FoxSettings

Data-Driven Game Settings Menu for Unreal Engine

User Guide
Version 1.3.0
Supported Engine: UE 5.4, 5.6 - 5.7

Alchemy Fox Studio

Table of Contents

Installation

Option A: Via Unreal Engine Tab

1. Open the Epic Games Launcher.
2. Go to the Unreal Engine tab.
3. Click Marketplace and search for FoxSettings.
4. Click Install to Engine and select your engine version.
5. Open your project — FoxSettings is available under Edit → Plugins.

Option B: Via Fab Library

1. Open the Epic Games Launcher.
2. Click Fab in the left sidebar → My Library.
3. Search for FoxSettings.
4. Click Install Plugin and choose the engine version.
5. Open your project — the plugin is already enabled.

Option C: Manual Installation

1. Copy FoxSettings/ into YourProject/Plugins/.
2. Right-click your .uproject file → Generate Visual Studio project files.
3. Open project, go to Edit → Plugins, enable FoxSettings, restart.
4. All nodes appear under FoxSettings|Category when you right-click in any Blueprint graph.

Supported: UE 5.4, 5.6 - 5.7

Quick Start

Five steps to a working settings menu — all in Blueprint, no C++ needed.

1. **Get Fox Settings** — pure node, returns the subsystem from any Blueprint.
2. **Create a DataAsset** — Content Browser → Miscellaneous → Data Asset → FoxSettingsTabData. Add entries (SettingID, DisplayName, WidgetType, CVarName).
3. **Register Tab** — in BeginPlay: Get Fox Settings → Register Tab → [Your DataAsset].
4. **Toggle Fox Settings Menu** — bind to ESC. Creates widget, shows cursor. Call again to close.
5. **Apply / Revert / Reset** — wire the three buttons in your menu widget.

■ Settings are defined in DataAssets and bound to Console Variables — no code needed.

Step-by-Step: Setting Up Your Menu

This walkthrough takes you from a fresh project to a fully working settings menu.

Step 1: Enable the Plugin

Open your project → Edit → Plugins → search FoxSettings → enable → Restart Now.

Step 2: Create Your First DataAsset

Content Browser → right-click → Miscellaneous → Data Asset → choose FoxSettingsTabData. Name it DA_FoxSettings_Graphics. Double-click to open.

Step 3: Configure the Tab

Set Tab Display Name: "Graphics" and Tab Sort Order: 0 (lower = further left).

Step 4: Add Setting Entries

Click + under Settings. Example entries:

Step 5: Register the Tab in Blueprint

PlayerController → Event BeginPlay:

Step 6: Add the Menu Toggle

InputAction ESC (Pressed) → Toggle Fox Settings Menu

Step 7: Wire Apply / Revert / Reset

Default WBP_FoxSettingsMenu has these wired. For custom menus:

Step 8: Add More Tabs

Repeat Steps 2-5 for Audio, Controls, Gameplay, Accessibility. Tab Sort Order: 0, 1, 2...

Step 9: Test

Click Play → press ESC → change settings → Apply → values persist after restart.

Step-by-Step: Personalizing Your Menu

Customize colors, fonts, widget appearance, and build your own widget Blueprints.

Step 1: Create a Style DataAsset

Content Browser → Data Asset → FoxSettingsStyleData. Name: DA_FoxSettings_MyStyle.

Step 2: Customize Colors

BackgroundColor — Menu background. AccentColor — Active tab, highlights. TextColor — Labels. SliderBarColor / SliderHandleColor — Slider track + thumb. TabActiveColor / TabInactiveColor — Tab states. FocusHighlightColor — Gamepad outline. ButtonColor / ButtonHoverColor — Actions. RowEvenColor / RowOddColor — Zebra stripes.

Step 3: Customize Fonts

TitleFont — Menu title. LabelFont — Setting labels. ValueFont — Values. SectionHeaderFont — Dividers.

Step 4: Customize Layout

RowHeight (default: 48). LabelWidth (250). WidgetWidth (300). RowSpacing (2). ContentPadding.

Step 5: Apply the Style

Get Fox Settings → Set Style Data → DA_FoxSettings_MyStyle

Step 6: Custom Widget Blueprints (Advanced)

Create Widget Blueprint → parent: FoxCVarSlider / FoxCVarToggle / FoxCVarDropdown / etc. Design layout. Override Refresh From CVar to update visuals.

Step 7: Register Custom Widget Classes

Get Default Widget Class Overrides → Break → Replace SliderClass → Make → Pass to CreateWidgetsForTab

Step 8: Customize Keybind Display Names

DataAsset: FoxKeybindOverrideData → set ActionName, DisplayName, Category, SortOrder, bHidden. Get Fox Settings → Set Keybind Override Data → [DataAsset]

How It Works

Every setting is a CVar binding: a widget reads/writes an engine Console Variable. On Apply, values save to INI.

Widget Types (9 Types)

Slider

Float/Int range. SliderMin, SliderMax, SliderStep, bSliderIsInteger, SliderValueSuffix.

Toggle

On/Off checkbox. CVar "0" or "1".

Switch

Visual slide switch. Same as Toggle, different look. Uses Switch_On/Switch_Off textures.

Dropdown

Multiple options. Array of {DisplayName, CVarValue}.

Resolution Picker

Auto-detects available resolutions. No CVar needed.

Keybind Row

3-slot rebinding: KB Primary, KB Secondary, Gamepad.

Button

Action trigger. Fires OnButtonPressed delegate.

Header

Visual section divider. Groups settings with same Section field.

Spacer

Empty space between settings.

Conditional Visibility

RequiredCVar — Only show if CVar exists. Example: "r.NGX.DLSS.Enable" → hides if no DLSS plugin.

RequiredCVarValue — Only show if CVar has this value. "r.NGX.DLSS.Enable"="1" → show only when DLSS on.

bDesktopOnly / bConsoleOnly — Platform filtering. Desktop = PC/Mac. Console = PS/Xbox/Switch.

Settings Menu Examples

Graphics Tab

Resolution — ResolutionPicker. No CVar needed. Auto-detects monitor resolutions.

Window Mode — Dropdown. r.FullScreenMode: "Fullscreen"→"0", "Windowed Fullscreen"→"1", "Windowed"→"2".

FPS Limit — Dropdown. t.MaxFPS: "30"→"30", "60"→"60", "120"→"120", "Unlimited"→"0".

V-Sync — Toggle. r.VSync. Default: "1".

Shadow Quality — Dropdown. sg.ShadowQuality: "Low"→"0" through "Ultra"→"3".

Brightness — Slider. r.TonemapperGamma. Min=1.0, Max=3.0, Step=0.1.

Audio Tab (Custom Delegates)

Master Volume — Slider. bUseCustomDelegate=true. Min=0, Max=100, bSliderIsInteger=true, Suffix="%".

Music Volume — Slider. Same pattern. Default=80.

Sound Effects — Slider. Same pattern. Default=100.

Dialogue — Slider. Same pattern. Default=100.

Ambient — Slider. Same pattern. Default=70.

Mute on Focus Loss — Toggle. bUseCustomDelegate=true. Default="1".

■ Bind to OnCustomSettingChanged → Switch on SettingID → divide value by 100 for 0.0-1.0 → route to Sound Mix system. Mute on Focus Loss uses FApp::SetUnfocusedVolumeMultiplier.

Accessibility Tab

Colorblind Mode — Dropdown. fox.ColorDeficiencyType: "None"→"0", "Deuteranopia"→"1", "Protanopia"→"2", "Tritanopia"→"3".

Colorblind Strength — Slider. fox.ColorDeficiencySeverity. Min=0, Max=10, Step=1.

■ Auto-calls SetColorVisionDeficiencyType on Apply. No extra code needed.

Keybinding System

UE4: Auto-reads Project Settings → Input → Action/Axis Mappings. UE5: Scans Input Mapping Contexts.

No DataAsset needed for basic keybindings — they are auto-imported.

Override Display Names — DataAsset FoxKeybindOverrideData: "IA_Jump"→"Jump", category "Movement".

How Rebinding Works — Click slot → "Press any key..." → press key → assigned. ESC cancels.

Conflict Resolution — If key already bound → swap dialog. Accept = swap, Cancel = keep old.

Saving — Only changed bindings save to INI.

Gamepad Support

Input	Action
D-Pad Up/Down	Navigate settings
D-Pad Left/Right	Change value (slider/dropdown)
A Button	Confirm / Toggle checkbox
B Button	Back / Close menu
LB / RB	Previous / Next tab
Start	Apply settings
Select	Reset to defaults

■ Auto-detects mouse ↔ gamepad. Focus highlight appears in gamepad mode, disappears on mouse move.

Upscaling (DLSS / FSR / TSR)

Works with any upscaling plugin through CVars. Use RequiredCVar to auto-hide when plugin missing.

Upscaler	Enable CVar	Quality CVar
TSR (UE5)	r.AntiAliasingMethod (=4)	r.ScreenPercentage

DLSS	r.NGX.DLSS.Enable	r.NGX.DLSS.Quality
FSR 3	r.FidelityFX.FSR3.Enabled	r.FidelityFX.FSR3.QualityMode
XeSS	r.XeSS.Enabled	r.XeSS.Quality

CVar Browser (Editor Tool)

Open: Window → Tools → FoxToolkit CVar Browser

Search — Type to filter by CVar name or description. Real-time filtering.

Category Filter — Dropdown: Rendering, Scalability, Audio, Streaming, etc.

Detail Panel — Shows CVar name, type, current value, default, engine description.

Test Value — Change a CVar temporarily in the editor. Not saved.

Copy Name — Copy CVar name to clipboard. Paste into your DataAsset.

Add to Tab — Copies a ready-to-paste FFoxSettingEntry to clipboard.

■ Only shows ~200 game-relevant CVars — not all 5000+ internal engine CVars.

Styling / Reskinning

DataAsset FoxSettingsStyleData: colors, fonts, row height, label/widget width, spacing, padding.

Get Fox Settings → Set Style Data → [Your Style DataAsset]

All widget C++ classes can be subclassed in Blueprint (Widget Blueprint → parent FoxCVarSlider etc.).

Live Style Updates

Call Apply Style To Menu after modifying StyleData properties at runtime (e.g. TextScale, HighContrast). This re-applies all colors, fonts, and row backgrounds to the open menu without closing it.

Localization

All strings use LOCTEXT (namespace "FoxSettings"). Localization Dashboard → Gather → Translate → Compile.

Saving & Loading

How Settings Are Saved

1. When the player clicks Apply, your Blueprint calls Apply All Fox Settings.

2. The subsystem iterates all CVar bindings and writes their current values to the engine Console Variables.

3. Changed values are written to the INI file: Saved/Config/[Platform]/FoxSettings.ini

On Windows with the editor, this is typically: Saved/Config/WindowsNoEditor/FoxSettings.ini

4. Only settings that differ from their DefaultValue (in the DataAsset) are saved. Unchanged settings rely on engine defaults.

5. Keybindings are saved separately via the engine InputSettings system.

How Settings Are Loaded

1. On game startup, the UFoxSettingsSubsystem initializes automatically (GameInstanceSubsystem).

2. It reads FoxSettings.ini and restores all saved CVar values.

3. When you call Register Tab, the subsystem creates CVar bindings for each setting entry.

4. Each binding checks: Is there a saved value in the INI? If yes → use it. If no → use the DefaultValue from the DataAsset.

5. The CVar is set immediately — the engine applies the setting on load.

INI File Format

The INI file uses a simple key-value format under the section [FoxSettings]:

Saving & Loading (continued)

Revert vs. Reset

Revert All Fox Settings — Undoes all changes since the last Apply. Restores the state from when the menu was opened. Use for a "Cancel" button.

Reset All Fox Settings To Defaults — Resets every setting to its DefaultValue from the DataAsset. This does NOT auto-save — the player must still click Apply.

Custom Delegate Settings

Settings with bUseCustomDelegate=true do NOT write to a CVar. Instead, they fire the OnCustomSettingChanged delegate with SettingID and Value. You handle saving in your own game system (e.g. Sound Mix, game-specific variables).

These settings still save their value to FoxSettings.ini — the INI just stores the string, but does not apply it to a CVar.

Keybinding Persistence

Keybindings use the engine-native InputSettings system for saving. When Apply Keybindings is called (automatically by Apply All Fox Settings), changed keybindings are written to: Saved/Config/[Platform]/Input.ini

Unchanged keybindings are NOT saved — they use the engine defaults from Project Settings → Input.

Platform Support

Platform	Config Path
Windows	Saved/Config/WindowsNoEditor/
Linux	Saved/Config/LinuxNoEditor/
Mac	Saved/Config/MacNoEditor/
PS5	Saved/Config/PS5/
Xbox Series	Saved/Config/XSX/
Switch	Saved/Config/Switch/

■ Settings are per-user, per-platform. No cloud sync is provided — use your own SaveGame system for cloud saves.

Category Reference

1. Subsystem Access (1 Node)

Get Fox Settings

Returns UFoxSettingsSubsystem from any world context. Pure node — no setup needed.

Parameter	Type	Description
WorldContext	UObject*	Auto-filled by engine (self reference)
ReturnValue	UFoxSettingsSubsystem*	The settings subsystem instance

■ Drag off return value to access Register Tab, Apply Settings, Toggle Menu, etc.

2. Menu Control (5 Nodes)

Show Fox Settings Menu

Create menu widget, add to viewport, set GameAndUI input mode, show mouse cursor.

Hide Fox Settings Menu

Remove menu from viewport, restore GameOnly input mode, hide cursor.

Toggle Fox Settings Menu

Show if hidden, hide if shown. One-line menu integration.

Is Fox Settings Menu Visible

Pure check — is the menu widget currently in the viewport?

Apply Style To Menu

Re-apply all style changes (colors, fonts) to the currently open menu widget. Call after modifying StyleData at runtime (e.g. TextScale, HighContrast).

3. Settings Read/Write (6 Nodes)

Set Fox Setting

Set single setting by SettingID. Updates CVar immediately but does NOT save to INI.

Get Fox Setting

Read current value of a setting as string.

Apply All Fox Settings

Apply all pending changes. Writes CVars, saves INI, fires delegates.

Revert All Fox Settings

Undo all changes since last Apply. Restores CVar values.

Reset All Fox Settings To Defaults

Reset every setting to DefaultValue from DataAsset.

Has Pending Fox Settings Changes

True if any setting was changed but not yet applied. Pure.

Refresh All Bindings

Refresh all widget values from current CVar state. Called automatically by Apply. Call manually to sync UI after external CVar changes.

4. Tab Management (5 Nodes)

Register Tab

Register a FoxSettingsTabData DataAsset as a tab. Call in BeginPlay.

Unregister Tab

Remove a previously registered tab from the menu.

Get Registered Tabs

Returns all registered tabs sorted by TabSortOrder. Pure.

Get Tab By Index

Get specific tab. Index 0 = first tab (lowest SortOrder).

Get Num Tabs

Number of registered tabs. Pure.

5. Widget Creation (4 Nodes)

Get Default Widget Class Overrides

Returns struct with default WBP_ Blueprint classes for each widget type.

Create Widgets For Tab

Creates all setting widgets for a tab. Returns sorted array.

Create Keybind Widgets

Creates keybind rows from auto-scanned input mappings.

Wrap Widget In Zebra Row

Wraps widget in alternating-color row container for consistent appearance.

6. Hardware Detection (8 Nodes)

Detect Hardware

Returns FFoxHardwareInfo struct with full system specs.

Recommend Preset

Auto-recommend quality preset based on detected hardware.

Recommend Preset From Info

Same as Recommend Preset but from a provided FFoxHardwareInfo struct.

Get Available Resolutions

All supported resolutions the monitor reports.

Get Desktop Resolution

Native monitor resolution.

Is DLSS Available

True if DLSS plugin is loaded and GPU supports it. Pure.

Is FSR Available

True if FSR plugin is loaded. Pure.

Get GPU Vendor

Identifies GPU manufacturer. Pure.

7. CVar Binding (14 Nodes + 1 Delegate)

■ Access via: Get Fox Settings → Get Binding (SettingID) or Get All Bindings.

Get Current Value

Current CVar value as string.

Get Current Value As Float

Typed getter for float CVars.

Get Current Value As Int

Typed getter for integer CVars.

Get Current Value As Bool

Typed getter for boolean CVars.

Set Value

Set CVar value as string. Immediate but not saved.

Set Value Float / Int / Bool

Typed setters. Same as Set Value but type-safe.

Is CVar Valid

True if CVar exists in engine. Pure.

Should Be Visible

Checks RequiredCVar + RequiredCVarValue condition. Pure.

Revert

Restore to last saved/applied value.

Reset To Default

Reset to DefaultValue from DataAsset entry.

Has Pending Change

True if value differs from last saved state. Pure.

Get CVar Name

Returns the CVar name string this binding is attached to. Pure.

Get Setting ID

Returns the SettingID (FName) from the DataAsset entry. Pure.

OnValueChanged (Delegate)

Fires whenever this binding's value changes. Params: SettingID, NewValue.

8. Keybind Nodes (14 Nodes + 2 Delegates)

■ Access via: Get Fox Settings → Get Input Settings.

Scan Input Mappings

Auto-scan all input actions from engine. Called automatically on init.

Get All Keybindings

All auto-scanned bindings with current key assignments.

Get All Categories

Unique category names from override DataAsset.

Get Keybindings By Category

Filtered by category name.

Start Listening For Key

Starts "press any key" mode for specific action + slot.

Cancel Listening

Cancel the current listening mode. Restores previous key.

Is Listening

True if waiting for player key press. Pure.

Process Detected Key

Process a key press during listening mode. Returns EFoxRebindResult (Success, Conflict, Swapped, Cancelled, InvalidKey, WrongInputDevice).

Rebind Key

Directly assign a key without listening flow.

Is Keyboard Key Bound

Check if a keyboard key is already bound. Returns true + conflicting action name. Use for custom conflict UI.

Is Gamepad Button Bound

Check if a gamepad button is already bound. Returns true + conflicting action name.

Reset All Keybindings

Restore every action to engine defaults.

Apply Keybindings

Write pending changes to engine InputSettings.

Has Pending Keybind Changes

True if any keybinding was changed but not yet applied. Pure.

OnStartedListening (Delegate)

Fires when listening mode begins. Params: ActionName, Slot.

OnKeyDetected (Delegate)

Fires when a key press is detected during listening. Params: ActionName, DetectedKey, Slot.

9. Keybind Row Widget (7 Nodes)

Initialize From Keybind Entry

Initialize widget from FFoxKeybindEntry data.

Start Rebind Slot

Start rebinding: Primary, Secondary, or Gamepad.

Cancel Rebind

Cancel active rebind. Restores old key text.

Reset To Default

Reset this action to engine default keys.

Get Slot Display Text

Human-readable key name for a slot. Pure.

Get Action Display Name / Category

Returns display name and category. Pure.

Refresh Slot Displays

BlueprintNativeEvent. Override in WBP for custom visuals.

10. Gamepad Navigation (12 Nodes + 1 Delegate)

Set Focusable Widgets

Set the ordered list of widgets for D-Pad navigation.

Add Focusable Widget

Append single widget to focusable list.

Clear Focusable Widgets

Remove all. Call before rebuilding tab content.

Navigate Up / Down

Move focus to previous / next widget in list.

Navigate Left / Right

Decrease / increase current value (slider step, dropdown option).

Confirm

A button. Toggles checkbox, opens dropdown, confirms selection.

Cancel

B button. Fires OnCancelPressed delegate. Wire to close menu.

Set Focus Index / Get Focus Index

Direct focus control.

Get Focused Widget

Returns the currently focused widget. Pure.

Notify Mouse Input

Switch to mouse mode — hide focus highlight.

Notify Gamepad Input

Switch to gamepad mode — show focus highlight.

Get Current Input Mode / Is Gamepad Mode

Query input mode. Pure.

OnCancelPressed (Delegate)

Fires when B button / Cancel is pressed. Wire to close menu or show confirm dialog.

11. Delegates (6 Nodes)

OnSettingsApplied

Fires when Apply All Fox Settings is called. No parameters.

OnCustomSettingChanged

Fires for bUseCustomDelegate settings. Params: SettingID, Value.

OnButtonPressed

Fires when Button widget is clicked. Param: SettingID.

OnRebindComplete

Key rebind succeeded. Params: ActionName, NewKey, Slot.

OnRebindConflict

Key already bound elsewhere. Params: ActionName, ConflictingAction, Key.

OnInputModeChanged

Mouse ↔ Gamepad transition detected. Param: NewMode.

12. Style & Data (3 Nodes)

Set Style Data

Apply a FoxSettingsStyleData DataAsset. Colors, fonts, layout.

Get Style Data

Get currently active style DataAsset. Pure.

Set Keybind Override Data

Apply FoxKeybindOverrideData. Display names, categories, visibility.

Complete Blueprint Examples

Example 1: Basic Settings Menu (5 Nodes)

Goal: Settings menu with ESC toggle from PlayerController.

BeginPlay

Get Fox Settings → Register Tab (DA_Graphics) → Register Tab (DA_Audio)

InputAction ESC

Toggle Fox Settings Menu → returns true (opened) or false (closed)

In-Menu Buttons

Apply Button → Apply All Fox Settings

Cancel Button → Revert All Fox Settings → Hide Fox Settings Menu

Reset Button → Reset All Fox Settings To Defaults

■ That's it. 5 nodes for a complete settings menu.

Example 2: Controls Tab with Keybindings

Goal: Mouse Sensitivity + Invert Y (regular settings) followed by all keybind rows.

LoadTab (when player clicks a tab)

Step 1: Create regular setting widgets

Get Fox Settings → Get Bindings Map → store 'Bindings'
Get Default Widget Class Overrides → store 'Overrides'
Create Widgets For Tab → For Each → Wrap Widget In Zebra Row → Add Child to ScrollBox

Step 2: If Controls tab — add keybind rows after regular settings

Branch (Is Controls Tab?) → Create Keybind Widgets → For Each → Wrap Widget In Zebra Row → Add to ScrollBox

■ Result: Mouse Sensitivity slider + Invert Y toggle at top, then all keybind rows below.

Example 3: Auto-Detect Hardware on First Launch

Goal: Detect GPU, auto-select quality preset on first game launch.

BeginPlay

Get Fox Settings → Register Tab (DA_Graphics) → Get Fox Setting ("ShadowQuality") → Branch (Return Value == ""?)

True: First Launch (no saved settings)

Recommend Preset → Switch on EFoxGraphicsPreset:

Ultra → Set Fox Setting ("ShadowQuality", "3")
High → Set Fox Setting ("ShadowQuality", "2")
Medium → Set Fox Setting ("ShadowQuality", "1")
Low → Set Fox Setting ("ShadowQuality", "0")
→ Apply All Fox Settings ← saves auto-detected preset

False: Returning Player

(nothing needed — FoxSettings loads saved values from INI on init)

Multiplayer

Settings are per-client, stored locally. Safe on Dedicated Servers (no UI code runs server-side). No special configuration needed.

FAQ

Q: My setting doesn't appear in the menu.

A: Check that SettingID is not empty, and that RequiredCVar exists (if set).

Q: Can I add settings without CVars?

A: Yes. Set bUseCustomDelegate=true, leave CVarName empty. Bind to OnCustomSettingChanged.

Q: Settings don't persist after restart.

A: Make sure Apply All Fox Settings is called when the player clicks Apply.

Q: Can I use this with UMG?

A: Yes. All widget classes are UUserWidget subclasses designed for UMG.

Q: How do I register custom CVars?

A: Use TAutoConsoleVariable<> in C++ or add to DefaultEngine.ini. Reference by name in DataAsset.

Q: How do Audio volume sliders work?

A: Audio sliders use bUseCustomDelegate with 0-100 integer range. In your OnCustomSettingChanged handler, divide by 100 for 0.0-1.0 and apply via SoundMix.

FoxToolkit Ecosystem

FoxSettings is part of the FoxToolkit family. All plugins work standalone. Install FoxCore (free) to enable cross-plugin communication.

Support

Questions or bugs? Contact Alchemy Fox Studio.

Documentation: alchemy-fox.de/settings

Website: alchemy-fox.de
Email: support@alchemy-fox.de