Table of Contents

© 2026 Unreal Toolkit. All Rights Reserved.

Introduction

The Mining System by Unreal Toolkit is a fully modular and highly customizable mining system designed for smooth integration into any Unreal Engine project. It comes with a basic inventory and equipment system that can be easily swapped for your own custom systems, ensuring smooth implementation into existing gameplay mechanics.

Mining mechanics are fully customizable, supporting the option for both per-hit harvesting and experience gain as well as on-empty resource collection. The system also includes optional skill levels, which can be toggled on or off to match your game's progression system.

The system provides immediate functionality and prototyping capabilities with six ore nodes, each with a corresponding harvested ore item, as well as three pickaxes and five additional loot items (three crystals and two rocks). These included assets allow you to start using the system right away, while the data tables are designed for easy expansion, making it simple to add custom nodes, ores, tools, and loot.

With intuitive settings and plug-and-play functionality, the Mining System provides a complete foundation for mining mechanics while remaining flexible for customization and expansion. Integration is simple and efficient.

© 2026 Unreal Toolkit. All Rights Reserved.

Quick Start / Installation Guide

Follow these steps to quickly set up and test the Mining System in your project:

1 Migrate the Mining System into Your Project

- In the **Content Browser**, locate the **Mining** folder included in the asset pack.
- Right-click the folder and select **Migrate**.
- In the pop-up window, ensure all necessary assets are selected (**Uncheck the Demo Folder**), then click **OK**.
- Choose your project's **Content** folder as the migration destination.

2 Add Mining Functionality to Your Player Character

- Open your **Player Character Blueprint**.
- In the Components panel, click **Add Component** and search for `AC_Mining`.
- Select `AC_Mining` to add it to your character.

3 Place a Mineable Ore Vein in the World

- In the **Content Browser**, locate `BP_OreVein`.
- Drag `BP_OreVein` into the level.
- In the **Details Panel**, set the **OreVeinID** to determine the type of ore.

4 Place a Pickaxe in the World

- Locate `BP_Item` in the **Content Browser**.
- Drag `BP_Item` into the level.
- In the **Details Panel**, set the **ItemID** to a pickaxe (e.g., "Pickaxe_Basic").

5 Test the Mining System

- Press **Play** to start the game.
- Equip the **Pickaxe** from the inventory.
- Approach the **Ore Vein** and **strike it with the pickaxe**.
- Observe the mining mechanics in action, including **ore depletion, loot drops, and experience gain**.

Congratulations! The Mining System is now active in your project. From here, you can customize settings, expand the data tables, and refine the mechanics to fit your game.

© 2026 Unreal Toolkit. All Rights Reserved.

Core Features

- Fully Modular & Data-Driven System
  - Uses data tables for defining ore types, pickaxes, loot drops, skill levels, and mining mechanics.
  - Easily expandable—add new ore veins, tools, and loot with minimal effort.

- Customizable Mining Mechanics
  - Supports harvesting per hit and/or on empty to fit different gameplay styles.
  - Depletion settings allow ore veins to:
    - Change material upon depletion.
    - Swap to an empty mesh.
    - Fully disappear.

- Skill & Progression System (Optional)
  - Experience gain per hit or when the ore is depleted.
  - Skill levels for mining, which can be enabled or disabled in settings.
  - Pickaxes can have mining level requirements and bonus yield to encourage progression.

- Dynamic Loot System
  - Customizable additional loot drops with rarity-based drop chances.
  - Chance-based spawning allows for rare gems, crystals, or other bonus items.

- Pre-Configured & Customizable Assets
  - 6 ore nodes, each with a corresponding harvested ore item.
  - 3 pickaxes, each with adjustable mining efficiency and skill requirements.
  - 5 additional loot items, including 3 crystals and 2 rock types.
  - Easy to expand with custom assets

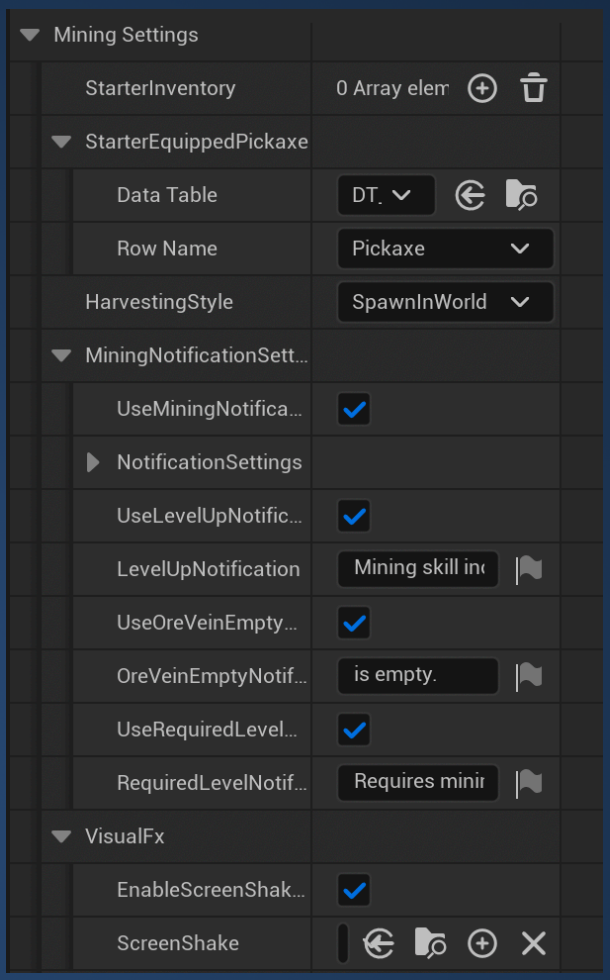
- Visual & Audio Effects for Immersion
  - Niagara FX for mining impact, ore spawn, and level up effects.
  - Customizable sound effects for pickaxe swings, mining hits, ore spawning, and depletion.
  - Optional screen shake for an added sense of impact.

- Easy Integration with Existing Systems
  - Comes with a basic inventory system that can be easily swapped out for your own.

© 2026 Unreal Toolkit. All Rights Reserved.

Customization Options

The Mining System offers extensive customization options, allowing developers to tailor the mechanics, visuals, and overall experience to fit their game. Below is a breakdown of the key settings available in AC_Mining, DT_OreVein, DT_MiningItem, and Inventory Settings.



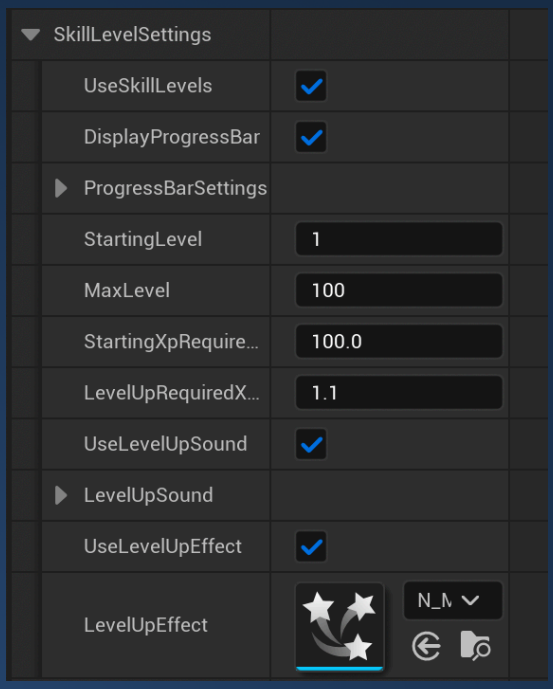
Mining Settings (Mining Component Settings)

The AC_Mining component, added to the player, provides settings for mining behavior, animations, and experience gain.

- Mining Style – Choose between two mining methods:
  - Spawn in World – Ore and additional loot spawn in the world.
  - Auto Add to Inventory – Harvested ore is added directly to the player's inventory.
- Experience Gain Mode – Determines when XP is awarded:
  - Per Hit – XP is awarded with each mining hit.
  - On Empty – XP is awarded when the ore vein is depleted.
- Use Skill Level Requirement – Toggle whether skill levels are required to equip pickaxes.
- Use Customization Settings for Mining Effects – Enables custom VFX for mining.
- Sound & Animation Settings:
  - Set the Mining Swing Sound.
  - Set the Mining Hit Sound Array (Randomized selection per hit).
  - Adjust volume and pitch variation for dynamic audio variation.
  - Select the Mining Hit Animation.
- Effects Settings:
  - Choose the Mining Hit Effect.
  - Enable/disable On Ore Spawn Effect for world-spawned ore.

© 2026 Unreal Toolkit. All Rights Reserved.

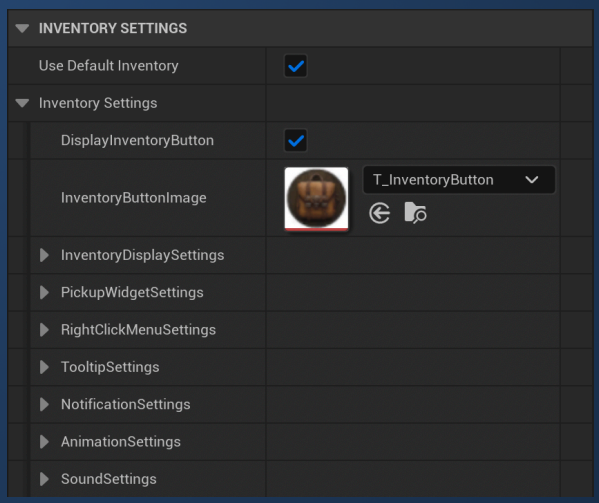
Skill Level Settings (AC_Mining)



- Toggle whether or not to use the skill level system
- Toggle the use of the progress bar
  - Set progress bar settings to determine location, bar color, text settings, icon, etc.
  - Set the starting level for the mining skill, as well as the maximum level
  - Set the XP required to level up
    - The experience system works with a multiplier to increase the amount of XP needed to level up with each level gained. Customize the multiplier to adjust how much the amount of XP needed for the next level changes.
  - Toggle the use of a level up sound effect, and select custom sound effects from a drop down menu
  - Toggle the use of the level up visual effect, and select custom visual effects from a drop down menu of niagara effects

The Progress Bar widget **WBP_MiningSkillBar** updates itself and can be placed anywhere. You can place it in any custom widget and it will update itself.

Inventory Settings (AC_Mining)

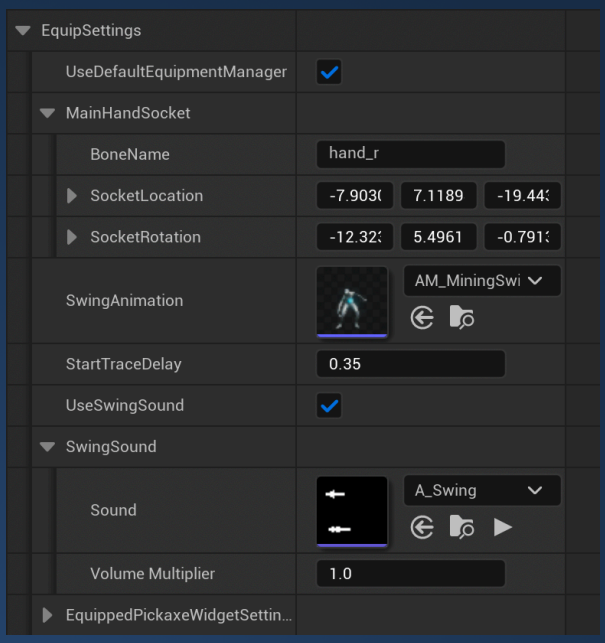


The mining system uses a default inventory system that can be swapped out for your own custom inventory system if desired. The default inventory system is fully customizable.

- Toggle whether or not to use the default inventory system
- Toggle the use of the Inventory Button widget and select an icon for the inventory button.
- Notifications:
  - Customize Pickup Notification Text (e.g., "+1 [ItemName]").
  - Adjust Notification Duration.
- Full customization settings for inventory widget aesthetics
  - Pickup item from world widget
  - Inventory widget
  - right-click menu
- Set inventory notification settings, determine notification text, duration, and aesthetic
- Pickup & Drop Settings
  - Enable/disable Pickup Animation when collecting items, and set custom animations from a drop down menu.
  - Enable/disable Inventory Sound Effects for adding/removing items, and set custom sound effects from a drop down menu.

© 2026 Unreal Toolkit. All Rights Reserved.

Equipment Settings (AC_Mining)

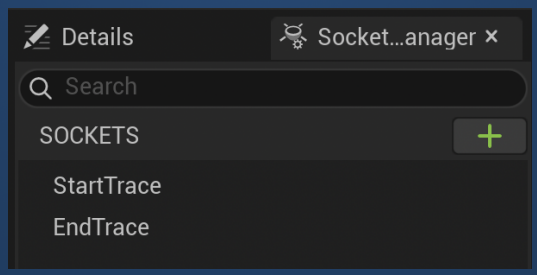


- Toggle whether or not to use the default equipment system
- Set the main hand socket data for your custom character mesh - this is where the pickaxe will visibly appear on your character.
  - Set the bone name
  - Set the local location and rotation for the socket to fine tune the positioning of the equipped pickaxe in your character's hand.
- Set the pickaxe swing animation
  - Set the delay from the start of the swing animation to the start of the pickaxe hit trace, allowing you to fine tune the swing mechanics
- Toggle the use of a pickaxe swing sound and set the swing sound
- Toggle whether or not the equipped pickaxe widget should hide itself if no pickaxe is equipped.
  - Fully customize the aesthetics of the widget

The equipped pickaxe widget **WBP_EquippedPickaxe** updates itself and can be placed anywhere. You can place it in any custom widget and it will update itself and remain functional.

Preparing Custom Pickaxe Meshes

The Mining system uses sockets added to the pickaxe meshes to determine where to detect if the pickaxe hits an ore vein during the swing animation.

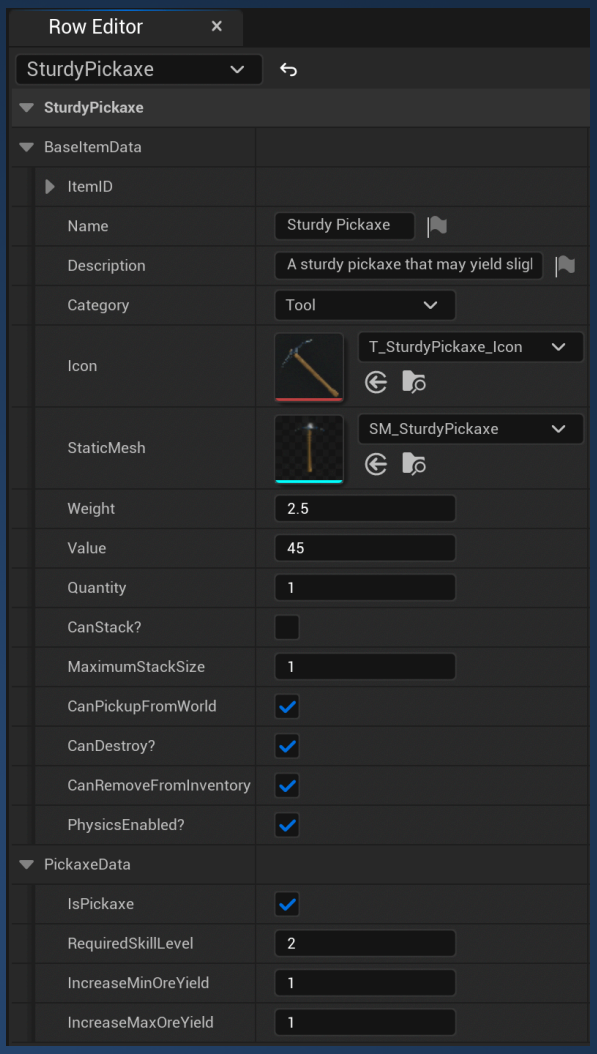


- Open your pickaxe mesh
- In the details panel, click the Socket Manager tab and click + to add a socket
  - The name of the sockets **must match exactly** to the socket names shown
  - **StartTrace**
  - **EndTrace**
- Position the sockets where you want them on your mesh. The very tip of the pickaxe, as shown in the example.

© 2026 Unreal Toolkit. All Rights Reserved.

Ore, Loot Items, and Pickaxes

The DT_MiningItem data table controls ore and item data, as well as pickaxes and their stats. To add items, **open DT_MiningItems and click +ADD** to add a new row. Fill in the item data to add your item to the system.



Base Item Data (DT_MiningItem)

- ItemID – A reference to the data table row.
- Name - The item's name
- Description - A description of the item. Will appear in item tooltips and other UI.
- Icon - The icon/image for the item
- Static Mesh - The item's mesh
- Weight - The physical weight of the item
- Value - The item's monetary value
- Quantity - The quantity that 1 instance represents
- CanStack? - Toggles if the item can stack
- MaximumStackSize - Sets the maximum stack size
- CanPickupFromWorld - Enables/disables the ability to pick up the item from the world
- CanDestroy - Enables/disables the ability to destroy the item from the inventory
- CanRemoveFromInventory - Enables/disables the ability to remove the item from the inventory by dropping or destroying it.
- PhysicsEnabled? - Toggles physics for the item when it's in the world.

Pickaxe Data (DT_MiningItem)

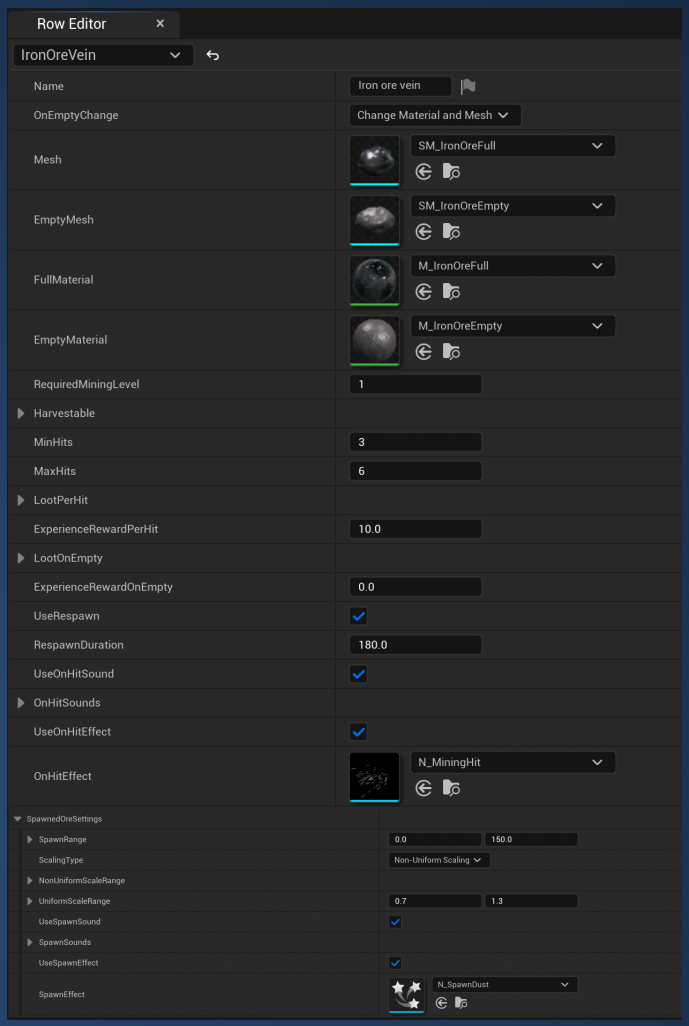
- RequiredSkillLevel - Sets the minimum skill level needed to equip the pickaxe
- IncreaseMinOreYield - Increases the minimum amount of ore yielded from the vein by the set amount
- IncreaseMaxOreYield - Increases the maximum amount of ore yielded from the vein by the set amount

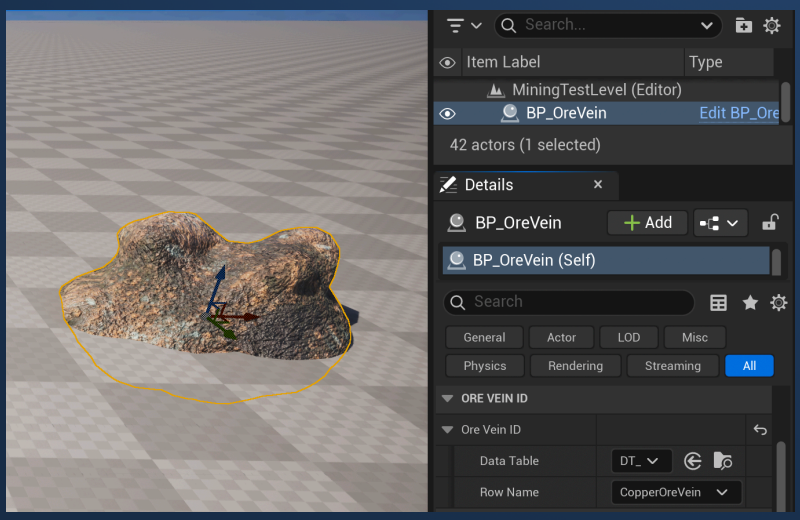
© 2026 Unreal Toolkit. All Rights Reserved.

Ore Veins (DT_OreVein)

The DT_OreVein data table controls the properties of different ore nodes.

- Set the name of the ore vein
- Set the depletion behavior with OnEmptyChange
  - Change Material - Changes the ore's material to a depleted version
  - Change Mesh - Replaces the mesh with a depleted version
  - Destroy Mesh - Removes the ore node upon depletion
  - No Change - Keeps the ore node visually intact.
- Set the harvestable item with the ore Item ID.
- Skill Level Requirement – Set a required Mining Level for the ore.
- Min/Max Yield Per Hit – Define the amount of ore collected per mining hit.
- Additional Loot System:
  - Enable/disable Chance-Based Additional Loot.
  - Define a list of extra items that may drop, each with:
    - Drop Chance (%).
    - Min/Max Quantity.
- Set a respawn duration, after the determined time the ore will respawn
- Toggle the use of hit sounds, and select the hit sounds for the ore
- Toggle the use of hit effects, and select a niagara effect to play at the impact point of the hit
- Customize the spawned ore settings
  - Set a range for how far the ore can spawn from the hit point
  - Set scaling style for the spawned ore
  - Set sound and effects for ore spawning in the world





Add Ore To The World

• Drag BP_OreVein into the world
• With it selected, find ORE VEIN ID in the details panel
• Select the row name of the ore you're adding

The ore will automatically update based on the data in DT_OreVein

© 2026 Unreal Toolkit. All Rights Reserved.

Integration / Advanced Usage

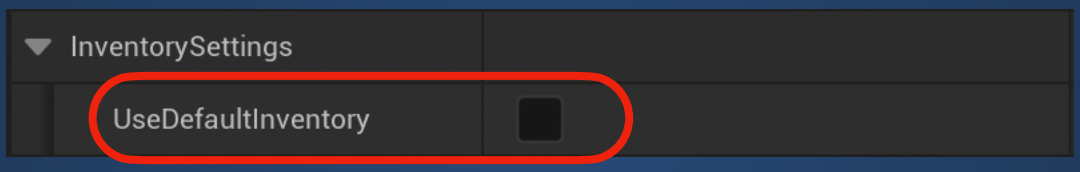
The Mining System is designed to be modular and easy to integrate with other inventory and/or equipment systems. If you are using a custom inventory system, you can override the built-in functions to ensure seamless integration.

Integrating a Custom Inventory System

By default, the mining system includes AC_DefaultInventory, a basic inventory setup designed for easy testing and quick implementation. However, if you are using your own inventory system, you will need to override a few key functions to replace the default item management logic with your own.

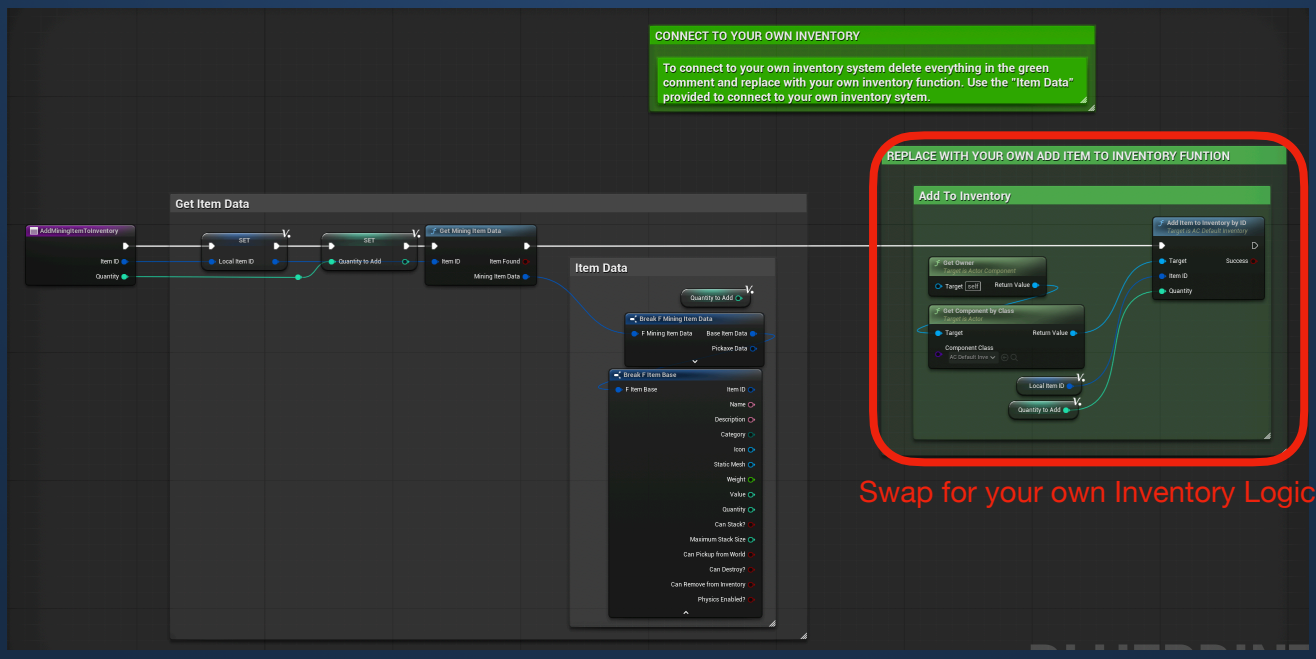
Overriding The Inventory

In AC_Mining there are the following inventory related functions that you can easily replace with your own Inventory functions. Open each of the functions below in AC_Mining, and replace the logic with your inventory functions. If you are integrating with your own custom inventory, ensure to uncheck “UseDefaultInventory” found in the Inventory Settings in AC_Mining. (From your player character blueprint, select AC_Mining, and find the settings in the details panel.)



The Inventory functions are set up to expose the item data for easy integration. Replace the green commented section with your own inventory function. Use the exposed Item Data to hook into your inventory function.

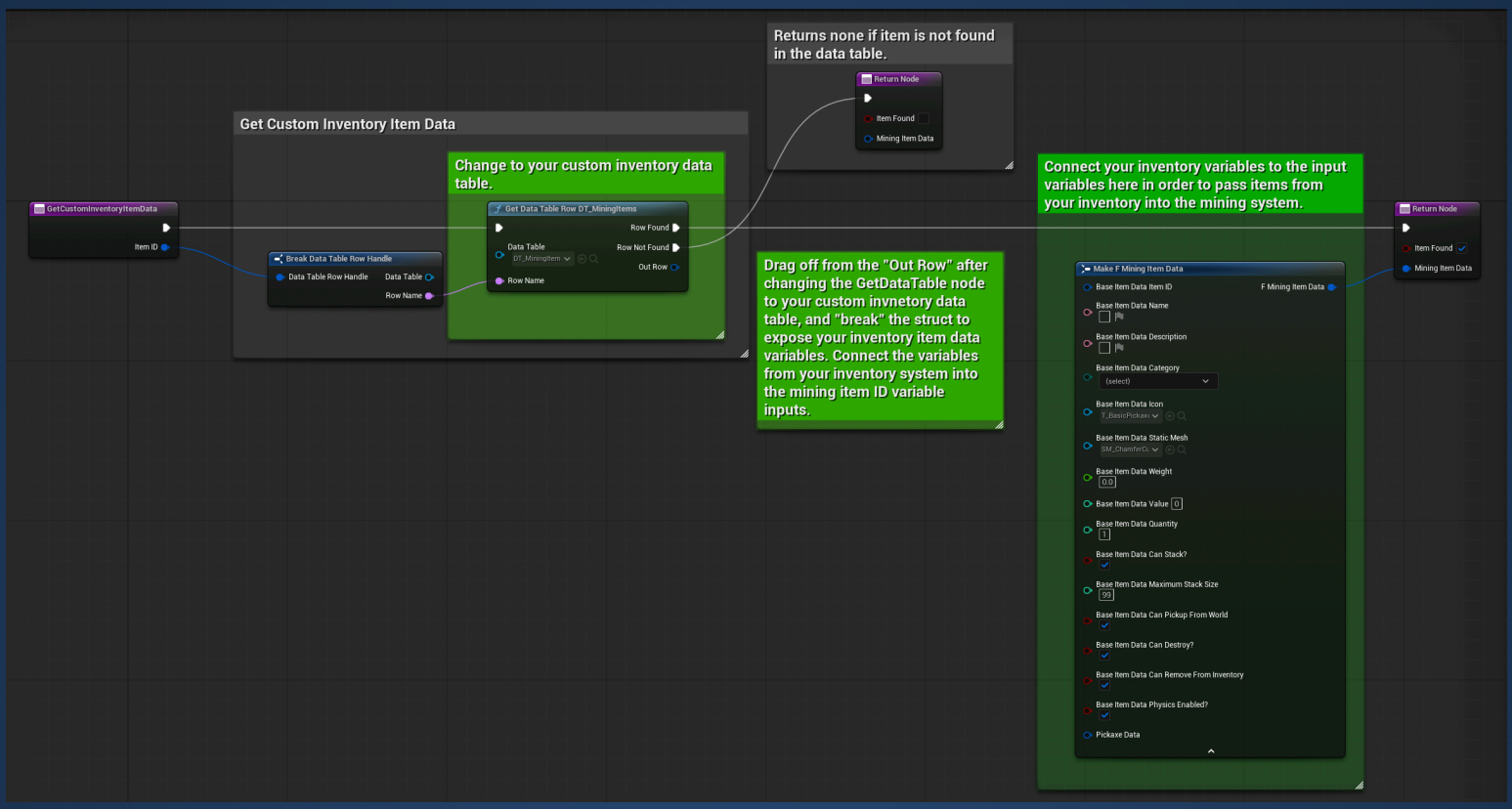
Do this for the two inventory functions:
AddMiningItemToInventory
RemoveMiningItemFromInventory



© 2026 Unreal Toolkit. All Rights Reserved.

To make the mining system able to read item data from your custom inventory, you must edit the **GetCustomInventoryItemData** function found in **BPFL_CustomInventoryIntegration.**

- Open **BPFL_CustomInventoryIntegration**
- Select the function **GetCustomInventoryItemData**



- Change the Get data table row node to your custom inventory data table.
- Drag off from the out row and “break” to expose your custom item data.
- Connect your custom item data variables to the corresponding variables in the “MakeMiningItemData” section of the function.

This allows you to set your own custom items as loot items. After doing this you will still be able to use DT_MiningItems to add new ore/items/pickaxes to the system.

After editing those **3 inventory functions** the mining system will work with your inventory by passing item data from DT_MiningItem into your custom inventory system, and item data from your custom inventory system into the mining system.

© 2026 Unreal Toolkit. All Rights Reserved.

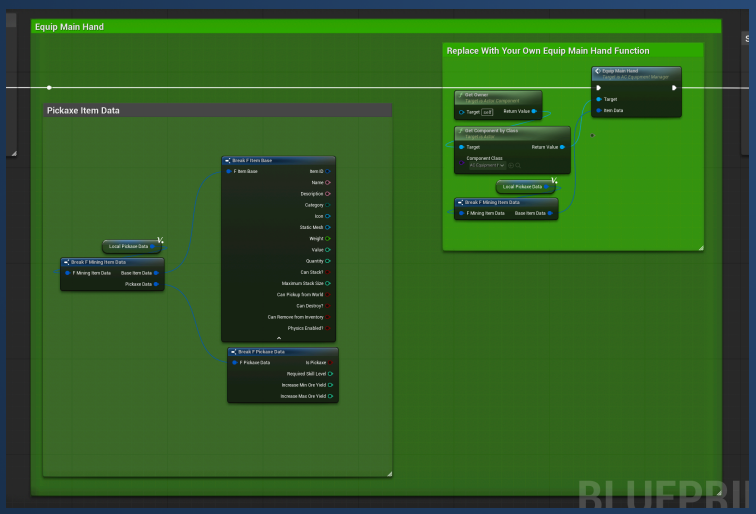
Integrating a Custom Equipment System

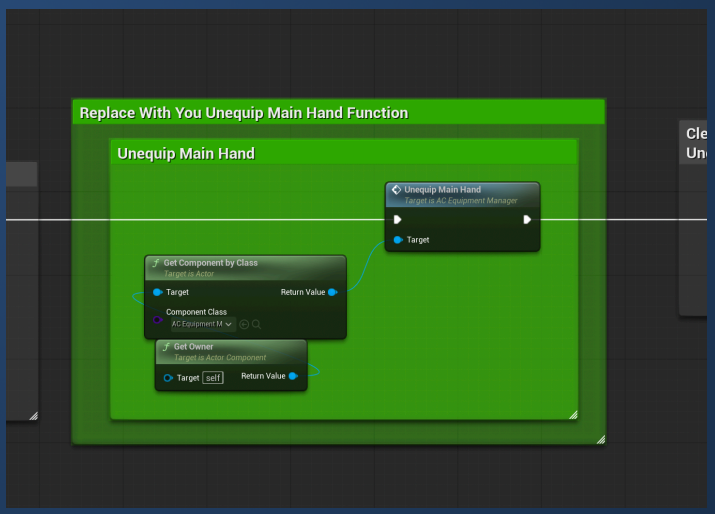
By default, the mining system includes AC_DefaultEquipmentManager, a basic equipment system designed for easy testing and quick implementation. However, if you are using your own equipment system, you will need to override a few key functions.

Overriding The Equipment System

In AC_Mining there are the following equipment related functions that you can easily replace with your own equip functions. Open each of the functions below in AC_Mining, and replace the logic with your equip functions. If you are integrating with your own custom equipment system, ensure to uncheck “UseDefaultEquipmentManager” found in the Equipment Settings in AC_Mining. (From your player character blueprint, select AC_Mining, and find the settings in the details panel.)

The EquipPickaxe and UnequipPickaxe functions are set up for you to edit. Open AC_Mining, and edit only the green commented areas of the EquipPickaxe and UnequipPickaxe functions. The mining system handles the logic for marking “PickaxeEquipped”, you’re overriding the part of the function that visibly attaches the item mesh to your player, and marking something like “MainHandEquipped” in your custom equipment system. This will allow the mining system to communicate with your custom equipment system whenever the mining system calls its equip/ unequip functions.

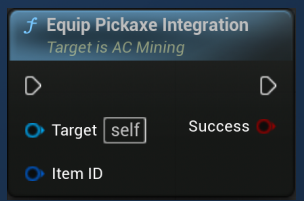


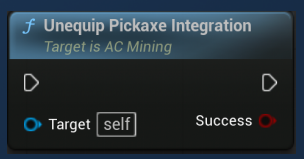


In your own custom equipment system, add the “EquipPickaxe_Integration” function into your equip function. This function will check if the item is a pickaxe, and if so it will mark “PickaxeEquipped” in the mining system. You can add this at the end of your custom equip function.

In your unequip function, add the “UnequipPickaxe_Integration”. This function will check if the item being unequipped is a pickaxe, and if so it will uncheck “PickaxeEquipped” in the mining system.

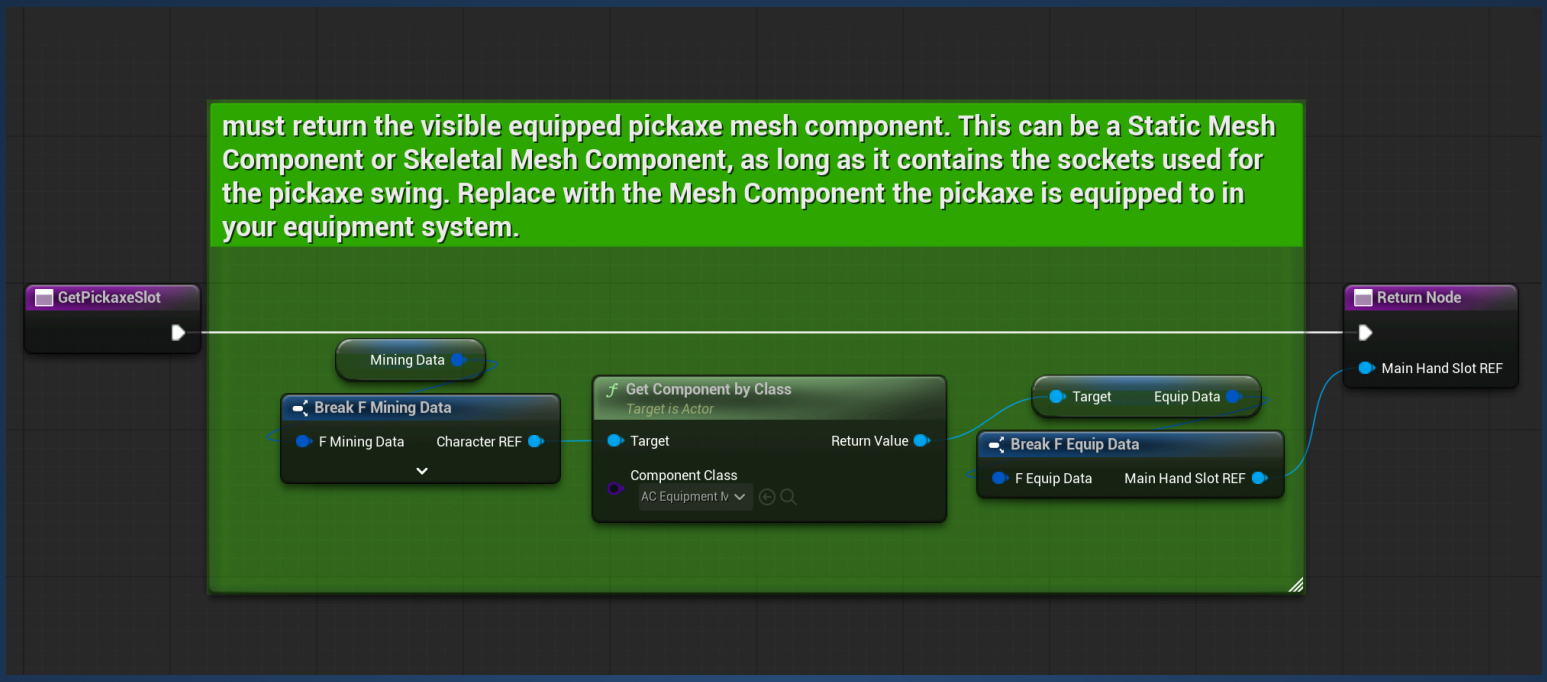
For both of these functions, feed in the ItemID (Data table row) of the item being equipped or unequipped.





© 2026 Unreal Toolkit. All Rights Reserved.

The Mining system sometimes needs a reference to the pickaxe slot, or main hand slot of the player character. To do so it uses an overridable function “GetPickaxeSlot”, that you can edit to return a reference to the slot being used from your custom equipment system. This reference can be either a static mesh component or a skeletal mesh component.



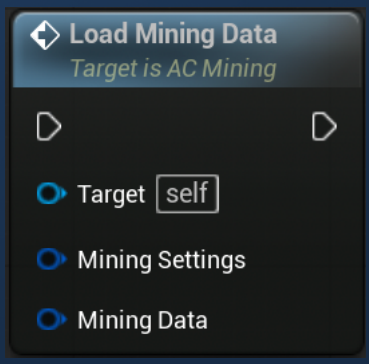
Editing these equipment functions and adding the integration functions to your custom equipment system allows the Mining System to integrate seamlessly with your custom equipment system. This enables your equipment system to manage equipped and unequipped states independently while still allowing the Mining System to communicate with it when needed.

Integrating Save/Load Functions With Your Save System

The mining system uses two variables to store all data. MiningSettings, and MiningData. AC_Mining includes a SaveMiningData and LoadMiningData function to make it easy to set up with your save system.

In your Save Game Blueprint, in your save function, call the SaveMiningData function, and promote the two outputs (MiningSettings, MiningData) to variables within your save game blueprint. Make sure that in the save game function, you are setting these variables from the output of the SaveMiningData function. This will store the data in your save game file.

In your Load Game function, after accessing the save game file, pull off of it and drag the MiningData and MiningSettings variables into the LoadMiningData function. This will ensure that the mining data and settings are loaded from your saved data.



© 2026 Unreal Toolkit. All Rights Reserved.

Quick Reference Guide

Actor Components	
AC_Mining	The component that adds mining functionality to the player character. Add it to your player character blueprint.
AC_DefaultInventory	The component that adds inventory functionality to the player character. AC_Mining will add this automatically if you have “UseDefaultInventory” set to True.
Blueprints	
BP_OreVein	The blueprint that represents Ore Veins (Mining Nodes) in the world. Drag into the world and set the OreVeinID to set the vein.
BP_MiningItem	The blueprint that represents Items in the world. Drag into the world and set the ItemID to set the item.
Structs	
F_OreVeinData	The struct that contains Ore Vein data, used for DT_OreVein
F_MiningItemData	The struct that contains Item Data. Used for DT_MiningItems
Data Tables	
DT_OreVein	The data table that allows you to add custom Ore Veins (Mining Nodes), and/or edit existing ones.
DT_MiningItems	The data table that allows you to add custom Items (Pickaxes or Ore, etc.) and/or edit existing ones.

© 2026 Unreal Toolkit. All Rights Reserved.

Support & Contact

For any questions, issues, or feedback, please contact us at:

support@unreal-toolkit.com

Explore all of our assets on: Unreal-Toolkit.com

© 2026 Unreal Toolkit. All Rights Reserved.