Player Stats Modules

API Reference Guide

Third-Person Finite State Machine

and

Movement

Documentation v1.0
Last Updated 07/11/2025

Table of Contents

Overview

The player stats system is composed of 4 independent, modular ScriptableObjects:

- **PlayerProgressionModule** - Experience tracking, leveling, stat scaling
- **PlayerHealthModule** - Health, damage, death, regeneration
- **PlayerStaminaModule** - Stamina usage and regeneration
- **PlayerManaModule** - Mana usage and regeneration

Each module is fully self-contained and can be used independently. They optionally integrate with each other for enhanced functionality (e.g., health/stamina/mana scale with player level from ProgressionModule if installed).

Quick Setup

Note: *If using the StatsVariant Player Prefab, all modules are pre-configured.*

One-Click Installation (Recommended)

This method is recommended for beginners and provides the fastest setup:

- Select your player GameObject in the Hierarchy
- Go to **MiST → Quick Setup → Add All Stats Modules**
- Select which modules you want (Health, Stamina, Mana, Progression)
- Click **Install Selected Modules**

This creates the module assets and adds them to your PlayerStateMachine automatically.

Manual Installation (Advanced)

For users who prefer manual control:

Step 1: Create Module Assets

- Right-click in Project → Create → MiST → Player Modules → Progression
- Right-click in Project → Create → MiST → Player Modules → Health
- Right-click in Project → Create → MiST → Player Modules → Stamina
- Right-click in Project → Create → MiST → Player Modules → Mana

Step 2: Add to Player

- Select your player GameObject
- Find PlayerStateMachine component
- Add the created modules to the "Modules" list

Step 3: Configure Values

Configure values in the Inspector of each ScriptableObject (see Configuration sections below).

1. PlayerProgressionModule

Provides experience tracking, leveling, and stat scaling multipliers for other modules.

Configuration

Field	Type	Default	Description
startingLevel	int	1	Starting player level
xpRequiredCurve	AnimationCurve	Linear	XP required per level (X=level, Y=XP)
statIncreasePerLevel	float	0.1	% increase to max stats per level (0.1 = 10%)
restoreResourcesOnLevelUp	bool	true	Fully restore HP/Stamina/Mana on level up

Getting the Module

```
PlayerProgressionModule progression =
playerStateMachine.GetModule<PlayerProgressionModule>();

if (progression == null)

{

  Debug.LogError("PlayerProgressionModule not found!");

  return;

}
```

Properties

Property	Description
int CurrentLevel { get; }	Current player level
float CurrentXP { get; }	Current XP amount
float ExperienceToNextLevel { get; }	XP required for next level

bool RestoreResourcesOnLevelUp { get; }	Config value (for other modules)

Methods

Method	Description
void GainExperience(float amount)	Award XP (auto-levels if threshold reached)
float GetStatMultiplier()	Returns stat multiplier (1.0 at level 1, increases with level)

Events

Event	Description
event Action OnLevelUp;	Fired when player levels up
event Action<float> OnExperienceGained;	Fired when XP awarded (passes amount)

Example Usage

```
// Award XP

progression.GainExperience(100f);


// Subscribe to level up

progression.OnLevelUp += () => {

  Debug.Log($"Level up! Now level {progression.CurrentLevel}");

  ShowLevelUpEffect();

  PlayLevelUpSound();

};


// Track XP progress

progression.OnExperienceGained += (amount) => {

  float progress = progression.CurrentXP / progression.ExperienceToNextLevel;

  UpdateXPBar(progress);

};
```

EventManager Integration

Event Name	Parameter	Description
"Player.ExperienceGained"	float	XP amount gained
"Player.LevelUp"	int	New level

2. PlayerHealthModule

Provides health system with damage, healing, death, and auto-regeneration.

Configuration

Field	Type	Default	Description
baseMaxHealth	float	100	Maximum health at level 1
healthRegenRate	float	5	HP regenerated per second
healthRegenDelay	float	5	Seconds after damage before regen starts

Note: MaxHealth automatically scales with player level if PlayerProgressionModule is present.

Getting the Module

```
PlayerHealthModule health =
playerStateMachine.GetModule<PlayerHealthModule>();

if (health == null)

{

   Debug.LogError("PlayerHealthModule not found!");

   return;

}
```

Properties

Property	Description
float CurrentHealth { get; }	Current HP
float MaxHealth { get; }	Max HP (scales with level if ProgressionModule present)
bool IsDead { get; }	Is player dead?

Methods

Method	Description
void TakeDamage(float damage)	Deal damage to player
void Heal(float amount)	Heal player by amount
void RestoreHealth()	Fully restore health to max
void Revive(float healthAmount)	Revive player (0 = full health)

Events

Event	Description
event Action<float> OnHealthChanged;	Fired when HP changes (passes current HP)
event Action OnDeath;	Fired when player dies

Example Usage

```
// Deal damage

health.TakeDamage(25f);



// Heal

health.Heal(50f);



// Subscribe to events

health.OnHealthChanged += (currentHP) => {

  Debug.Log($"Health: {currentHP}/{health.MaxHealth}");

  UpdateHealthBar(currentHP, health.MaxHealth);

};



health.OnDeath += () => {

  Debug.Log("Player died!");

  ShowDeathScreen();

};
```

```
// Check if dead
if (health.IsDead)
{
    health.Revive(100f);
}
```

EventManager Integration

Event Name	Parameter	Description
"Player.HealthChanged"	float	Current health
"Player.Death"	none	Player died

3. PlayerStaminaModule

Provides stamina system for actions like sprinting, dodging, and blocking.

Configuration

Field	Type	Default	Description
baseMaxStamina	float	100	Maximum stamina at level 1
staminaRegenRate	float	20	Stamina regenerated per second
staminaRegenDelay	float	2	Seconds after use before regen starts

Note: MaxStamina automatically scales with player level if PlayerProgressionModule is present.

Getting the Module

```
PlayerStaminaModule stamina =
playerStateMachine.GetModule<PlayerStaminaModule>();
if (stamina == null)
{
    Debug.LogError("PlayerStaminaModule not found!");
    return;
```

}

Properties

Property	Description
float CurrentStamina { get; }	Current stamina
float MaxStamina { get; }	Max stamina (scales with level if ProgressionModule present)

Methods

Method	Description
bool Use(float amount)	Returns false if insufficient
void Restore(float amount)	Restore stamina (0 = restore to max)

Events

Event	Description
event Action<float> OnStaminaChanged;	Fired when stamina changes
event Action OnStaminaDepleted;	Fired when stamina reaches 0

Example Usage

```
// Check and use stamina

if (stamina.Use(20f))

{

   Dodge();

}

else

{

   ShowLowStaminaWarning();

}



// Subscribe to depletion

stamina.OnStaminaDepleted += () => {

   StopSprinting();
```

```
};


// Restore stamina

stamina.Restore(50f);
```

EventManager Integration

Event Name	Parameter	Description
"Player.StaminaDepleted"	none	Stamina reached 0

4. PlayerManaModule

Provides mana system for spells and abilities.

Configuration

Field	Type	Default	Description
baseMaxMana	float	100	Maximum mana at level 1
manaRegenRate	float	10	Mana regenerated per second
manaRegenDelay	float	3	Seconds after use before regen starts

Note: MaxMana automatically scales with player level if PlayerProgressionModule is present.

Getting the Module

```
PlayerManaModule mana = playerStateMachine.GetModule<PlayerManaModule>();

if (mana == null)

{

   Debug.LogError("PlayerManaModule not found!");

   return;

}
```

Properties

Property	Description

float CurrentMana { get; }	Current mana
float MaxMana { get; }	Max mana (scales with level if ProgressionModule present)

Methods

Method	Description
bool Use(float amount)	Returns false if insufficient
void Restore(float amount)	Restore mana (0 = restore to max)

Events

Event	Description
event Action<float> OnManaChanged;	Fired when mana changes

Example Usage

```
// Cast spell

if (mana.Use(30f))

{

   CastFireball();

}



// Mana potion

mana.Restore(50f);



// Subscribe to changes

mana.OnManaChanged += (currentMana) => {

   UpdateManaBar(currentMana, mana.MaxMana);

};
```

EventManager Integration

Event Name	Parameter	Description
"Player.ManaChanged"	float	Current mana

Module Integration

How Modules Work Together

Modules can optionally integrate with each other without hard dependencies. For example:

```
// PlayerHealthModule checks for PlayerProgressionModule

public float MaxHealth

{

  get

  {

    var progression = cachedStateMachine?.GetModule<PlayerProgressionModule>();

    float multiplier = progression?.GetStatMultiplier() ?? 1f;

    return baseMaxHealth * multiplier;

  }

}
```

Benefits:

- Use modules independently (e.g., Health-only without Progression)
- MaxHealth/MaxStamina/MaxMana automatically scale with level if ProgressionModule present
- Resources can auto-restore on level-up if restoreResourcesOnLevelUp is enabled

Example: Level Scaling

```
// At Level 1:

// - baseMaxHealth = 100, multiplier = 1.0 → MaxHealth = 100

// - baseMaxStamina = 100, multiplier = 1.0 → MaxStamina = 100


// At Level 5 (with statIncreasePerLevel = 0.1):
```

```
// - multiplier = 1.0 + (5-1) * 0.1 = 1.4
// - MaxHealth = 100 * 1.4 = 140
// - MaxStamina = 100 * 1.4 = 140
```

Integration Examples

Example 1: Sprint with Stamina Drain

```
// In PlayerFreeMovementState
public override void Enter()
{
    var stamina = stateMachine.GetModule<PlayerStaminaModule>();
    if (stamina != null)
    {
        stateMachine.InputBridge.SprintStartEvent += () => StartSprint(stamina);
        stateMachine.InputBridge.SprintStopEvent += StopSprint;
        stamina.OnStaminaDepleted += StopSprint;
    }
}

private void StartSprint(PlayerStaminaModule stamina)
{
    if (isSprinting) return;
    isSprinting = true;
    stateMachine.StartCoroutine(DrainStaminaWhileSprinting(stamina));
}
```

```
private IEnumerator DrainStaminaWhileSprinting(PlayerStaminaModule stamina)
{
  while (isSprinting)
  {
    if (!stamina.Use(10f * Time.deltaTime))
    {
      StopSprint();
      yield break;
    }
    yield return null;
  }
}
```

Example 2: Health Pickup Item

```
public class HealthPotion : MonoBehaviour
{
  [SerializeField] private float healAmount = 50f;

  private void OnTriggerEnter(Collider other)
  {
    var sm = other.GetComponent<PlayerStateMachine>();
    if (sm != null)
    {
      var health = sm.GetModule<PlayerHealthModule>();
```

```
        if (health != null)
        {
            health.Heal(healAmount);
            Destroy(gameObject);
        }
    }
  }
}
```

Example 3: Enemy Dealing Damage

```
public class Enemy : MonoBehaviour
{
  [SerializeField] private float damageAmount = 25f;

  private void OnCollisionEnter(Collision collision)
  {
    var sm = collision.gameObject.GetComponent<PlayerStateMachine>();
    if (sm != null)
    {
      var health = sm.GetModule<PlayerHealthModule>();
      if (health != null)
      {
        health.TakeDamage(damageAmount);
      }
    }
```

```
    }
}
```

Example 4: Ability Costing Mana

```
public class FireballAbility
{
    private const float ManaCost = 30f;

    public bool TryCast(PlayerStateMachine sm)
    {
        var mana = sm.GetModule<PlayerManaModule>();
        if (mana == null) return false;

        if (mana.Use(ManaCost))
        {
            SpawnFireball();
            return true;
        }
        return false;
    }
}
```

Example 5: XP Reward on Kill

```
public class Enemy : MonoBehaviour
{
    [SerializeField] private float xpReward = 50f;
```

```
    public void Die(GameObject killer)
    {
        var sm = killer.GetComponent<PlayerStateMachine>();
        if (sm != null)
        {
            var progression = sm.GetModule<PlayerProgressionModule>();
            if (progression != null)
            {
                progression.GainExperience(xpReward);
            }
        }
        Destroy(gameObject);
    }
}
```

Example 6: Dodge Ability Costing Stamina

```
public class DodgeAbility
{
    private const float StaminaCost = 25f;

    public bool TryDodge(PlayerStateMachine sm)
    {
        var stamina = sm.GetModule<PlayerStaminaModule>();
        if (stamina == null) return false;
```

```
        if (stamina.Use(StaminaCost))
        {
            PerformDodgeRoll();
            return true;
        }
        return false;
    }
}
```

EventManager Integration (Global Events)

All modules broadcast events globally via EventManager for cross-system communication:

```
using MistInteractive.ThirdPerson;

// Listen to global events
EventManager.StartListening<float>("Player.HealthChanged", OnPlayerHealthChanged);
EventManager.StartListening("Player.Death", OnPlayerDeath);
EventManager.StartListening<int>("Player.LevelUp", OnPlayerLevelUp);

// Unsubscribe
EventManager.StopListening<float>("Player.HealthChanged", OnPlayerHealthChanged);
```

All Available Events

Event Name	Parameter	Source Module
"Player.HealthChanged"	float	PlayerHealthModule

"Player.Death"	none	PlayerHealthModule
"Player.StaminaDepleted"	none	PlayerStaminaModule
"Player.ManaChanged"	float	PlayerManaModule
"Player.ExperienceGained"	float	PlayerProgressionModule
"Player.LevelUp"	int	PlayerProgressionModule

Troubleshooting

Module not found

- Use MiST → Quick Setup → Add All Stats Modules for automatic setup
- If manual: Check that module assets are in the Modules list
- Verify the assets are not null in Inspector

Events not firing

- Make sure you subscribed correctly: health.OnHealthChanged += Method;
- Check that module was installed (happens in Start())
- Verify you didn't subscribe too early (before Start())

Stats not regenerating

- Check regen delays in module configuration
- Make sure coroutines are running (module needs PlayerStateMachine)
- Verify regeneration isn't interrupted (using resource resets delay)

Level not increasing

- Check xpRequiredCurve in Inspector
- Verify curve has values for your target levels
- Use Debug.Log to check CurrentXP vs ExperienceToNextLevel

Max stats not scaling with level

- Ensure PlayerProgressionModule is installed alongside Health/Stamina/Mana modules
- Check statIncreasePerLevel value in PlayerProgressionModule
- Verify the module is in the same PlayerStateMachine