FoxData

JSON, CSV, DataTable & Runtime Tables for Blueprints
User Guide

Version 1.0
Alchemy Fox Studio

Supported Engine: UE 5.2·5.4, 5.6·5.7

Table of Contents

Installation

Option A: Via Unreal Engine Tab

1. Open the Epic Games Launcher and go to the Unreal Engine tab.
2. Search for FoxData.
3. Click **Install to Engine**, select your Engine version, and click **Install**.
4. Open your project, go to **Edit > Plugins**, enable FoxData, then restart the editor.

Option B: Via Fab Library

1. Open the Epic Games Launcher and go to **Fab**.
2. Open **My Library**, search for FoxData and click on it.
3. Click **Install Plugin**, select your Engine version, and click **Install**.
4. Open your project, go to **Edit > Plugins**, enable FoxData, then restart the editor.

Option C: Manual Installation

1. Copy the FoxData/ folder into **YourProject/Plugins/**.
2. Right-click the .uproject file and select **Generate Visual Studio project files**.
3. Open the project, go to **Edit > Plugins**, enable FoxData, then restart the editor.

All nodes appear under **FoxData | <Category>** when you right-click in any Blueprint graph.

Supported Engine Version: **UE 5.2·5.4, 5.6·5.7**

Quick Start Tutorial

Every FoxData node is a static Blueprint function — no setup, no initialization, no subsystem required (unless noted). Below are step-by-step examples showing how to use FoxData in real project scenarios.

Example 1: Save and Load Player Config via JSON

Goal: Persist player settings (volume, keybinds, difficulty) in a JSON file between sessions.

Build the JSON on Save

Create a Create JSON Object node. Call Set JSON Field three times: Key="Volume" Value=0.8, Key="Difficulty" Value="Hard", Key="MouseSensitivity" Value=1.2.

Write to Disk

Call Save JSON to File with FilePath="Saved/Config/PlayerPrefs.json" and bPrettyPrint=true. Check bSuccess output.

Load on Game Start

In BeginPlay, call Load JSON from File with the same path. Feed the Handle into Get JSON Field Number / Get JSON Field String to restore each setting.

Play and Test

Change a setting and quit. Relaunch — settings persist. No SaveGame object required. The JSON file is human-readable in Saved/Config/.

Example 2: Parse a CSV Loot Table

Goal: Load a loot table from a CSV string and pick a random row for a chest drop.

Prepare Your CSV

Your CSV has headers: ItemName,Rarity,DropChance. Load it as an FString (e.g. from a file or embedded constant).

Parse with Headers

Call Parse CSV with Headers. You get a Headers array and a Rows array. RowCount tells you how many items are in the table.

Pick a Random Row

Use a Random Integer (0 to RowCount-1) to index into Rows. Call Get CSV Column with ColumnIndex=0 to extract all ItemNames, then index directly.

Play and Test

Open a chest — a random item name from your CSV is selected and printed. To add new items, just edit the CSV string. No DataTable asset recompile needed.

Example 3: Runtime Inventory Table

Goal: Manage a player's inventory at runtime without a UDataTable — add, remove, and filter rows dynamically.

Create the Table

In BeginPlay, call Create Runtime Table. Store the UFoxRuntimeTable* reference in a variable "InventoryTable".

Add Items

When the player picks up an item, call Add Row with RowName=ItemID and RowData as your FInventoryItem USTRUCT. bSuccess confirms the add.

Filter by Rarity

To display only "Rare" items in the UI, call Filter Rows with FieldName="Rarity", CompareValue="Rare", Comparison=Equal. Returns matching row names instantly.

Remove on Use

When an item is consumed, call Remove Row with its RowName. Call Get Row Count to update your HUD item counter.

Node Reference

Complete reference for all 53 FoxData nodes organized by category. Each node includes a description, inputs, outputs, and a use case hint.

1. JSON (17 Nodes)

1.1 Parse Json String

Provides 1.1 Parse Json String functionality.

Input	Type	Description
JsonString	String	

Output	Type	Description
Return Value	UFoxJsonObject*	
bSuccess	Bool	Boolean flag

Use Case: JSON helper — read or build JSON data.

1.2 Get Json Field String

Provides 1.2 Get Json Field String functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name

Output	Type	Description
Return Value	String	
bFound	Bool	Boolean flag

Use Case: JSON helper — read or build JSON data.

1.3 Get Json Field Number

Provides 1.3 Get Json Field Number functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name

Output	Type	Description
Return Value	Float	
bFound	Bool	Boolean flag

Use Case: JSON helper — read or build JSON data.

1.4 Get Json Field Bool

Provides 1.4 Get Json Field Bool functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name

Output	Type	Description
Return Value	Bool	
bFound	Bool	Boolean flag

Use Case: JSON helper — read or build JSON data.

1.5 Get Json Field Object

Provides 1.5 Get Json Field Object functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name

Output	Type	Description
Return Value	UFoxJsonObject*	
bFound	Bool	Boolean flag

Use Case: JSON helper — read or build JSON data.

1.6 Get Json Field Array

Provides 1.6 Get Json Field Array functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name

Output	Type	Description
Return Value	Array of UFoxJsonObject*	
bFound	Bool	Boolean flag

Use Case: JSON helper — read or build JSON data.

1.7 Json To String

Provides 1.7 Json To String functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
bPrettyPrint	Bool	Boolean flag

Output	Type	Description
Return Value	String	

Use Case: JSON helper — read or build JSON data.

1.8 Create Json Object

Provides 1.8 Create Json Object functionality.

(none)

Output	Type	Description
Return Value	UFoxJsonObject*	

Use Case: JSON helper — read or build JSON data.

1.9 Set Json Field String

Provides 1.9 Set Json Field String functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name
Value	String	The computed value

(none)

Use Case: JSON helper — read or build JSON data.

1.10 Set Json Field Number

Provides 1.10 Set Json Field Number functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name
Value	Float	The computed value

(none)

Use Case: JSON helper — read or build JSON data.

1.11 Set Json Field Bool

Provides 1.11 Set Json Field Bool functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name
Value	Bool	The computed value

(none)

Use Case: JSON helper — read or build JSON data.

1.12 Set Json Field Object

Provides 1.12 Set Json Field Object functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name
Value	UFoxJsonObject*	The computed value

(none)

Use Case: JSON helper — read or build JSON data.

1.13 Add To Json Array

Provides 1.13 Add To Json Array functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
ArrayKey	String	Unique key name
Element	UFoxJsonObject*	

(none)

Use Case: JSON helper — read or build JSON data.

1.14 Save Json To File

Provides 1.14 Save Json To File functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
FilePath	String	
bPrettyPrint	Bool	Boolean flag

Output	Type	Description
Return Value	Bool	

Use Case: JSON helper — read or build JSON data.

1.15 Load Json From File

Provides 1.15 Load Json From File functionality.

Input	Type	Description
FilePath	String	

Output	Type	Description
Return Value	UFoxJsonObject*	
bSuccess	Bool	Boolean flag

Use Case: JSON helper — read or build JSON data.

1.16 Json Array Length

Provides 1.16 Json Array Length functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
ArrayKey	String	Unique key name

Output	Type	Description
Return Value	Int	

Use Case: JSON helper — read or build JSON data.

1.17 Json Has Field
Provides 1.17 Json Has Field functionality.

Input	Type	Description
Handle	UFoxJsonObject*	
Key	String	Unique key name

Output	Type	Description
Return Value	Bool	

Use Case: JSON helper — read or build JSON data.

2. Serialization (4 Nodes)

2.1 Struct To Json String
Provides 2.1 Struct To Json String functionality.

Input	Type	Description
Struct	Int	

Output	Type	Description
Return Value	String	
bSuccess	Bool	Boolean flag

Use Case: Serialization helper — struct ↔ JSON or JSON-to-JSON merge.

2.2 Json String To Struct
Provides 2.2 Json String To Struct functionality.

Input	Type	Description
JsonString	String	

Output	Type	Description
Return Value	Bool	
Struct	Int	

Use Case: Serialization helper — struct ↔ JSON or JSON-to-JSON merge.

2.3 Merge Json Objects

Provides 2.3 Merge Json Objects functionality.

Input	Type	Description
Base	UFoxJsonObject*	
Override	UFoxJsonObject*	

Output	Type	Description
Return Value	UFoxJsonObject*	

Use Case: Serialization helper — struct ↔ JSON or JSON-to-JSON merge.

2.4 Get Json Keys

Provides 2.4 Get Json Keys functionality.

Input	Type	Description
Handle	UFoxJsonObject*	

Output	Type	Description
Return Value	Array of String	

Use Case: Serialization helper — struct ↔ JSON or JSON-to-JSON merge.

3. CSV (4 Nodes)

3.1 Parse Csv String

Provides 3.1 Parse Csv String functionality.

Input	Type	Description
CsvString	String	
Delimiter	String	

Output	Type	Description
Return Value	Int	
Rows	Array of String	

Use Case: CSV helper — parse, extract or write CSV strings.

3.2 Parse Csv With Headers

Provides 3.2 Parse Csv With Headers functionality.

Input	Type	Description
CsvString	String	
Delimiter	String	

Output	Type	Description
Return Value	Int	
Headers	Array of String	
Rows	Array of String	

Use Case: CSV helper — parse, extract or write CSV strings.

3.3 Get Csv Column

Provides 3.3 Get Csv Column functionality.

Input	Type	Description
Rows	Array of String	
ColumnIndex	Int	Zero-based index

Output	Type	Description
Return Value	Array of String	

Use Case: CSV helper — parse, extract or write CSV strings.

3.4 Save Array As Csv

Provides 3.4 Save Array As Csv functionality.

Input	Type	Description
Headers	Array of String	
Rows	Array of String	
FilePath	String	

Output	Type	Description
Return Value	Bool	

Use Case: CSV helper — parse, extract or write CSV strings.

4. DataTable (5 Nodes)

4.1 Get Data Table Row Count

Provides 4.1 Get Data Table Row Count functionality.

Input	Type	Description
DataTable	UDataTable*	

Output	Type	Description
Return Value	Int	

Use Case: DataTable helper — query rows by index, field or random pick.

4.2 Get Data Table As Map

Provides 4.2 Get Data Table As Map functionality.

Input	Type	Description
DataTable	UDataTable*	

Output	Type	Description
RowNames	Array of Name	
JsonRows	Array of String	

Use Case: DataTable helper — query rows by index, field or random pick.

4.3 Get Random Data Table Row

Provides 4.3 Get Random Data Table Row functionality.

Input	Type	Description
DataTable	UDataTable*	

Output	Type	Description
Return Value	Name	
RowJson	String	
bSuccess	Bool	Boolean flag

Use Case: DataTable helper — query rows by index, field or random pick.

4.4 Filter Data Table Rows

Provides 4.4 Filter Data Table Rows functionality.

Input	Type	Description
DataTable	UDataTable*	
FieldName	Name	
CompareValue	String	
Comparison	EFoxDataComparison	

Output	Type	Description
Return Value	Array of Name	

Use Case: DataTable helper — query rows by index, field or random pick.

4.5 Find Data Table Row By Field

Provides 4.5 Find Data Table Row By Field functionality.

Input	Type	Description
DataTable	UDataTable*	
FieldName	Name	
SearchValue	String	

Output	Type	Description
Return Value	Name	
bFound	Bool	Boolean flag

Use Case: DataTable helper — query rows by index, field or random pick.

5. Config (5 Nodes)

5.1 Load Config File

Provides 5.1 Load Config File functionality.

Input	Type	Description
FilePath	String	

Output	Type	Description
Return Value	Bool	
Config	Map	

Use Case: Config persistence — read/write key-value pairs to disk.

5.2 Save Config File

Provides 5.2 Save Config File functionality.

Input	Type	Description
FilePath	String	
Config	Map	

Output	Type	Description
Return Value	Bool	

Use Case: Config persistence — read/write key-value pairs to disk.

5.3 Get Config Value

Provides 5.3 Get Config Value functionality.

Input	Type	Description
FilePath	String	
Section	String	
Key	String	Unique key name

Output	Type	Description
Return Value	String	
bFound	Bool	Boolean flag

Use Case: Config persistence — read/write key-value pairs to disk.

5.4 Set Config Value

Provides 5.4 Set Config Value functionality.

Input	Type	Description
FilePath	String	
Section	String	
Key	String	Unique key name
Value	String	The computed value

Output	Type	Description
Return Value	Bool	

Use Case: Config persistence — read/write key-value pairs to disk.

5.5 Has Config Key
Provides 5.5 Has Config Key functionality.

Input	Type	Description
FilePath	String	
Section	String	
Key	String	Unique key name

Output	Type	Description
Return Value	Bool	

Use Case: Config persistence — read/write key-value pairs to disk.

6. RuntimeTable (15 Nodes)

6.1 Create Runtime Table
Provides 6.1 Create Runtime Table functionality.

(none)

Output	Type	Description
Return Value	UFoxRuntimeTable*	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.2 Create Runtime Table From Data Table
Provides 6.2 Create Runtime Table From Data Table functionality.

Input	Type	Description
Source	UDataTable*	

Output	Type	Description
Return Value	UFoxRuntimeTable*	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.3 Create Runtime Table From Json
Provides 6.3 Create Runtime Table From Json functionality.

Input	Type	Description
JsonString	String	

Output	Type	Description
Return Value	UFoxRuntimeTable*	
bSuccess	Bool	Boolean flag

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.4 Create Runtime Table From Csv

Provides 6.4 Create Runtime Table From Csv functionality.

Input	Type	Description
CsvString	String	
Delimiter	String	

Output	Type	Description
Return Value	UFoxRuntimeTable*	
bSuccess	Bool	Boolean flag

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.5 Add Runtime Row

Provides 6.5 Add Runtime Row functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	
RowName	Name	
RowData	Int	

Output	Type	Description
Return Value	Bool	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.6 Remove Runtime Row

Provides 6.6 Remove Runtime Row functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	
RowName	Name	

Output	Type	Description
Return Value	Bool	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.7 Update Runtime Row

Provides 6.7 Update Runtime Row functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	
RowName	Name	
RowData	Int	

Output	Type	Description
Return Value	Bool	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.8 Get Runtime Row

Provides 6.8 Get Runtime Row functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	
RowName	Name	

Output	Type	Description
Return Value	Bool	
RowData	Int	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.9 Has Runtime Row

Provides 6.9 Has Runtime Row functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	
RowName	Name	

Output	Type	Description
Return Value	Bool	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.10 Get Runtime Table Row Names

Provides 6.10 Get Runtime Table Row Names functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	

Output	Type	Description
Return Value	Array of Name	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.11 Get Runtime Table Row Count

Provides 6.11 Get Runtime Table Row Count functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	

Output	Type	Description
Return Value	Int	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.12 Filter Runtime Rows

Provides 6.12 Filter Runtime Rows functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	
FieldName	String	Identifier name
CompareValue	String	
Comparison	EFoxDataComparison	

Output	Type	Description
Return Value	Array of Name	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.13 Sort Runtime Rows

Provides 6.13 Sort Runtime Rows functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	
FieldName	String	Identifier name
bAscending	Bool	Boolean flag

Output	Type	Description
Return Value	Array of Name	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.14 Export Runtime Table To Json

Provides 6.14 Export Runtime Table To Json functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	
bPrettyPrint	Bool	Boolean flag

Output	Type	Description
Return Value	String	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

6.15 Export Runtime Table To Csv

Provides 6.15 Export Runtime Table To Csv functionality.

Input	Type	Description
Table	UFoxRuntimeTable*	

Output	Type	Description
Return Value	String	

Use Case: Runtime table — DataTable-like rows you can edit at runtime.

7. String (3 Nodes)

7.1 String To Map

Provides 7.1 String To Map functionality.

Input	Type	Description
Input	String	
PairDelimiter	String	
KeyValueDelimiter	String	Unique key name

Output	Type	Description
Return Value	Map	

Use Case: String/URL utility — encode, decode or map conversion.

7.2 Map To String

Provides 7.2 Map To String functionality.

Input	Type	Description
Map	Map	
PairDelimiter	String	
KeyValueDelimiter	String	Unique key name

Output	Type	Description
Return Value	String	

Use Case: String/URL utility — encode, decode or map conversion.

7.3 Encode Decode Url

Provides 7.3 Encode Decode Url functionality.

Input	Type	Description
Input	String	
bEncode	Bool	Boolean flag

Output	Type	Description
Return Value	String	

Use Case: String/URL utility — encode, decode or map conversion.

Implementation Notes

- UFoxJsonObject wraps TSharedPtr with UCLASS so it's BP-referenceable + GC'd.
- Wildcard struct pins require CustomThunk UFUNCTION with manual FFrame stack handling.
- Runtime Table uses JSON as internal format for type-agnostic storage. Struct conversion via FJsonObjectConverter
- on Get/Set.
- CSV parsing handles quoted fields, escaped commas, and multiline values.
- Config file parsing uses GConfig API where possible, falls back to manual parsing for arbitrary files.
- All file operations use FFileHelper with platform-safe paths (FPaths::ProjectSavedDir() default).