payload: [EntityMaxHealthChangedContext](#).

[EntityResurrectedGameEvent](#)

Event raised when an entity is resurrected. Payload: [ResurrectionContext](#).

[EntityResurrectedGameEventListener](#)

Event raised when an entity is resurrected. Payload: [ResurrectionContext](#).

[PreDamageGameEvent](#)

Event raised before damage is processed. Payload: [PreDamageContext](#).

[PreDamageGameEventListener](#)

[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

Event raised to notify the resolution of a damage attempt (applied or prevented). Payload: [DamageResolutionContext](#).

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

Inheritance

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

Implements

```
IRaisable<DamageResolutionContext>
```

Inherited Members

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

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<DamageResolutionContext>
```

Inheritance

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

Inherited Members

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

Class EntityDiedContext

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

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

```
public class EntityDiedContext : IHasVictim, IHasTarget, IHasSource
```

Inheritance

object ← EntityDiedContext

Implements

IHasVictim, IHasTarget, IHasSource

Constructors

EntityDiedContext(EntityCore, EntityCore,
DamageResolutionContext)

```
public EntityDiedContext(EntityCore victim, EntityCore source, DamageResolutionContext  
damageResolutionContext)
```

Parameters

victim EntityCore

source EntityCore

damageResolutionContext [DamageResolutionContext](#)

Properties

DamageResolutionContext

```
public DamageResolutionContext DamageResolutionContext { get; }
```

Property Value

[DamageResolutionContext](#)

Source

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

Property Value

EntityCore

Target

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

Property Value

EntityCore

Victim

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

Property Value

EntityCore

Class EntityDiedGameEvent

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

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

Event raised when an entity dies. Payload: [EntityDiedContext](#).

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

Inheritance

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

Implements

IRaisable<[EntityDiedContext](#)>

Inherited Members

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

Class EntityDiedGameEventListener

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

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

Event raised when an entity dies. Payload: [EntityDiedContext](#).

```
public class EntityDiedGameEventListener : GameEventListenerGeneric1<EntityDiedContext>
```

Inheritance

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

Inherited Members

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

Class EntityGainedHealthGameEvent

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

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

Event raised when an entity gains health. Payload: [EntityHealthChangedEventArgs](#).

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

Inheritance

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

Implements

IRaisable<[EntityHealthChangedEventArgs](#)>

Inherited Members

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

Class EntityGainedHealthGameEventListener

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

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

Event raised when an entity gains health. Payload: [EntityHealthChangedEventArgs](#).

```
public class EntityGainedHealthGameEventListener :  
GameEventListenerGeneric1<EntityHealthChangedEventArgs>
```

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<EntityHealthChangedEventArgs> ← EntityGainedHealthGameEventListener
```

Inherited Members

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

Class EntityHealedGameEvent

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

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

Event raised when an entity is about to be healed. Payload: [PreHealContext](#).

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

Inheritance

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

Implements

IRaisable<[ReceivedHealContext](#)>

Inherited Members

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

Class EntityHealedGameEventListener

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

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

Event raised when an entity is about to be healed. Payload: [PreHealContext](#).

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

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<ReceivedHealContext> ← EntityHealedGameEventListener
```

Inherited Members

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

Class EntityHealthChangedEventArgs

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

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

```
public class EntityHealthChangedEventArgs : IHasTarget, IHasValueChange<long>
```

Inheritance

object ← EntityHealthChangedEventArgs

Implements

IHasTarget, IHasValueChange<long>

Constructors

EntityHealthChangedEventArgs(EntityCore, long, long)

```
public EntityHealthChangedEventArgs(EntityCore target, long previousValue, long newValue)
```

Parameters

target EntityCore

previousValue long

newValue long

Properties

AbsAmount

The absolute amount of change between PreviousValue and NewValue.

```
public long AbsAmount { get; }
```

Property Value

long

NewValue

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

Property Value

long

PreviousValue

```
public long PreviousValue { get; }
```

Property Value

long

Target

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

Property Value

EntityCore

Class EntityLostHealthGameEvent

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

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

Event raised when an entity loses health. Payload: [EntityHealthChangedEventArgs](#).

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

Inheritance

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

Implements

IRaisable<[EntityHealthChangedEventArgs](#)>

Inherited Members

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

Class EntityLostHealthGameEventListener

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

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

Event raised when an entity loses health. Payload: [EntityHealthChangedEventArgs](#).

```
public class EntityLostHealthGameEventListener :  
GameEventListenerGeneric1<EntityHealthChangedEventArgs>
```

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<EntityHealthChangedEventArgs> ← EntityLostHealthGameEventListener
```

Inherited Members

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

Class EntityMaxHealthChangedEventArgs

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

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

```
public class EntityMaxHealthChangedEventArgs : IHasTarget, IHasValueChange<long>
```

Inheritance

object ← EntityMaxHealthChangedEventArgs

Implements

IHasTarget, IHasValueChange<long>

Constructors

EntityMaxHealthChangedEventArgs(EntityCore, long, long)

```
public EntityMaxHealthChangedEventArgs(EntityCore target, long previousValue, long newValue)
```

Parameters

target EntityCore

previousValue long

newValue long

Properties

AbsAmount

The absolute amount of change between PreviousValue and NewValue.

```
public long AbsAmount { get; }
```

Property Value

long

NewValue

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

Property Value

long

PreviousValue

```
public long PreviousValue { get; }
```

Property Value

long

Target

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

Property Value

EntityCore

Class EntityMaxHealthChangedGameEvent

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

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

Event raised when an entity's maximum health changes. Payload: [EntityMaxHealthChangedEventArgs](#).

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

Inheritance

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

Implements

IRaisable<[EntityMaxHealthChangedEventArgs](#)>

Inherited Members

GameEventGeneric1<EntityMaxHealthChangedEventArgs>.OnEventRaised ,
GameEventGeneric1<EntityMaxHealthChangedEventArgs>.Raise(EntityMaxHealthChangedEventArgs) ,
GameEventGeneric1<EntityMaxHealthChangedEventArgs>.RegisterListener(GameEventListenerGeneric1<E
ntityMaxHealthChangedEventArgs>) ,
GameEventGeneric1<EntityMaxHealthChangedEventArgs>.UnregisterListener(GameEventListenerGeneric1
<EntityMaxHealthChangedEventArgs>)

Class

EntityMaxHealthChangedGameEventListener

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

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

Event raised when an entity's maximum health changes. Payload: [EntityMaxHealthChangedEventArgs](#).

```
public class EntityMaxHealthChangedGameEventListener :  
GameEventListenerGeneric1<EntityMaxHealthChangedEventArgs>
```

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<EntityMaxHealthChangedEventArgs> ←  
EntityMaxHealthChangedGameEventListener
```

Inherited Members

```
GameEventListenerGeneric1<EntityMaxHealthChangedEventArgs>._event ,  
GameEventListenerGeneric1<EntityMaxHealthChangedEventArgs>._response ,  
GameEventListenerGeneric1<EntityMaxHealthChangedEventArgs>.OnEventRaised(EntityMaxHealthChange  
dContext)
```

Class EntityResurrectedGameEvent

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

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

Event raised when an entity is resurrected. Payload: [ResurrectionContext](#).

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

Inheritance

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

Implements

IRaisable<[ResurrectionContext](#)>

Inherited Members

GameEventGeneric1<ResurrectionContext>.OnEventRaised ,
GameEventGeneric1<ResurrectionContext>.Raise(ResurrectionContext) ,
GameEventGeneric1<ResurrectionContext>.RegisterListener(GameEventListenerGeneric1<ResurrectionC
ontext>) ,
GameEventGeneric1<ResurrectionContext>.UnregisterListener(GameEventListenerGeneric1<Resurre
ction Context>)

Class EntityResurrectedGameEventListener

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

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

Event raised when an entity is resurrected. Payload: [ResurrectionContext](#).

```
public class EntityResurrectedGameEventListener :  
GameEventListenerGeneric1<ResurrectionContext>
```

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<ResurrectionContext> ← EntityResurrectedGameEventListener
```

Inherited Members

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

Class PreDamageGameEvent

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

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

Event raised before damage is processed. Payload: [PreDamageContext](#).

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

Inheritance

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

Implements

IRaisable<[PreDamageContext](#)>

Inherited Members

GameEventGeneric1<PreDamageContext>.OnEventRaised ,
GameEventGeneric1<PreDamageContext>.Raise(PreDamageContext) ,
GameEventGeneric1<PreDamageContext>.RegisterListener(GameEventListenerGeneric1<PreDamageCon
text>) ,
GameEventGeneric1<PreDamageContext>.UnregisterListener(GameEventListenerGeneric1<PreDamageC
ontext>)

Class PreDamageGameEventListener

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

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

```
public class PreDamageGameEventListener : GameEventListenerGeneric1<PreDamageContext>
```

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<PreDamageContext> ← PreDamageGameEventListener
```

Inherited Members

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

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 : GameEventGeneric1<PreHealContext>, IRaisable<PreHealContext>
```

Inheritance

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

Implements

[IRaisable<PreHealContext>](#)

Inherited Members

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

Class PreHealGameEventListener

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

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

The entity is about to be healed.

```
public class PreHealGameEventListener : GameEventListenerGeneric1<PreHealContext>
```

Inheritance

```
object ← Object ← Component ← Behaviour ← MonoBehaviour ←  
GameEventListenerGeneric1<PreHealContext> ← PreHealGameEventListener
```

Inherited Members

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

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

[ExpCollectionStrategySO](#)

Abstract base class for experience collection strategies. Implements the Template Method pattern to define when and how experience should be collected. Derive from this class to create custom experience collection logic.

[ExpCollector](#)

Collects experience when this entity deals killing blows. Requires an [EntityDiedGameEventListener](#) on the GameObject. Uses a configurable [ExpCollectionStrategySO](#) to determine when and how experience is collected. If no custom strategy is assigned, falls back to the default strategy from [IAstraRpgHealthConfig](#).

Class ExpCollectionStrategySO

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

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

Abstract base class for experience collection strategies. Implements the Template Method pattern to define when and how experience should be collected. Derive from this class to create custom experience collection logic.

```
public abstract class ExpCollectionStrategySO : ScriptableObject
```

Inheritance

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

Derived

[DamageSourceKillExpStrategySO](#), [DirectKillExpStrategySO](#), [FirstMatchExpStrategySO](#)

Methods

CalculateExpAmount(IExpSource, float)

Calculates the final experience amount to collect based on the source and multiplier. Override this method to customize experience calculation logic.

```
protected virtual long CalculateExpAmount(IExpSource expSource, float multiplier)
```

Parameters

expSource IExpSource

The experience source providing the base amount.

multiplier float

The multiplier to apply to the base experience.

Returns

long

The final experience amount to add.

CollectExp(EntityCore, IExpSource, float)

Performs the actual experience collection: calculates the amount, adds it to the collector's level, and marks the source as harvested. Override this method to customize the entire collection process (e.g., for party XP sharing).

```
protected virtual bool CollectExp(EntityCore collector, IExpSource expSource,  
float multiplier)
```

Parameters

collector EntityCore

The entity collecting the experience.

expSource IExpSource

The experience source to collect from.

multiplier float

The multiplier to apply to the base experience.

Returns

bool

True if experience was collected, false otherwise.

TryCollectExp(EntityCore, EntityCore, DamageResolutionContext)

Attempts to collect experience from a victim entity and add it to the collector's level. This is the main public interface for experience collection. Handles validation, extraction of IExpSource, calculation, adding to EntityLevel, and marking as harvested.

```
public virtual bool TryCollectExp(EntityCore collector, EntityCore victim,  
DamageResolutionContext dmgResolutionContext)
```

Parameters

collector EntityCore

The entity attempting to collect experience.

victim EntityCore

The entity that died and may provide experience.

dmgResolutionContext [DamageResolutionContext](#)

The damage resolution that caused the death.

Returns

bool

True if experience was successfully collected, false otherwise.

TryGetExpSource(EntityCore, out IExpSource)

Attempts to extract an IExpSource from the victim's EntityCore. Override this method to customize how the IExpSource is retrieved.

```
protected virtual bool TryGetExpSource(EntityCore victim, out IExpSource expSource)
```

Parameters

victim EntityCore

The entity core to extract the IExpSource from.

expSource IExpSource

The extracted IExpSource, or null if not found.

Returns

bool

True if an IExpSource was found, false otherwise.

Validate(EntityCore, EntityCore, DamageResolutionContext, out float)

Override this method to implement the specific validation logic for the strategy.

```
protected abstract bool Validate(EntityCore collector, EntityCore victim,  
DamageResolutionContext dmgResolutionContext, out float multiplier)
```

Parameters

collector EntityCore

The entity attempting to collect experience.

victim EntityCore

The entity that died and may provide experience.

dmgResolutionContext [DamageResolutionContext](#)

The damage resolution that caused the death.

multiplier float

Output multiplier to apply to the base experience (1.0 = full exp).

Returns

bool

True if the strategy's conditions are met, false otherwise.

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. Uses a configurable [ExpCollectionStrategySO](#) to determine when and how experience is collected. If no custom strategy is assigned, falls back to the default strategy from [IAstraRpgHealthConfig](#).

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

Inheritance

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

Properties

ActiveStrategy

Gets the active experience collection strategy. Returns the custom strategy if assigned, otherwise the default from configuration.

```
public ExpCollectionStrategySO ActiveStrategy { get; }
```

Property Value

[ExpCollectionStrategySO](#)

Methods

CheckCollectKillExp(EntityDiedContext)

Attempts to collect experience from an entity that died. Delegates all validation, extraction, calculation, and collection logic to the active strategy.

```
public void CheckCollectKillExp(EntityDiedContext context)
```

Parameters

context [EntityDiedContext](#)

The context of the entity death event.

Namespace ElectricDrill.AstraRpgHealth. Experience.Strategies

Classes

[DamageSourceKillExpStrategySO](#)

Experience collection strategy that grants experience when the damage source matches a configured source. Useful for environmental damage, traps, or any other specific damage source scenarios. For example: create an "Environmental" DamageSource and assign it to grant XP when enemies die from fall damage.

[DirectKillExpStrategySO](#)

Experience collection strategy that grants experience when the collector is the dealer of the killing blow. This is the default behavior - only the entity that dealt the final damage receives experience.

[FirstMatchExpStrategySO](#)

Composite experience collection strategy that evaluates a list of strategies in order. Delegates to the first strategy that successfully collects experience. Useful for implementing fallback behavior (e.g., try direct kill first, then environmental).

Class DamageSourceKillExpStrategySO

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

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

Experience collection strategy that grants experience when the damage source matches a configured source. Useful for environmental damage, traps, or any other specific damage source scenarios. For example: create an "Environmental" DamageSource and assign it to grant XP when enemies die from fall damage.

```
[CreateAssetMenu(fileName = "Damage Source Kill Exp Strategy", menuName = "Astra RPG
Health/Exp Collection Strategies/Damage Source Kill")]
public class DamageSourceKillExpStrategySO : ExpCollectionStrategySO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [ExpCollectionStrategySO](#) ← DamageSourceKillExpStrategySO

Inherited Members

[ExpCollectionStrategySO.TryCollectExp\(EntityCore, EntityCore, DamageResolutionContext\)](#) ,
[ExpCollectionStrategySO.CollectExp\(EntityCore, IExpSource, float\)](#) ,
[ExpCollectionStrategySO.CalculateExpAmount\(IExpSource, float\)](#) ,
[ExpCollectionStrategySO.TryGetExpSource\(EntityCore, out IExpSource\)](#).

Methods

Validate(EntityCore, EntityCore, DamageResolutionContext, out float)

Override this method to implement the specific validation logic for the strategy.

```
protected override bool Validate(EntityCore collector, EntityCore victim,
DamageResolutionContext dmgResolutionContext, out float multiplier)
```

Parameters

collector EntityCore

The entity attempting to collect experience.

victim EntityCore

The entity that died and may provide experience.

dmgResolutionContext [DamageResolutionContext](#)

The damage resolution that caused the death.

multiplier float

Output multiplier to apply to the base experience (1.0 = full exp).

Returns

bool

True if the strategy's conditions are met, false otherwise.

Class DirectKillExpStrategySO

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

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

Experience collection strategy that grants experience when the collector is the dealer of the killing blow. This is the default behavior - only the entity that dealt the final damage receives experience.

```
[CreateAssetMenu(fileName = "Direct Kill Exp Strategy", menuName = "Astra RPG Health/Exp Collection Strategies/Direct Kill")]
public class DirectKillExpStrategySO : ExpCollectionStrategySO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [ExpCollectionStrategySO](#) ← DirectKillExpStrategySO

Inherited Members

[ExpCollectionStrategySO.TryCollectExp\(EntityCore, EntityCore, DamageResolutionContext\)](#) ,
[ExpCollectionStrategySO.CollectExp\(EntityCore, IExpSource, float\)](#) ,
[ExpCollectionStrategySO.CalculateExpAmount\(IExpSource, float\)](#) ,
[ExpCollectionStrategySO.TryGetExpSource\(EntityCore, out IExpSource\)](#).

Methods

Validate(EntityCore, EntityCore, DamageResolutionContext, out float)

Override this method to implement the specific validation logic for the strategy.

```
protected override bool Validate(EntityCore collector, EntityCore victim,
DamageResolutionContext dmgResolutionContext, out float multiplier)
```

Parameters

collector EntityCore

The entity attempting to collect experience.

victim EntityCore

The entity that died and may provide experience.

dmgResolutionContext [DamageResolutionContext](#)

The damage resolution that caused the death.

multiplier float

Output multiplier to apply to the base experience (1.0 = full exp).

Returns

bool

True if the strategy's conditions are met, false otherwise.

Class FirstMatchExpStrategySO

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

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

Composite experience collection strategy that evaluates a list of strategies in order. Delegates to the first strategy that successfully collects experience. Useful for implementing fallback behavior (e.g., try direct kill first, then environmental).

```
[CreateAssetMenu(fileName = "First Match Exp Strategy", menuName = "Astra RPG Health/Exp Collection Strategies/First Match")]
public class FirstMatchExpStrategySO : ExpCollectionStrategySO
```

Inheritance

object ← [Object](#) ← [ScriptableObject](#) ← [ExpCollectionStrategySO](#) ← FirstMatchExpStrategySO

Inherited Members

[ExpCollectionStrategySO.CollectExp\(EntityCore, IExpSource, float\)](#) ,
[ExpCollectionStrategySO.CalculateExpAmount\(IExpSource, float\)](#) ,
[ExpCollectionStrategySO.TryGetExpSource\(EntityCore, out IExpSource\)](#).

Methods

TryCollectExp(EntityCore, EntityCore, DamageResolutionContext)

Override TryCollectExp to delegate to the first matching sub-strategy. The winning strategy handles the complete collection workflow.

```
public override bool TryCollectExp(EntityCore collector, EntityCore victim,
DamageResolutionContext dmgResolutionContext)
```

Parameters

collector EntityCore

victim EntityCore

dmgResolutionContext [DamageResolutionContext](#)

Returns

bool

Validate(EntityCore, EntityCore, DamageResolutionContext, out float)

Not used in FirstMatchExpStrategy since TryCollectExp is overridden.

```
protected override bool Validate(EntityCore collector, EntityCore victim,  
DamageResolutionContext dmgResolutionContext, out float multiplier)
```

Parameters

collector EntityCore

victim EntityCore

dmgResolutionContext [DamageResolutionContext](#)

multiplier float

Returns

bool

Namespace ElectricDrill.AstraRpgHealth.GameActions.Actions

Classes

[ResurrectGameActionSO<TContext>](#)

Base generic game action for resurrecting entities with EntityHealth components. Can be configured to resurrect with either a percentage of max HP or a flat HP amount.

Class ResurrectGameActionSO<TContext>

Namespace: [ElectricDrill.AstraRpgHealth.GameActions.Actions](#)

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

Base generic game action for resurrecting entities with EntityHealth components. Can be configured to resurrect with either a percentage of max HP or a flat HP amount.

```
public abstract class ResurrectGameActionSO<TContext> : GameAction<TContext>,  
IExecutable<TContext> where TContext : Component
```

Type Parameters

TContext

The context type that must be a Component with EntityHealth.

Inheritance

```
object ← Object ← ScriptableObject ← GameAction<TContext> ←  
ResurrectGameActionSO<TContext>
```

Implements

IExecutable<TContext>

Derived

[ResurrectComponentGameActionSO](#)

Inherited Members

```
GameAction<TContext>.DisplayName , GameAction<TContext>.ExecuteAsyncForUnityEvent(TContext) ,  
GameAction<TContext>.OnOperationCanceled() ,  
GameAction<TContext>.RunFireAndForget\(TContext, MonoBehaviour\)
```

Fields

_flatHpToRecover

```
[SerializeField]  
[Tooltip("Flat amount of HP to restore on resurrection (used when 'Use Percentage HP'
```

```
is false)")]  
protected long _flatHpToRecover
```

Field Value

long

_healSource

```
[SerializeField]  
[Tooltip("The heal source to attribute the resurrection to")]  
protected HealSourceSO _healSource
```

Field Value

[HealSourceSO](#)

_percentageHpToRecover

```
[SerializeField]  
[Tooltip("The percentage of max HP to restore (0-100)")]  
[Range(0, 100)]  
protected int _percentageHpToRecover
```

Field Value

int

_usePercentageHp

```
[SerializeField]  
[Tooltip("If true, resurrect with a percentage of max HP. If false, use flat HP amount.")]  
protected bool _usePercentageHp
```

Field Value

bool

Methods

ExecuteAsync(TContext, CancellationToken)

Executes the resurrection logic asynchronously. Attempts to get EntityHealth from the context and resurrects with configured HP settings.

```
public override Awaitable ExecuteAsync(TContext context, CancellationToken cancellationToken  
= default)
```

Parameters

context TContext

The context component, expected to have or be EntityHealth.

cancellationToken CancellationToken

Token to cancel the operation.

Returns

[Awaitable](#)

An Awaitable that completes when resurrection finishes.

Namespace ElectricDrill.AstraRpgHealth.GameActions.Actions.Component

Classes

[ResurrectComponentGameActionSO](#)

A game action used to resurrect an entity. Can be configured to resurrect with either a percentage of max HP or a flat HP amount. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Resurrect.

Class ResurrectComponentGameActionSO

Namespace: [ElectricDrill.AstraRpgHealth.GameActions.Actions.Component](#)

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

A game action used to resurrect an entity. Can be configured to resurrect with either a percentage of max HP or a flat HP amount. Provided as a ScriptableObject for easy assignment in the editor. Create instances via Assets -> Create -> Astra RPG Health / Death strategies / Resurrect.

```
[CreateAssetMenu(fileName = "Resurrect Game Action", menuName = "Astra RPG Framework/Game Actions/Context: Component/Resurrect")]
public class ResurrectComponentGameActionSO : ResurrectGameActionSO<Component>,  
IExecutable<Component>
```

Inheritance

```
object ← Object ← ScriptableObject ← GameAction<Component> ←  
ResurrectGameActionSO<Component> ← ResurrectComponentGameActionSO
```

Implements

```
IExecutable<Component>
```

Inherited Members

```
ResurrectGameActionSO<Component>.usePercentageHp,  
ResurrectGameActionSO<Component>.percentageHpToRecover,  
ResurrectGameActionSO<Component>.flatHpToRecover,  
ResurrectGameActionSO<Component>.healSource,  
ResurrectGameActionSO<Component>.ExecuteAsync(Component, CancellationToken),  
GameAction<Component>.DisplayName,  
GameAction<Component>.ExecuteAsyncForUnityEvent(Component),  
GameAction<Component>.OnOperationCanceled(),  
GameAction<Component>.RunFireAndForget(Component, MonoBehaviour)
```

Namespace ElectricDrill.AstraRpgHealth.Heal

Classes

[HealAmountContext](#)

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.

[HealSourceSO](#)

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.

[LifestealConfigSO](#)

ScriptableObject for configuring lifesteal mappings for different damage types.

[LifestealStatConfig](#)

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

[PreHealContext](#)

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

[PreHealContext.PreHealInfoStepBuilder](#)

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

[ReceivedHealContext](#)

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.

[ReceivedHealContext.ReceivedHealInfoStepBuilder](#)

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

Interfaces

[IHealable](#)

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

[PreHealContext.HealInfoAmount](#)

Builder step interface for specifying the heal amount.

[PreHealContext.HealInfoHealer](#)

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

[PreHealContext.HealInfoSource](#)

Builder step interface for specifying the heal source.

[PreHealContext.HealInfoTarget](#)

Builder step interface for specifying the target entity.

[ReceivedHealContext.HealInfoAmount](#)

Builder step: provide the heal amount.

[ReceivedHealContext.HealInfoHealer](#)

Builder step: provide the healer entity.

[ReceivedHealContext.HealInfoSource](#)

Builder step: provide the heal source.

[ReceivedHealContext.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 HealAmountContext

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 HealAmountContext
```

Inheritance

object ← HealAmountContext

Constructors

HealAmountContext(long, long, long)

Create a new instance describing the provided heal values.

```
public HealAmountContext(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 PreHealContext.

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

Property Value

long

Class HealSourceSO

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 HealSourceSO : ScriptableObject
```

Inheritance

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

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 [PreHealContext](#).

```
public interface IHealable
```

Methods

Heal(PreHealContext)

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

```
ReceivedHealContext Heal(PreHealContext context)
```

Parameters

`context` [PreHealContext](#)

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

Returns

[ReceivedHealContext](#)

A [ReceivedHealContext](#) 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(DamageAmountContext)

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

```
public long Evaluate(DamageAmountContext amounts)
```

Parameters

amounts [DamageAmountContext](#)

DamageAmountContext 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 LifestealConfigSO

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 LifestealConfigSO : ScriptableObject
```

Inheritance

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

Properties

LifestealMappings

Read-only mapping from DamageType to LifestealStatConfig.

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

Property Value

[IReadOnlyDictionary](#)<[DamageTypeSO](#), [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 HealSourceSO LifestealSource { get; }
```

Property Value

[HealSourceSO](#)

LifestealStat

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

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

Property Value

Stat

Class PreHealContext

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 PreHealContext : IHasTarget, IHasSource
```

Inheritance

object ← PreHealContext

Implements

IHasTarget, IHasSource

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 [PreHealContext](#). Use the returned builder to set amount, source and healer.

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

Property Value

[PreHealContext.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 healing.

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

Property Value

[HealSourceSO](#)

IsCritical

Whether this heal is flagged as a critical heal.

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

Property Value

bool

Source

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

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

Property Value

EntityCore

Target

The target entity receiving the healing.

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

Property Value

EntityCore

Interface PreHealContext.HealInfoAmount

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

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

Builder step interface for specifying the heal amount.

```
public interface PreHealContext.HealInfoAmount
```

Methods

WithAmount(long)

Specify the base heal amount.

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

Parameters

amount long

Returns

[PreHealContext.HealInfoSource](#)

Interface PreHealContext.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 PreHealContext.HealInfoHealer
```

Methods

WithHealer(EntityCore)

Specify the entity that performed the heal.

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

Parameters

healer EntityCore

Returns

[PreHealContext.HealInfoTarget](#)

Interface PreHealContext.HealInfoSource

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

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

Builder step interface for specifying the heal source.

```
public interface PreHealContext.HealInfoSource
```

Methods

WithSource(HealSourceSO)

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

```
PreHealContext.HealInfoHealer WithSource(HealSourceSO healSource)
```

Parameters

healSource [HealSourceSO](#)

Returns

[PreHealContext.HealInfoHealer](#)

Interface PreHealContext.HealInfoTarget

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

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

Builder step interface for specifying the target entity.

```
public interface PreHealContext.HealInfoTarget
```

Methods

WithTarget(EntityCore)

Specify the target entity.

```
PreHealContext.PreHealInfoStepBuilder WithTarget(EntityCore target)
```

Parameters

target EntityCore

Returns

[PreHealContext.PreHealInfoStepBuilder](#)

Class PreHealContext.PreHealInfoStepBuilder

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

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

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

```
public sealed class PreHealContext.PreHealInfoStepBuilder : PreHealContext.HealInfoAmount,  
    PreHealContext.HealInfoSource, PreHealContext.HealInfoHealer, PreHealContext.HealInfoTarget
```

Inheritance

object ← PreHealContext.PreHealInfoStepBuilder

Implements

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

Methods

Build()

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

```
public PreHealContext Build()
```

Returns

[PreHealContext](#)

WithAmount(long)

Set the base heal amount.

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

Parameters

`amount` long

Returns

[PreHealContext.HealInfoSource](#)

WithCriticalMultiplier(double)

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

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

Parameters

`multiplier` double

Returns

[PreHealContext.PreHealInfoStepBuilder](#)

WithHealer(EntityCore)

Set the entity that performed the heal.

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

Parameters

`healer` EntityCore

Returns

[PreHealContext.HealInfoTarget](#)

WithIsCritical(bool)

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

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

Parameters

isCritical bool

Returns

[PreHealContext.PreHealInfoStepBuilder](#)

WithSource(HealSourceSO)

Set the heal source.

```
public PreHealContext.HealInfoHealer WithSource(HealSourceSO healSource)
```

Parameters

healSource [HealSourceSO](#)

Returns

[PreHealContext.HealInfoHealer](#)

WithTarget(EntityCore)

Set the target entity.

```
public PreHealContext.PreHealInfoStepBuilder WithTarget(EntityCore target)
```

Parameters

target EntityCore

Returns

[PreHealContext.PreHealInfoStepBuilder](#)

Class ReceivedHealContext

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 ReceivedHealContext : IHasTarget, IHasSource
```

Inheritance

object ← ReceivedHealContext

Implements

IHasTarget, IHasSource

Constructors

ReceivedHealContext(HealAmountContext, PreHealContext, EntityCore)

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

```
public ReceivedHealContext(HealAmountContext healAmount, PreHealContext preHealContext,  
EntityCore target)
```

Parameters

healAmount [HealAmountContext](#)

Resolved heal amount information.

preHealContext [PreHealContext](#)

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 [ReceivedHealContext](#) using a step builder.

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

Property Value

[ReceivedHealContext.HealInfoAmount](#)

HealAmount

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

```
public HealAmountContext HealAmount { get; }
```

Property Value

[HealAmountContext](#)

HealSource

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

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

Property Value

[HealSourceSO](#)

IsCritical

True if the heal was a critical heal.

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

Property Value

bool

Source

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

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

Property Value

EntityCore

Target

The entity that receives the heal.

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

Property Value

EntityCore

Interface ReceivedHealContext.HealInfoAmount

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

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

Builder step: provide the heal amount.

```
public interface ReceivedHealContext.HealInfoAmount
```

Methods

WithAmount(HealAmountContext)

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

```
ReceivedHealContext.HealInfoSource WithAmount(HealAmountContext healAmount)
```

Parameters

healAmount [HealAmountContext](#)

Returns

[ReceivedHealContext.HealInfoSource](#)

Interface ReceivedHealContext.HealInfoHealer

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

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

Builder step: provide the healer entity.

```
public interface ReceivedHealContext.HealInfoHealer
```

Methods

WithHealer(EntityCore)

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

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

Parameters

`healer EntityCore`

Returns

[ReceivedHealContext.HealInfoTarget](#)

Interface ReceivedHealContext.HealInfoSource

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

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

Builder step: provide the heal source.

```
public interface ReceivedHealContext.HealInfoSource
```

Methods

WithSource(HealSourceSO)

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

```
ReceivedHealContext.HealInfoHealer WithSource(HealSourceSO healSource)
```

Parameters

healSource [HealSourceSO](#)

Returns

[ReceivedHealContext.HealInfoHealer](#)

Interface ReceivedHealContext.HealInfoTarget

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

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

Builder step: provide the target entity and finalize options.

```
public interface ReceivedHealContext.HealInfoTarget
```

Methods

WithTarget(EntityCore)

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

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

Parameters

target EntityCore

Returns

[ReceivedHealContext.ReceivedHealInfoStepBuilder](#)

Class

ReceivedHealContext.ReceivedHealInfoStepBuilder

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

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

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

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

Inheritance

object ← ReceivedHealContext.ReceivedHealInfoStepBuilder

Implements

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

Properties

Builder

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

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

Property Value

[ReceivedHealContext.HealInfoAmount](#)

Methods

Build()

Build the final [ReceivedHealContext](#) instance.

```
public ReceivedHealContext Build()
```

Returns

[ReceivedHealContext](#)

WithAmount(HealAmountContext)

Set the resolved [HealAmountContext](#).

```
public ReceivedHealContext.HealInfoSource WithAmount(HealAmountContext healAmount)
```

Parameters

healAmount [HealAmountContext](#)

Returns

[ReceivedHealContext.HealInfoSource](#)

WithHealer(EntityCore)

Set the entity that performed the heal.

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

Parameters

healer EntityCore

Returns

[ReceivedHealContext.HealInfoTarget](#)

WithIsCritical(bool)

Optionally mark the heal as critical.

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

Parameters

`isCritical` bool

Returns

[ReceivedHealContext.ReceivedHealInfoStepBuilder](#)

WithSource(HealSourceSO)

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

```
public ReceivedHealContext.HealInfoHealer WithSource(HealSourceSO healSource)
```

Parameters

`healSource` [HealSourceSO](#)

Returns

[ReceivedHealContext.HealInfoHealer](#)

WithTarget(EntityCore)

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

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

Parameters

`target` EntityCore

Returns

[ReceivedHealContext.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

[ResurrectionContext](#)

Contains information about an entity resurrection event.

Interfaces

[IResurrectable](#)

Interface IResurrectable

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

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

```
public interface IResurrectable
```

Methods

Resurrect(Percentage, HealSourceSO)

Resurrects the entity with a percentage of its maximum HP.

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

Parameters

withHpPercent Percentage

The percentage of maximum HP to resurrect with.

healSource [HealSourceSO](#)

The heal source associated with the resurrection.

Resurrect(long, HealSourceSO)

Resurrects the entity with a fixed amount of HP.

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

Parameters

withHp long

The fixed amount of HP to resurrect with.

healSource [HealSourceSO](#)

The heal source associated with the resurrection.

Class ResurrectionContext

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

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

Contains information about an entity resurrection event.

```
public class ResurrectionContext : IHasTarget, IHasValueChange<long>
```

Inheritance

object ← ResurrectionContext

Implements

IHasTarget, IHasValueChange<long>

Constructors

ResurrectionContext(EntityCore, long, long, HealSourceSO)

Initializes a new instance of the ResurrectionContext class.

```
public ResurrectionContext(EntityCore target, long previousHp, long newHp,  
HealSourceSO healSource)
```

Parameters

target EntityCore

The entity being resurrected.

previousHp long

The HP before resurrection.

newHp long

The HP after resurrection.

healSource [HealSourceSO](#)

The heal source associated with the resurrection.

Properties

AbsAmount

The absolute amount of change between PreviousValue and NewValue.

```
public long AbsAmount { get; }
```

Property Value

long

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 HealSourceSO HealSource { get; }
```

Property Value

[HealSourceSO](#)

NewValue

Gets the health points the entity was resurrected with.

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

Property Value

long

PreviousValue

Gets the health points the entity had before resurrection (typically at or below death threshold).

```
public long PreviousValue { get; }
```

Property Value

long

Target

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

Property Value

EntityCore

Namespace ElectricDrill.AstraRpgHealth. Scaling.Components

Classes

[HealthScalingComponentSO](#)

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

Class HealthScalingComponentSO

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 HealthScalingComponentSO : ScalingComponent
```

Inheritance

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

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.

Namespace ElectricDrill.AstraRpgHealth.Utils

Classes

[EntityCoreHealthExtensions](#)

Class EntityCoreHealthExtensions

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

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

```
public static class EntityCoreHealthExtensions
```

Inheritance

object ← EntityCoreHealthExtensions

Methods

GetEntityHealth(EntityCore)

Gets the EntityHealth component associated with this EntityCore. Result is cached for performance similar to internal fields.

```
public static EntityHealth GetEntityHealth(this EntityCore entity)
```

Parameters

entity EntityCore

Returns

[EntityHealth](#)