🏕️ Survival Game Framework

Complete Data-Driven Survival System for Unreal Engine 5

Version 1.2.0 **UE 5.6 / 5.7** **Multiplayer Ready**

100% Blueprint Friendly

⚡ Quick Start Guide

Get your survival game running in under 5 minutes! This guide walks you through the essential setup.

1 Enable the Plugin

Open your project in Unreal Engine, then:

1. Go to **Edit** → **Plugins** from the top menu
2. Search for **"Survival Game Framework"**
3. Check the **Enabled** checkbox
4. Click **Restart Now** when prompted

2 Add Components to Your Character

Open your Character Blueprint and add these components from the Components panel:

- **Survival Inventory Component** — Inventory and hotbar management
- **Survival Needs Component** — Health, hunger, thirst, stamina
- **Survival Crafting Component** — Recipe crafting system
- **Survival Building Component** — Structure placement

> ℹ️ **How to Add Components**

> In your Blueprint, click **Add Component**, then search for "Survival". You'll see all available components listed.

3 Create Items DataTable

The plugin uses DataTables to define items. Here's how to create one:

1. Right-click in **Content Browser**
2. Select **Miscellaneous** → **DataTable**
3. In the popup, select **SurvivalItemData** as Row Structure
4. Name it **DT_Items** and save

4 Add Your First Items

Open your new DataTable and create some basic items:

1. Click **+ Add** to create a new row
2. Name the row **Wood** (this becomes the Item ID)
3. Set **Display Name** to "Wood"
4. Set **Category** to Resource
5. Set **Max Stack Size** to 50
6. Repeat for other items: Stone, Apple, Water, etc.
7. Save the DataTable

5 Connect the DataTable

Link your DataTable to the component:

1. Open your Character Blueprint
2. Select the **Survival Inventory Component**
3. In the Details panel, find **Items Data Table**
4. Set it to your **DT_Items** DataTable

6 Test Your Setup

In your Character's Event Graph, add this test logic:

Event BeginPlay — Give player starting items

```
    BeginPlay

  Survival Inventory Component →

  🔸 Add Item ("Wood", 10) →

  🔸 Add Item ("Stone", 5)
```

This gives the player 10 Wood and 5 Stone when the game starts

💡 **You're Ready!**

> Press **Play** and test! Your character now has an inventory system. Continue reading to set up needs, crafting, and building.

🔧 Complete Setup Guide

This section covers the complete setup process in detail, including all components and DataTables.

Components Overview

The plugin provides five main components that work together:

👜 Survival Inventory Component

Complete inventory system with slots, hotbar, weight management, item usage, and equipment. Requires Items DataTable.

❤️ Survival Needs Component

Manages survival needs: health, hunger, thirst, stamina, temperature, and custom needs. Requires Needs DataTable.

🔨 Survival Crafting Component

Recipe-based crafting system with crafting stations, progress tracking, and recipe unlocking. Requires Recipes DataTable.

🏠 Survival Building Component

Building placement and demolition system with snapping, preview, and validation. Requires Building DataTable.

📦 Survival Storage Component

Add to chests, crates, etc. Provides external storage that players can access. Uses same slot system as inventory.

Required DataTables

Create these DataTables for full functionality:

DATATABLE NAME	ROW STRUCTURE	PURPOSE	ASSIGN TO
`DT_Items`	SurvivalItemData	Define all items (resources, tools, food, etc.)	Inventory Component, Crafting Component
`DT_Needs`	SurvivalNeedData	Define survival needs (hunger, thirst, etc.)	Needs Component
`DT_Recipes`	SurvivalCraftingRecipe	Define crafting recipes	Crafting Component
`DT_Buildings`	SurvivalBuildingData	Define building pieces	Building Component

Component Relationships

Some components need references to others:

- **Crafting Component** needs reference to **Inventory Component** (to consume/give items)
- **Building Component** needs reference to **Inventory Component** (to consume materials)
- **Needs Component** applies effects from items via **Inventory Component**

Event BeginPlay – Link components together

```
BeginPlay
```

```
  Get Survival Inventory Component →


  Survival Crafting Component → Set Inventory Component


  Survival Building Component → Set Inventory Component
```

Do this in BeginPlay to ensure components can communicate

🎒 Inventory System

The inventory system manages items, stacking, weight, hotbar, and item usage.

Item Properties (DataTable)

Each item in your Items DataTable has these properties:

Basic Info

PROPERTY	TYPE	DESCRIPTION
Item ID	Name	Unique identifier (e.g., "Wood", "IronOre")
Display Name	Text	Name shown to player
Description	Text	Item description for tooltips
Category	Enum	None, Resource, Tool, Weapon, Food, Drink, Medicine, Building, Armor, Misc
Rarity	Enum	Common, Uncommon, Rare, Epic, Legendary
Icon	Texture2D	Item icon for UI
World Mesh	StaticMesh	Mesh for dropped items

Inventory Properties

PROPERTY	DEFAULT	DESCRIPTION
Max Stack Size	1	How many can stack in one slot (1 = not stackable)
Weight	1.0	Weight per unit (affects carry capacity)
Base Value	0	Value for trading systems

Consumable Properties

PROPERTY	DEFAULT	DESCRIPTION
Can Use	false	Can this item be consumed?
Use Effects	[]	Array of effects when consumed (see Effect Types below)

Equipment Properties

PROPERTY	DEFAULT	DESCRIPTION
Can Equip	false	Can this item be equipped?
Weapon Damage	0	Damage dealt if weapon
Attack Speed	1.0	Attack speed multiplier
Attack Range	150	Attack range in cm

Effect Types for Consumables

When defining Use Effects for consumable items:

- **RestoreHealth** — Heals health by Value
- **RestoreHunger** — Reduces hunger by Value
- **RestoreThirst** — Reduces thirst by Value
- **RestoreStamina** — Restores stamina by Value
- **DamageHealth** — Damages health by Value (poison, etc.)
- **BuffSpeed** — Speed boost for Duration seconds
- **BuffStrength** — Damage boost for Duration seconds
- **BuffDefense** — Defense boost for Duration seconds

Inventory Blueprint Functions

Adding & Removing Items

`Add Item`

`Add item to inventory. Returns amount actually added (may be less if inventory full or weight limit).`

- `Item ID (Name)`
- `Quantity (int, default: 1)`
- `Amount Added (int)`

`Remove Item`

Remove item from inventory (searches all slots). Returns amount actually removed.

- Item ID (Name)
- Quantity (int)
- Amount Removed (int)

Remove Item From Slot

Remove item from specific slot index.

- Slot Index (int)
- Quantity (int)
- Amount Removed (int)

Query Functions

Has Item

Check if player has at least Quantity of Item ID.

- Item ID (Name)
- Quantity (int, default: 1)
- Has Item (bool)

Get Item Count

Get total count of item across all slots.

- Item ID (Name)
- Count (int)

Get Slot

Get slot data at index (ItemID, Quantity, Durability).

- Slot Index (int)
- Slot (FSurvivalInventorySlot)

Slot Management

Swap Slots

Swap contents of two slots. Use for drag-drop in UI.

- Slot A (int)
- Slot B (int)

Split Stack

Split Amount items from Source to Target slot.

- Source Slot (int)
- Target Slot (int)
- Amount (int)
- Success (bool)

Hotbar Functions

Select Hotbar Slot

Select hotbar slot (0-7 typically).

- Index (int)

Scroll Hotbar

Scroll hotbar selection up or down.

- Scroll Up (bool)

Get Selected Item ID

Get ID of currently selected hotbar item.

- Item ID (Name)

Weight Functions

Get Current Weight / Get Weight Percentage / Is Over Encumbered

Query weight status. Weight Percentage returns 0-1+ (can exceed 1 if overweight).

Item Usage

Use Item

Use/consume item in slot. Applies effects to Needs Component.

- Slot Index (int)
- Success (bool)

Use Selected Item

Use currently selected hotbar item.

- Success (bool)

Drop Item

Drop item from slot into world.

- Slot Index (int)
- Quantity (int)

- Success (bool)

Inventory Events

EVENT	PARAMETERS	WHEN IT FIRES
On Inventory Changed	Slot Index, New Slot Data	Any slot changes
On Item Added	Item ID, Quantity	Items added to inventory
On Item Removed	Item ID, Quantity	Items removed from inventory
On Hotbar Selection Changed	New Index	Hotbar selection changes
On Weight Changed	New Weight	Total weight changes

❤️ Survival Needs System

The needs system manages survival stats like health, hunger, thirst, and stamina with automatic drain, regeneration, and threshold effects.

Need Types

Built-in need types (plus 3 custom slots):

- **Health** — Core survival stat, death at 0
- **Hunger** — Drains over time, affects health when depleted
- **Thirst** — Drains faster than hunger
- **Stamina** — Used for sprinting, regenerates when idle
- **Sleep** — Optional fatigue system
- **Temperature** — Environmental survival
- **Oxygen** — For underwater/space
- **Radiation** — Hazard exposure
- **Sanity** — Mental state
- **Custom1/Custom2/Custom3** — Your own stats

Need Properties (DataTable)

Each need in your Needs DataTable has these properties:

Basic Settings

PROPERTY	DEFAULT	DESCRIPTION
Need Type	Health	Which need this defines
Display Name	-	Name shown to player
Enabled	true	Is this need active in your game?
Max Value	100	Maximum value for this need
Starting Value	100	Value at game start

Drain Settings

PROPERTY	DEFAULT	DESCRIPTION
Drains Over Time	true	Does this need decrease automatically?
Base Drain Rate	0.1	Points lost per second
Sprint Drain Multiplier	2.0	Drain multiplier when sprinting
Jump Drain Multiplier	1.5	Drain multiplier when jumping

Regeneration Settings

PROPERTY	DEFAULT	DESCRIPTION
Regenerates Over Time	false	Does this need increase automatically?
Base Regen Rate	0.0	Points gained per second
Regen Only When Idle	false	Only regen when not moving
Resting Regen Multiplier	5.0	Regen multiplier when resting/sleeping

Threshold Effects

Define effects that trigger at certain levels (e.g., "when hunger below 25%"):

PROPERTY	DESCRIPTION
Threshold Value	Percentage (0-100) where effect triggers
Trigger When Below	true = triggers below value, false = above
Notification Message	Message shown to player
Speed Multiplier	Movement speed modifier (0.5 = 50% speed)
Damage Per Second	Health damage while in this state
Stamina Drain Multiplier	Stamina drain modifier
Vision Effect Intensity	Post-process effect strength (0-1)

Death Settings

PROPERTY	DEFAULT	DESCRIPTION
Fatal At Zero	false	Does reaching 0 kill the player?
Death Message	-	Message shown when this need kills player

Needs Blueprint Functions

Query Functions

Get Need Value / Get Need Max Value / Get Need Percentage

Get current, max, or percentage (0-1) of a need.

- Need Type (ESurvivalNeedType)
- Value (float)

Is Need Critical / Is Need Depleted

Check if need is below 25% (critical) or at 0 (depleted).

- Need Type (ESurvivalNeedType)
- Result (bool)

Modification Functions

Modify Need Value

Add or subtract from need. Positive = add, negative = subtract.

- Need Type (ESurvivalNeedType)
- Delta (float)

Set Need Value

Set need to exact value.

- Need Type (ESurvivalNeedType)
- New Value (float)

Restore Need / Restore All Needs

Restore need(s) to maximum value.

Health Shortcuts

Apply Damage

Shortcut to damage health.

- Damage (float)

Heal

Shortcut to restore health.

- Amount (float)

Get Health / Get Max Health / Is Alive

Shortcuts for health queries.

Stamina Shortcuts

Use Stamina

Consume stamina. Returns false if not enough stamina.

- Amount (float)
- Success (bool)

Has Stamina

Check if have at least Amount stamina.

- Amount (float)
- Has Stamina (bool)

Effect Functions

`Get Speed Multiplier / Get Damage Per Second / Get Stamina Drain Multiplier`

Get combined effects from all active thresholds. Use to modify movement speed, apply damage, etc.

Needs Events

EVENT	PARAMETERS	WHEN IT FIRES
On Need Changed	Need Type, New Value, Max Value	Any need value changes
On Need Threshold Reached	Need Type, Threshold Index	Player enters a threshold zone
On Need Depleted	Need Type	Need reaches 0
On Player Death	-	Player dies (any cause)

🔨 Crafting System

The crafting system handles recipes, ingredients, crafting stations, and production.

Recipe Properties (DataTable)

PROPERTY	DESCRIPTION
Recipe ID	Unique identifier
Display Name	Name shown in crafting menu
Category	Tools, Weapons, Armor, Building, Food, Medicine, Resources, Misc
Required Station	None (hand), Workbench, Forge, Campfire, Anvil, Cauldron, etc.
Ingredients	Array of (Item ID, Amount) pairs
Result Item ID	Item produced
Result Amount	Quantity produced
Crafting Time	Seconds to craft (0 = instant)
Unlocked By Default	Available from start?
Required Level	Level needed to unlock

Crafting Stations

- **None** — Hand crafting, no station needed
- **Workbench** — Basic crafting
- **Forge** — Metal working
- **Campfire** — Cooking
- **Anvil** — Advanced metalwork
- **Cauldron** — Potions/alchemy
- **Saw Mill** — Wood processing
- **Loom** — Fabric/clothing
- **Furnace** — Smelting
- **Chemistry Table** — Advanced crafting

Crafting Blueprint Functions

`Start Crafting`

`Begin crafting a recipe. Consumes ingredients and starts timer.`

- `Recipe ID (Name)`
- `Success (bool)`

`Craft Instant`

`Craft immediately (ignores crafting time). Good for quick hand-crafting.`

- `Recipe ID (Name)`

- Success (bool)

Cancel Crafting

Cancel current crafting (ingredients NOT returned).

Can Craft

Check if can craft: has ingredients, at correct station, recipe unlocked.

- Recipe ID (Name)
- Can Craft (bool)

Has Ingredients

Check if player has all required ingredients.

Set Active Station / Leave Station

Call when player interacts with crafting station.

- Station (ESurvivalCraftingStation)

Get Available Recipes

Get recipes that can be crafted now (unlocked, correct station, has ingredients).

- Recipes (array)

Unlock Recipe

Unlock a recipe for the player.

- Recipe ID (Name)

Crafting Events

EVENT	PARAMETERS	WHEN IT FIRES
On Crafting Started	Recipe ID, Crafting Time	Crafting begins
On Crafting Progress	Recipe ID, Progress (0-1)	Every tick while crafting
On Crafting Completed	Recipe ID	Item crafted successfully
On Crafting Cancelled	Recipe ID	Crafting was cancelled
On Recipe Unlocked	Recipe ID	New recipe learned

🏠 Building System

The building system handles structure placement, preview, validation, and demolition.

Building Categories

- **Foundation** — Base structures
- **Wall** — Vertical structures
- **Floor** — Horizontal surfaces
- **Roof** — Ceiling/roof pieces
- **Door** — Doorways and doors
- **Window** — Window frames
- **Stairs** — Vertical access
- **Fence** — Perimeter structures
- **Furniture** — Interior items
- **Crafting** — Workstations
- **Storage** — Chests, containers
- **Decoration** — Decorative items

Building Blueprint Functions

`Enter Build Mode / Enter Demolish Mode / Exit Build Mode`

Switch between build mode, demolish mode, and normal gameplay.

Select Building

Select building piece to place.

- Building ID (Name)

Update Preview

Update preview position based on camera. Call every frame in build mode.

- Camera Location (Vector)
- Camera Forward (Vector)

Rotate Preview

Rotate building preview by degrees.

- Delta Yaw (float)

Place Building

Place building at current preview location.

- Success (bool)

Demolish Building

Demolish a building actor. Returns some materials.

- Building Actor (Actor)
- Success (bool)

Can Afford Building

Check if player has required materials.

- Building ID (Name)
- Can Afford (bool)

Building Events

EVENT	PARAMETERS	WHEN IT FIRES
On Build Mode Changed	NewMode (ESurvivalBuildMode)	Mode switches
On Building Selected	BuildingID (FName)	Building piece selected
On Building Placed	BuildingID (FName), InstanceID (int32), PlacedActor (AActor*)	Building placed successfully
On Building Demolished	InstanceID (int32), DemolishedActor (AActor*)	Building destroyed
On Placement Valid Changed	bIsValid (bool)	Preview validity changes

Demolition Functions (v1.1.0)

`DemolishBuildingByID`

Demolish building by instance ID (for hybrid HISM system).

- InstanceID (int32)
- Success (bool)

`GetBuildingIDUnderCursor`

Get HISM instance ID under cursor (for hybrid system).

- CameraLocation (FVector)
- CameraForward (FVector)
- InstanceID (int32)

🏗️ Hybrid Building System (v1.1.0)

The Hybrid Building System dramatically improves scalability by using **HISM (Hierarchical Instanced Static Mesh)** for static buildings and **Actors** only for interactive elements.

⚡ **Performance Comparison**

METRIC	ACTORS ONLY	HYBRID SYSTEM
Max Elements	~2,000	**50,000+**
Draw Calls (10k elements)	~10,000	**~50**
Memory (10k elements)	~1GB	**~100MB**
Network Overhead	High	**Low**

How It Works

The system automatically routes building placement:

- **Static Buildings → HISM:** Foundation, Floor, Wall, Roof, Stairs, Ramp, Fence, Pillar, Beam
- **Interactive Buildings → Actors:** Door, Storage, Station, Light, Defense, Decoration

Setup

1. Place **ASurvivalBuildingManager** in your level (one per map)

2. Assign **Building Data Table** to the manager
3. Ensure **bUseHybridSystem = true** on Building Components (default)
4. Done! Building Component auto-finds the manager

ASurvivalBuildingManager Functions

`PlaceBuilding`

`Automatically chooses HISM or Actor based on category. Returns InstanceID (positive for HISM, 0 for Actor).`

- `BuildingID (FName)`
- `Transform (FTransform)`
- `OwnerID (FGuid)`
- `Return: int32 (InstanceID)`
- `OutActor (ASurvivalBuildingPiece*&)`

`PlaceStaticBuilding`

`Force HISM placement (ignores category).`

- `BuildingID (FName)`
- `Transform (FTransform)`
- `OwnerID (FGuid)`

- Return: int32 (InstanceID)

PlaceInteractiveBuilding

Force Actor spawn (ignores category).

- BuildingID (FName)
- Transform (FTransform)
- OwnerID (FGuid)

- Return: ASurvivalBuildingPiece*

RemoveStaticBuilding

Remove HISM instance by ID.

- BuildingID (FName)
- InstanceID (int32)

- Return: bool

RemoveBuildingAtLocation

Remove building closest to location (checks both HISM and Actors).

- Location (FVector)
- Radius (float, default: 50.0f)

- Return: bool

DamageStaticBuilding

Apply damage to HISM instance. Auto-removes at 0 health.

- BuildingID (FName)
- InstanceID (int32)
- Damage (float)
- Return: bool

RepairStaticBuilding

Repair HISM instance.

- BuildingID (FName)
- InstanceID (int32)
- Amount (float)
- Return: bool

Query Functions

IsInteractiveCategory

Check if category spawns as Actor (not HISM).

- Category (ESurvivalBuildingCategory)
- Return: bool

GetStaticBuildingAtLocation

Find HISM instance near a world location.

- Location (FVector)
- Radius (float)
- Return: bool
- OutBuildingID (FName&)
- OutInstanceID (int32&)
- OutData (FSurvivalBuildingInstance&)

GetAllInstancesOfType

Get all instances of a specific building type.

- BuildingID (FName)
- Return: TArray<FSurvivalBuildingInstance>

GetTotalStaticBuildingCount / GetTotalInteractiveBuildingCount

Get total counts for each type.

- Return: int32

Save/Load Functions

GetAllStaticBuildingsForSave

Returns array of all HISM instances for serialization.

- Return: TArray<FSurvivalBuildingInstance>

LoadStaticBuildingsFromSave

Restore all HISM instances from saved data. Clears existing first.

- SavedBuildings (TArray<FSurvivalBuildingInstance>)

ClearAllBuildings

Remove all buildings (both HISM and Actors).

Manager Events

EVENT	PARAMETERS	WHEN IT FIRES
OnStaticBuildingPlaced	InstanceID (int32), BuildingID (FName)	HISM instance added
OnStaticBuildingRemoved	InstanceID (int32), BuildingID (FName)	HISM instance removed
OnStaticBuildingDamaged	InstanceID (int32), CurrentHealth (float), MaxHealth (float)	HISM instance damaged/repaired

FSurvivalBuildingInstance Structure

PROPERTY	TYPE	DESCRIPTION
InstanceID	int32	Unique ID within the HISM pool
BuildingID	FName	Reference to DataTable row
Transform	FTransform	World transform
Health	float	Current health
OwnerID	FGuid	Player who placed it
PlacedTimestamp	int64	Unix timestamp when placed

Customizing Interactive Categories

By default, these categories spawn as Actors:

- Door, Storage, Station, Light, Defense, Decoration

To modify, edit **InteractiveCategories** TSet on the Building Manager:

In Blueprint or C++

```
Building Manager → Interactive Categories → Add/Remove
```

Nanite Support

HISM automatically uses Nanite if your meshes have it enabled. No additional setup required.

> 💡 **Best Practice**
>
> Enable Nanite on all building meshes for best performance with large bases. HISM + Nanite = virtually unlimited static buildings.

Legacy Mode

To disable hybrid system and use Actors for everything (v1.0 behavior):

On Building Component

```
Set bUseHybridSystem = false
```

🌲 Foliage Harvesting System (v1.2.0)

Harvest trees, rocks, bushes, and other resources painted with UE5's **Foliage Tool** — no need to convert them to individual Actors.

> ⚡ **Why Foliage Harvesting?**
>
> UE5 Foliage uses Instanced Static Meshes for extreme performance — thousands of trees with minimal draw calls. This system lets you harvest those instances directly, keeping the performance benefits while adding full survival gameplay.

How It Works

- **Paint foliage normally** with UE5 Foliage Tool
- **Mesh Mappings** connect each foliage mesh to a ResourceID from your DataTable
- Manager **auto-discovers** all foliage ISM components at BeginPlay
- Player hits foliage → **line trace** returns ISM component + instance index
- Manager tracks **per-instance health** (only hit instances are tracked — zero overhead for untouched foliage)
- At 0 HP → instance **scales to zero** (hidden but index preserved) → drops given → respawn timer starts

Setup

1. Place **ASurvivalFoliageManager** in your level (one per map)
2. Assign your **Resources DataTable** (same one used by SurvivalResourceActor)

3 Create **Mesh Mappings**: for each foliage mesh, assign a ResourceID

4 Paint foliage normally with UE5 Foliage Tool

5 In your Character, do a line trace and call **HarvestFromHitResult()**

⚠️ **Mesh Mapping**

Each foliage mesh must be mapped to a ResourceID. For example: SM_Oak_Tree → "Oak_Tree", SM_Rock_Large → "Rock_Large". Multiple mesh variants can map to the same ResourceID.

ASurvivalFoliageManager Configuration

PROPERTY	DEFAULT	DESCRIPTION
ResourcesDataTable	None	Same DataTable as SurvivalResourceActor (FSurvivalResourceData rows)
MeshMappings	[]	Array of (FoliageMesh → ResourceID) pairs
HitSearchRadius	100.0	Search radius for finding foliage near hit location (cm)
bAutoDiscover	true	Auto-scan level for matching foliage at BeginPlay
StumpMeshes	{}	Optional: spawn stump/rubble mesh when resource depleted

Harvesting Functions

HarvestFromHitResult (Recommended)

Pass line trace result directly. Handles everything: tool check, damage, drops, depletion.

- HitResult (FHitResult)
- HarvesterInventory
- ToolUsed (ESurvivalToolType)
- ToolPower (default: 1.0)
- Return: bool (success)

HarvestFoliage (Manual)

Manual harvest with explicit component and index.

- HitLocation (FVector)
- HitComponent (UPrimitiveComponent*)
- HitItemIndex (from FHitResult::Item)
- HarvesterInventory
- ToolUsed
- ToolPower
- Return: bool

DamageFoliage

Apply damage without giving drops (fire, explosions, etc.)

- ISMComponent
- InstanceIndex
- Damage
- Return: bool

Query Functions

IsManagedComponent / GetResourceIDForComponent

Check if a hit component is managed foliage, and get its ResourceID.

CanHarvestAtLocation

Check if foliage at location can be harvested with given tool.

- Location (FVector)
- ToolType (ESurvivalToolType)
- Return: bool

GetInstanceHealthPercent / GetTotalFoliageCount / GetDepletedCount

Query foliage stats for UI or debugging.

Respawn Functions

ForceRespawnInstance / ForceRespawnAll

Force respawn depleted foliage. Automatic respawn uses RespawnTime from DataTable.

Save/Load

GetDepletedInstancesForSave

Returns only damaged/depleted instances. Untouched foliage is NOT saved (minimal save data).

- Return: TArray<FSurvivalFoliageInstance>

LoadDepletedInstancesFromSave

Restore depleted foliage from save. Matches by location (50cm tolerance).

- SavedInstances

Events

EVENT	PARAMETERS	WHEN IT FIRES
OnFoliageHit	InstanceIndex, RemainingHealth, MaxHealth	Foliage instance takes damage
OnFoliageDepleted	InstanceIndex, ResourceID	Foliage fully harvested (HP=0)
OnFoliageRespawned	InstanceIndex, ResourceID	Foliage respawned after timer

FSurvivalFoliageInstance Structure

PROPERTY	TYPE	DESCRIPTION
InstanceIndex	int32	Index in the ISM component
ResourceID	FName	Reference to Resources DataTable row
CurrentHealth	float	Current health
MaxHealth	float	Max health from DataTable
Transform	FTransform	World transform of instance
bIsDepleted	bool	Is this instance harvested?
RespawnTimer	float	Seconds until respawn (-1 = no respawn)

Performance Notes

> 💡 **Zero Overhead Design**
>
> - **Lazy tracking:** Only instances that have been hit are stored in memory
> - **0.5s tick:** Respawn timers update twice per second, not every frame
> - **Minimal save:** Only damaged/depleted instances are saved
> - **Scale-to-zero:** Hidden instances keep their index — no costly ISM rebuilds
> - 10,000 trees in foliage with 50 harvested = only 50 tracked entries

---

📖 How To Guides

How to: Create an Item Pickup

1. Create a new **Actor Blueprint**
2. Add a **Static Mesh** component
3. Add a **Sphere Collision** component
4. Create variables: **Item ID** (Name), **Quantity** (int)

Event ActorBeginOverlap

- Other Actor

Cast to Character →

Get Survival Inventory Component →

Add Item (Item ID, Quantity) →

Destroy Actor

How to: Create a Harvestable Resource (Tree/Rock)

1. Create Actor Blueprint with mesh and health variable
2. Create interface or direct function for "Harvest"
3. On harvest: subtract health, give resource, destroy when depleted

Harvest Function (in Resource Actor)

◆ Damage Amount

○ Harvester (Actor)

```
Health -= Damage Amount →

Get Inventory Component (from Harvester) →

Add Item ("Wood", 2) →

Branch (Health <= 0) → Destroy Actor
```

How to: Setup Hotbar Input

Input Action — Number Keys 1-8

```
   IA_Hotbar1 (Started)

Inventory Component → Select Hotbar Slot (0)
```

Repeat for keys 2-8 with indices 1-7

Input Action — Mouse Wheel

```
    IA_ScrollUp

  Inventory Component → Scroll Hotbar (true)

    IA_ScrollDown

  Inventory Component → Scroll Hotbar (false)
```

How to: Use Selected Item

Input Action — Use Item (E key or Right Click)

```
    IA_Use (Started)

  Inventory Component → Use Selected Item
```

This consumes the item and applies effects to Needs Component

How to: Create a Crafting Station

1. Create Actor Blueprint with mesh and interaction trigger
2. Create variable: **Station Type** (ESurvivalCraftingStation)

3 On interact: call Set Active Station on player's Crafting Component

On Interact (in Crafting Station Actor)

- Interacting Actor

Get Survival Crafting Component →

Set Active Station (Station Type) →

Open Crafting UI Widget

How to: Display Survival Stats HUD

Widget Event Construct — Bind to need changes

Get Owning Player Pawn → Get Survival Needs Component →

Bind Event to On Need Changed

On Need Changed (Bound Event)

- Need Type
- New Value
- Max Value

```
Switch on Need Type →

Health → Set Health Bar Percent (New Value / Max Value)

Hunger → Set Hunger Bar Percent...
```

How to: Setup Hybrid Building System (v1.1.0)

1. Place **ASurvivalBuildingManager** Actor in your level
2. Assign your **Building DataTable** to the manager
3. Ensure **bUseHybridSystem = true** on SurvivalBuildingComponent (default)
4. Component auto-finds manager in BeginPlay - done!

Manual Manager Setup (Optional)

```
   BeginPlay

Get Survival Building Component →

Set Building Manager (your manager reference)
```

Only needed if auto-find doesn't work or you have multiple managers

How to: Place Buildings with Hybrid System

Place Building via Component (Recommended)

Input Action

Building Component → PlaceBuilding →

Branch (Return Value) → Success/Fail

Component auto-routes to HISM or Actor based on category

Place Building via Manager (Direct)

- BuildingID (FName)
- Transform
- OwnerID (FGuid)

Building Manager → PlaceBuilding →

- InstanceID (>0 = HISM, 0 = Actor)
- OutActor (valid if Actor spawned)

How to: Demolish Buildings with Hybrid System

Demolish by Instance ID

Input Action

Building Component → GetBuildingIDUnderCursor (Camera Loc, Camera Fwd) →

Building Component → DemolishBuildingByID (InstanceID)

Works for HISM buildings. For Actor buildings use DemolishBuilding(Actor)

How to: Save/Load Hybrid Buildings

Save Buildings

Building Manager → GetAllStaticBuildingsForSave →

▣ TArray<FSurvivalBuildingInstance> → Serialize to SaveGame

Interactive Actor buildings use standard Actor serialization

Load Buildings

▣ Saved Buildings Array

Building Manager → LoadStaticBuildingsFromSave

Clears existing buildings and recreates from saved data

How to: Damage Buildings with Hybrid System

Apply Damage to HISM Building

◆ BuildingID

◆ InstanceID

◆ Damage Amount

Building Manager → DamageStaticBuilding →

Auto-removes at 0 health

How to: Find Building at Location

Get Static Building At Location

◆ World Location

◆ Search Radius

Building Manager → GetStaticBuildingAtLocation →

◆ Found

◆ OutBuildingID

◆ OutInstanceID

◆ OutData (FSurvivalBuildingInstance)

How to: Harvest Foliage Trees/Rocks (v1.2.0)

1. Place **ASurvivalFoliageManager** in your level
2. Assign **Resources DataTable** and create **Mesh Mappings**
3. Paint foliage normally with UE5 Foliage Tool

4 In Character Blueprint: line trace → call HarvestFromHitResult

Harvest Foliage (Character Blueprint)

```
    IA_Attack (Started)

  Line Trace by Channel (Camera Loc → Camera Fwd * Range) →

  Branch (Hit?) →

  Foliage Manager → IsManagedComponent (Hit Component) →

  Foliage Manager → HarvestFromHitResult (HitResult, Inventory, ToolType, ToolPower)
```

FHitResult::Item contains the ISM instance index automatically

📚 Quick Reference

All Enumerations

ENUM	VALUES
ESurvivalItemCategory	None, Resource, Tool, Weapon, Food, Drink, Medicine, Building, Armor, Misc
ESurvivalItemRarity	Common, Uncommon, Rare, Epic, Legendary
ESurvivalEffectType	None, RestoreHealth, RestoreHunger, RestoreThirst, RestoreStamina, DamageHealth, IncreaseHunger, IncreaseThirst, BuffSpeed, BuffStrength, BuffDefense
ESurvivalNeedType	Health, Hunger, Thirst, Stamina, Sleep, Temperature, Oxygen, Radiation, Sanity, Custom1, Custom2, Custom3
ESurvivalCraftingStation	None, Workbench, Forge, Campfire, Anvil, Cauldron, SawMill, Loom, Furnace, ChemistryTable
ESurvivalRecipeCategory	All, Tools, Weapons, Armor, Building, Food, Medicine, Resources, Misc
ESurvivalBuildingCategory	Foundation, Wall, Floor, Roof, Door, Window, Stairs, Fence, Furniture, Crafting, Storage, Decoration
ESurvivalBuildMode	None, Building, Demolishing

All Structures

STRUCTURE	PURPOSE
FSurvivalItemData	Item definition (DataTable row)
FSurvivalItemEffect	Effect from consuming item
FSurvivalInventorySlot	Single inventory slot (ItemID, Quantity, Durability)
FSurvivalNeedData	Need definition (DataTable row)
FSurvivalNeedState	Runtime need state
FSurvivalNeedThresholdEffect	Effect at need threshold
FSurvivalCraftingRecipe	Recipe definition (DataTable row)
FSurvivalCraftingIngredient	Recipe ingredient (ItemID, Amount)
FSurvivalBuildingData	Building piece definition (DataTable row)
FSurvivalGameSettings	Global game settings

🛠️ Troubleshooting

Items not adding to inventory

1. Is **Items Data Table** assigned to the Inventory Component?
2. Does the Item ID exist in your DataTable? (case-sensitive)
3. Is inventory full? Check **Find Empty Slot**
4. Is weight limit reached? Check **Is Over Encumbered**

Needs not draining

1. Is **Needs Data Table** assigned to the Needs Component?
2. Is the need's **bEnabled** set to true?
3. Is **bDrainsOverTime** set to true?
4. Is **Base Drain Rate** greater than 0?
5. Check **Drain Rate Multiplier** on the component

Crafting not working

1. Is **Recipes Data Table** assigned?

2. Is **Inventory Component** reference set on Crafting Component?
3. Does player have all ingredients? Use **Has Ingredients**
4. Is recipe unlocked? Use **Is Recipe Unlocked**
5. Is player at correct station? Use **Is At Correct Station**

Building placement always invalid

1. Is **Building Data Table** assigned?
2. Is there geometry to place on? (needs floor/foundation)
3. Check **Max Build Distance** setting
4. Check building's **Requires Foundation** property

Events not firing

1. Bind to events in **Event BeginPlay**, not Event Construct
2. Verify component reference is valid before binding
3. Check you're binding to the correct component

💡 **Debug Tips**

- Use **Print String** nodes to verify logic flow
- Check **Output Log** for warnings
- Use Blueprint breakpoints to step through code

> - Bind to events and print when they fire

Support

Email: **support@adrenalinegames.pl**

Documentation: **adrenalinegames.pl/survivalgametemplate**

Survival Game Framework v1.2.0

© 2026 Adrenaline Games. All rights reserved.