**Ruleset Index**

Version 0.9 for Ruleset Rel: 1 Ver: 9

Contents

[**Variables and Types**: 3](#_Toc18596028)

[**CharacterAbilitiesEnum** 3](#_Toc18596029)

[**CharacterDefensesEnum** 3](#_Toc18596030)

[**CharacterAttacksEnum** 4](#_Toc18596031)

[**WeaponDamageTypes** 4](#_Toc18596032)

[**maxPartySize** 5](#_Toc18596033)

[**baseDeflectionScore** 5](#_Toc18596034)

[**DeflectionPerLvl** 5](#_Toc18596035)

[**baseReflexScore** 5](#_Toc18596036)

[**ReflexPerLvl** 5](#_Toc18596037)

[**baseFortitudeScore** 6](#_Toc18596038)

[**FortitudePerLvl** 6](#_Toc18596039)

[**baseWillScore** 6](#_Toc18596040)

[**WillPerLvl** 6](#_Toc18596041)

[**baseDeathScore** 7](#_Toc18596042)

[**DeathPerLvl** 7](#_Toc18596043)

[**baseAwareScore** 7](#_Toc18596044)

[**AwarenesPerLvl** 7](#_Toc18596045)

[**abilityThreshold** 8](#_Toc18596046)

[**abilityPerPointPlus** 8](#_Toc18596047)

[**abilityPerPointMinus** 8](#_Toc18596048)

[**baseAttackScore** 8](#_Toc18596049)

[**attackGainPerLevel** 9](#_Toc18596050)

[**baseAdvantageScore** 9](#_Toc18596051)

[**advantageGainPerLevel** 9](#_Toc18596052)

[**Methods** 10](#_Toc18596053)

[**GetBaseDeflection(int agilityScore, int characterLevel)** 10](#_Toc18596054)

[**GetBaseWill(int intellectScore, int presenceScore, int characterLevel)** 10](#_Toc18596055)

[**GetBaseFort(int mightScore, int enduranceScore, int characterLevel)** 11](#_Toc18596056)

[**GetBaseRefex(int agilityScore, int perceptionScore, int characterLevel)** 12](#_Toc18596057)

[**GetBaseAttackScore(int perceptionScore, int characterLevel)** 12](#_Toc18596058)

[**GetBaseAdvantageScore(int agilityScore, int perceptionScore, int characterLevel)** 13](#_Toc18596059)

[**ReturnAbilityModifier(int abilityScore)** 13](#_Toc18596060)

[**GetCurrentLevel(int experiencePointsValue)** 14](#_Toc18596061)

[**ResolveHit (int attScore, int enemyDef, int enemyDR, float wpnMinDmg, float wpnMaxDmg)** 15](#_Toc18596062)

[**ResolveHit (int attScore, int enemyDef, int enemyDR, float wpnMinDmg, float wpnMaxDmg)** 15](#_Toc18596063)

[**Helper Classes**: 16](#_Toc18596064)

[**Hit** 16](#_Toc18596065)

[**WeaponUsed** 16](#_Toc18596066)

# **Variables and Types**:

**CharacterAbilitiesEnum**:

**Namespace:** Ruleset.

**Variable Type**: Public ENUM

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetAbilitiesEnum.cs

**Description**: Creates an enum variable that has all Character’s Main Abilities as possible choices.

Example:

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **using Ruleset;**
5. **public** **class** TestAbilities : MonoBehaviour
6. {
7. **public** **CharacterAbilitiesEnum** characterAbility;
9. // Start is called before the first frame update
10. **void** Start()
11. {
13. }
15. }

**CharacterDefensesEnum**:

**Namespace:** Ruleset.

**Variable Type**: Public ENUM

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetDefensesEnum.cs

**Description**: Creates an enum variable that has all Defenses as possible choices.

Example:

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **using** **Ruleset;**
5. **public** **class** TestDefenses : MonoBehaviour
6. {
7. **public** CharacterDefensesEnum characterDefense;
8. // Start is called before the first frame update
9. **void** Start()
10. {
12. }
14. }

**CharacterAttacksEnum**:

**Namespace:** Ruleset.

**Variable Type**: Public ENUM

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetAttacksEnum.cs

**Description**: Creates an enum variable that has all Attack Types as possible choices.

Example:

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **using** **Ruleset;**
5. **public** **class** TestAttacks : MonoBehaviour
6. {
7. **public** CharacterAttacksEnum characterAttack;
8. // Start is called before the first frame update
9. **void** Start()
10. {
12. }
14. }

**WeaponDamageTypes**:

**Namespace:** Ruleset.

**Variable Type**: Public ENUM

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetDamageTypesEnum.cs

**Description**: Creates an enum variable that has all **Damage Types** as possible choices.

Example:

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **using** **Ruleset;**
5. **public** **class** TestDamageTypes : MonoBehaviour
6. {
7. **public** WeaponDamageTypes damageType;
8. // Start is called before the first frame update
9. **void** Start()
10. {
12. }
13. }

**maxPartySize**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the Configured Maximum Party Size.

**baseDeflectionScore**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the Configured Starting Deflection Score.

**DeflectionPerLvl**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points per level gain the player get to his DEF score.

**baseReflexScore**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the Configured Starting Deflection Score.

**ReflexPerLvl**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points per level gain the player get to his REF score.

**baseFortitudeScore**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the Configured Starting FORT Score.

**FortitudePerLvl**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points per level gain the player get to his FORT score.

**baseWillScore**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the Configured Starting WILL Score.

**WillPerLvl**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points per level gain the player get to his WILL score.

**baseDeathScore**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the Configured Starting DTH Score.

**DeathPerLvl**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points per level gain the player get to his DTH score.

**baseAwareScore**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the Configured Starting AWARE Score.

**AwarenesPerLvl**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points per level gain the player get to his AWARE score.

**abilityThreshold**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the configured Plus/Minus threshold for Abilities.

**abilityPerPointPlus**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points the player gains for every point above the abilityThreshold.

**abilityPerPointMinus**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points the player looses for every point above the abilityThreshold.

**baseAttackScore**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the configured player’s starting Attack Score (toHit).

**attackGainPerLevel**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points per level gain the player get to his Attack score.

**baseAdvantageScore**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: Holds the configured player’s starting Advantage Score.

**advantageGainPerLevel**

**Namespace:** Ruleset.

**Type**: Public

**Return Type**:

**Parent Class**: RulesetGlossary

**Location**: Assets/Ruleset/Scripts/Libraries/RulesetGlossary.cs

**Description**: How many points per level gain the player get to his Advantage score.

**Methods**:

## **GetBaseDeflection(int agilityScore, int characterLevel)**

**Namespace:** None.

**Type**: Static

**Return Type**: INT

**Parent Class**: ClassMechanics

**Location**: Assets/Ruleset/Scripts/Libraries/ClassMechanics.cs

**Description**: Returns an Integer with the Character’s base Deflection score. No buffs, items etc are added.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **public** **class** TestBaseDeflectionScore : MonoBehaviour
5. {
6. // Start is called before the first frame update
7. **void** Start()
8. {
10. **int** baseDeflectionScore = **ClassMechanics**.**GetBaseDeflection**(12, 5);
11. }
13. // Update is called once per frame
14. **void** Update()
15. {
17. }
18. }

## **GetBaseWill(int intellectScore, int presenceScore, int characterLevel)**

**Namespace:** None.

**Type**: Static

**Return Type**: INT

**Parent Class**: ClassMechanics

**Location**: Assets/Ruleset/Scripts/Libraries/ClassMechanics.cs

**Description**: Returns an Integer with the Character’s base Will score. No buffs, items etc are added.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **public** **class** TestBaseWillScore : MonoBehaviour
5. {
6. // Start is called before the first frame update
7. **void** Start()
8. {
10. **int** baseWillScore = **ClassMechanics**.**GetBaseWill**(12, 15, 5);
11. }
13. // Update is called once per frame
14. **void** Update()
15. {
17. }
18. }

## **GetBaseFort(int mightScore, int enduranceScore, int characterLevel)**

**Namespace:** None.

**Type**: Static

**Return Type**: INT

**Parent Class**: ClassMechanics

**Location**: Assets/Ruleset/Scripts/Libraries/ClassMechanics.cs

**Description**: Returns an Integer with the Character’s base Fortitute score. No buffs, items etc are added.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **public** **class** TestBaseFortScore : MonoBehaviour
5. {
6. // Start is called before the first frame update
7. **void** Start()
8. {
10. **int** baseFortScore = **ClassMechanics**.**GetBaseFort**(12, 15, 5);
11. }
13. // Update is called once per frame
14. **void** Update()
15. {
17. }
18. }

## **GetBaseRefex(int agilityScore, int perceptionScore, int characterLevel)**

**Namespace:** None.

**Type**: Static

**Return Type**: INT

**Parent Class**: ClassMechanics

**Location**: Assets/Ruleset/Scripts/Libraries/ClassMechanics.cs

**Description**: Returns an Integer with the Character’s base Reflex score. No buffs, items etc are added.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **public** **class** TestBaseReflexScore : MonoBehaviour
5. {
6. // Start is called before the first frame update
7. **void** Start()
8. {
10. **int** baseReflexScore = **ClassMechanics**.**GetBaseReflex**(12, 15, 5);
11. }
13. // Update is called once per frame
14. **void** Update()
15. {
17. }
18. }

## **GetBaseAttackScore(int perceptionScore, int characterLevel)**

**Namespace:** None.

**Type**: Static

**Return Type**: INT

**Parent Class**: ClassMechanics

**Location**: Assets/Ruleset/Scripts/Libraries/ClassMechanics.cs

**Description**: Returns an Integer with the Character’s base Attack (tohit) score. No buffs, items etc are added.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **public** **class** TestBaseReflexScore : MonoBehaviour
5. {
6. // Start is called before the first frame update
7. **void** Start()
8. {
10. **int** baseAttackScore = **ClassMechanics**.**GetBaseAttackScore**(15, 5);
11. }
13. // Update is called once per frame
14. **void** Update()
15. {
17. }
18. }

## **GetBaseAdvantageScore(int agilityScore, int perceptionScore, int characterLevel)**

**Namespace:** None.

**Type**: Static

**Return Type**: INT

**Parent Class**: ClassMechanics

**Location**: Assets/Ruleset/Scripts/Libraries/ClassMechanics.cs

**Description**: Returns an Integer with the Character’s base Advantage score. No buffs, items etc are added.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **public** **class** TestBaseReflexScore : MonoBehaviour
5. {
6. // Start is called before the first frame update
7. **void** Start()
8. {
10. **int** baseAdvantageScore = **ClassMechanics**.**GetBaseAdvantageScore**(12, 15, 5);
11. }
13. // Update is called once per frame
14. **void** Update()
15. {
17. }
18. }

## **ReturnAbilityModifier(int abilityScore)**

**Namespace:** None.

**Type**: Static

**Return Type**: INT

**Parent Class**: ClassMechanics

**Location**: Assets/Ruleset/Scripts/Libraries/ClassMechanics.cs

**Description**: Returns an Integer with the Character’s ability Modifier based on the Ability Score you provide.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **using** Ruleset;
6. **public** **class** Test : MonoBehaviour
7. {
8. // Start is called before the first frame update
9. **void** Start()
10. {
11. //returns the total base modifier for a Might score of 15
12. **int** mightAbilityMod = **ClassMechanics**.**ReturnAbilityModifier**(15);
13. }
15. // Update is called once per frame
16. **void** Update()
17. {
19. }
20. }

## **GetCurrentLevel(int experiencePointsValue)**

**Namespace:** None.

**Type**: Static

**Return Type**: INT

**Parent Class**: ClassMechanics

**Requirements:** GameMaster Singleton Prefab.

**Location**: Assets/Ruleset/Scripts/Libraries/ClassMechanics.cs

**Description**: Returns an Integer with the Character’s level according the Experience points you’ve provided. It doesn’t handle level up function nor if the Player has advanced a level.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **using** Ruleset;
6. **public** **class** Test : MonoBehaviour
7. {
8. // Start is called before the first frame update
9. **void** Start()
10. {
11. //returns the player's level if he had 600 exp.
12. **int** playerCurrentLevel = GetCurrentLevel(600);
13. }
14. }

## **ResolveHit (int attScore, int enemyDef, int enemyDR, float wpnMinDmg, float wpnMaxDmg)**

**Namespace:** Ruleset.

**Type**: Static

**Return Type**: HIT

**Parent Class**: CombatMechanics

**Requirements:** Has 3 Overloads

**Location**: Assets/Ruleset/Scripts/Libraries/CombatMechanics.cs

**Description**: Returns a class of type HIT.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **using** Ruleset;
6. **public** **class** Test : MonoBehaviour
7. {
8. **public** **CombatMechanics.Hit** hit;
10. **void** Start()
11. {
12. hit = **CombatMechanics**.**ResolveHit**(attScore, defScore, defDR, wpnMin, wpnMax);   }
13. }

## **ResolveHit (int attScore, int enemyDef, int enemyDR, float wpnMinDmg, float wpnMaxDmg)**

**Namespace:** Ruleset.

**Type**: Static

**Return Type**: HIT

**Parent Class**: CombatMechanics

**Requirements:** Has 3 Overloads

**Location**: Assets/Ruleset/Scripts/Libraries/CombatMechanics.cs

**Description**: Returns a class of type HIT.

**Example:**

1. **using** System.Collections;
2. **using** System.Collections.Generic;
3. **using** UnityEngine;
4. **using** Ruleset;
6. **public** **class** Test : MonoBehaviour
7. {
8. **public** **CombatMechanics.Hit** hit;
10. **void** Start()
11. {
12. **WeaponUsed** wpnUsed = new WeaponUsed("Blade", WeaponDamageTypes.SLASH, 1.0f, 11.0f);
13. hit = **CombatMechanics**.**ResolveHit**(attScore, defScore, defDR, wpnUsed);
14. }
15. }

# **Helper Classes**:

**Hit**:

**Namespace:** Ruleset.

**Requirements**: CombatMechanics

**Returns**: Hit.hitType, Hit.success, Hit.damageDone.

**WeaponUsed**:

**Namespace:** Ruleset.

**Requirements**: string weaponName, WeaponDamageType dmdType, float minDmg, float maxDmg.

**Returns**: Null.