

BLInformatique — RealFPS_SaveSystem – User Guide

Unity 6000.6.0f1

RealFPS

 SaveSystem

SAVE. LOAD. CONTINUE YOUR STORY.



Save Game
2026/09/15 14:32



Save Game
2026/09/14 20:17



Save Game
2026/09/12 11:08



 Player
 Inventory
 Weapons
 Stats
 Missions
 World State
 Environment



 SAVE & LOAD
GAME STATES

 EASY SETUP
& INTEGRATION

 MODULAR
& EXTENSIBLE

 FULL SOURCE
CODE

Table of Contents

Optional Sections (10)

-  Overview
-  Installation and first setup
-  Your first save and load
-  Save slots and the Save UI
-  What the system saves
-  Saveable components and persistent IDs
-  Optional integrations
-  Loading and scene changes
-  Death Recovery
-  Troubleshooting

Components (20)

-  Components Overview
-  BLInformatique.RealFPS (1)
 - SaveablePersistentObjectState
-  BLInformatique.RealFPS.SaveSystem (19)
 - RealFPSSaveableEntity
 - RealFPSSaveManager
 - RealFPSSaveSlotButtonUI
 - RealFPSSaveUIFreezeBridge
 - RealFPSSaveUIManager
 - SaveableComputerUIState
 - SaveableDestructibleObject
 - SaveableDoor
 - SaveableEnemy
 - SaveableEnviro3State
 - SaveableExplosionEntity
 - SaveableGeoDayNightCycleProState
 - SaveableMissionManager
 - SaveableMissionObjectiveNotifier
 - SaveableMissionTriggerZone
 - SaveableMovingPlatform
 - SaveablePickup
 - SaveableToggleEvent
 - SaveableTransform

Optional Sections

Overview

RealFPS Save System is an add-on for Real FPS. It records player progress and the state of supported objects, then restores them from a named save slot. The main `RealFPSSaveManager` handles the player, inventory and equipment as well as registered world saveables. `RealFPSSaveUIManager` supplies an optional player-facing interface for selecting, saving, loading and deleting slots.

You can add saveable components to doors, pickups, enemies, destructible objects and transforms as needed. Bridges support optional systems such as missions, moving platforms, Enviro 3 and GeoDayNightCyclePro. Install and configure each optional system only when your project uses it.

The component reference at the end of this guide reproduces the inspector fields and English tooltips. Use the following chapters to set up a working scene and understand the behavior behind those fields.

Installation and first setup

Requirements

- A Real FPS project using Unity **6000.6.0f1**.
- The RealFPS Save System add-on and its included resources.
- The Real FPS components referenced by the save manager, including the player controller and the inventory or weapon systems you intend to save.

Create the manager

1. Import the add-on into the Real FPS project and let Unity finish compiling.
2. Open the gameplay scene and choose **Tools** → **BLInformatique** → **Real FPS** → **Addons** →  **Save System** → **Create** →  **Save Load Manager**. This menu creates the included `SaveLoadManager` resource prefab.
3. Select the created object. In `RealFPSSaveManager`, assign **Player Root**, **Player Health**, **Player Food**, **Player Stamina**, **Inventory Manager**, **Weapon Inventory Manager** and **Weapon Manager** where those systems are used. **Auto Find Player References** can resolve missing player references on `Awake`, but explicit scene references make setup easier to inspect.
4. Keep **Default Slot Id** as `slot_01` for a first test, or set a stable name of your choice.
5. Check **Known Items**. Add `ItemData` assets that might need to be restored, especially assets absent from the loaded scene. **Auto Build Item Registry** also gathers loaded items from assets, scene pickups and the current inventory.
6. Save the scene.

Optional integrations: The menu also offers **Create** →  **Save System Integrations**. Add it when you need the provided integration prefab; configure only the bridges applicable to your installed add-ons.

Verify one saveable object

Select a scene object whose state should survive a load. Add the relevant `Saveable...` component and its required `RealFPSSaveableEntity`. Ensure that each entity has a **unique persistent ID**. A duplicate ID can make two objects compete for the same saved entry. Start with a simple door, pickup or transform and expand after the first save/load test succeeds.

Your first save and load

1. Enter Play mode in a test scene containing the player and `RealFPSSaveManager`.
2. Move the player and change one observable state, such as picking up an item or moving a saveable object.
3. Call `QuickSaveDefaultSlot()` on the manager (for example from a temporary UI Button `On Click` event). The method uses **Default Slot Id**. Alternatively call `SaveToSlot(string slotId)` from your own code.

4. Change the player or object state again.

5. Call `QuickLoadDefaultSlot()` or `LoadFromSlot(string slotId)` using the same slot ID. Check that the player and configured saveables return to the saved state.

6. Repeat the test in a built player as well as the Editor before release.

`SaveToSlot` and `LoadFromSlot` return `bool`, which your own UI can use to report success or failure. `HasSlot(string slotId)` checks whether a slot exists; `DeleteSlot(string slotId)` removes it. Avoid binding the delete method to an unguarded gameplay shortcut.

Save slots and the Save UI

`RealFPSSaveUIManager` is an optional interface around the save manager. Assign its **Save Manager** and **Save Panel** references. It accepts a TextMeshPro or legacy input field for the current slot ID. With **Use Default Slot If Empty** enabled, an empty input resolves to the manager's default slot.

For a visible slot list, assign **Slots List Parent** and a **RealFPSSaveSlotButtonUI** prefab. The interface can refresh the list when opened, sort recent saves first and select the first entry. Connect the optional Save, Load, Delete, Refresh and Close buttons, or call `SaveCurrentSlot()`, `LoadCurrentSlot()`, `DeleteCurrentSlot()` and `RefreshSlotList()` from UI events.

The UI manager exposes `OpenPanel()`, `ClosePanel()` and `TogglePanel()`. Its old Input System toggle key defaults to **F5**; **Use New Input System** changes the input path. Check **Lock Cursor And Disable FPS** and the optional **FPS Controller** reference so gameplay input does not continue behind the open panel.

What the system saves

The save file records a slot ID, UTC save date and scene name. The manager then captures these groups when their source components exist:

Group	Saved information
Player	Position, rotation and crouch state; health, food, stamina and exhaustion.
Inventory	Item ID, name, quantity, stored energy and weapon instance data where applicable.
Weapons	Selected slot, weapon instances, magazine and reserve ammunition, energy, equipped state and light state.
Equipped item	Item identity, kind, equipped weapon slot and energy.
Runtime dropped items	Item, quantity, position, rotation, optional Rigidbody velocity and optional weapon state, when Save Runtime Dropped Items is enabled.
World objects	States returned by the registered saveable components, indexed by persistent save ID and component type.
Optional weather and clock	Enviro 3 state when its bridge is present and enabled; GeoDayNightCyclePro state through its saveable integration.

The manager restores player state, inventories and equipment, runtime dropped items and registered world objects in order, then runs the optional post-restore pass. It attempts to restore equipped device energy after inventory reconstruction. Provide all `ItemData` assets that might be needed after loading through **Known Items** or the automatic registry; an unresolved item cannot be rebuilt correctly.

✚ Saveable components and persistent IDs

The manager discovers components implementing `IRealFPSSaveable` and stores each component's serialized state under its save ID and component type. Add the matching saveable to every world object whose state matters. Examples from the current package include `SaveableDoor`, `SaveablePickup`, `SaveableEnemy`, `SaveableExplosionEntity`, `SaveableDestructibleObject`, `SaveableToggleEvent`, `SaveableComputerUIState`, `SaveableTransform` and `SaveablePersistentObjectState`.

Most saveables require `RealFPSSaveableEntity`, which supplies a unique persistent ID. Generate an ID once for a scene object, then keep it stable across saves and builds. When duplicating a scene object, inspect both IDs and generate a new ID for the copy. `SaveableTransform` can save position, rotation, scale and optional Rigidbody velocity; its inspector controls whether coordinates use local or world space and whether the Rigidbody is temporarily made kinematic during restore.

Practical scene test: Save, change the object's state, load, and confirm the expected state. Then leave Play mode, reopen the scene and check that its persistent ID remains the same.

🔗 Optional integrations

Integration	Component or bridge	Setup check
Enviro 3	<code>SaveableEnviro3State</code>	Assign or auto-find the bridge; leave Save Enviro 3 State enabled on the manager.
GeoDayNightCyclePro	<code>SaveableGeoDayNightCycleProState</code>	Assign the compatible cycle component or enable its automatic lookup; choose which clock values to save.
Mission System	<code>SaveableMissionManager</code> , <code>SaveableMissionObjectiveNotifier</code> , <code>SaveableMissionTriggerZone</code>	Add the relevant saveables to mission objects. The mission manager bridge restores its runtime state after mission initialization and participates in the post-restore pass.
Moving Platform System	<code>SaveableMovingPlatform</code>	Add it to a configured moving platform with its persistent entity.

The optional bridges let the Save System compile and run without forcing every other add-on into every project. For each installed integration, perform a save/load test of a state that visibly changed, including a test after restarting the scene.

🟦 Loading and scene changes

`LoadFromSlot` checks the scene stored in the save. With **Reload Current Scene Before Restore** enabled, even a save from the current scene reloads it first to reset runtime objects, AI and coroutines. A save from another scene loads that scene. **Auto Restore After Scene Load** must remain enabled for the pending data to be applied after such a reload. Keep the target scene available to the build.

The manager can create a full-screen loading UI automatically when **Auto Create Loading UI** is enabled and no **Loading Canvas Group** is assigned. You can instead assign your own CanvasGroup and text. Configure **Loading Message**, **Loading Fade Duration** and **Minimum Loading Display Duration** to suit the project, then verify that the overlay clears after both same-scene and cross-scene loads.

Continue with **Death Recovery** to configure the player death menu and the Load Last Save action.

Death Recovery

`RealFPSDeathRecovery` belongs to **Real FPS Core** and is attached to the player. It requires `FPSHealth`. It can use the Save System through an optional runtime bridge: Core can still run without the Save System add-on.

Set up the death menu

1. Add **RealFPSDeathRecovery** to the player (`BLInformatique/Real FPS/Player/`  `Add Death Recovery`). Keep one instance on that player.
2. Choose **ShowDeathMenu** in **Recovery Mode**. Assign a death panel in `FPSHealth` ; it is shown when the player's death is detected.
3. Connect a **Restart** button to `RealFPSDeathRecovery.RestartCurrentScene()` and a **Load Last Save** button to `RealFPSDeathRecovery.LoadLastSave()` using Unity Button `On Click` events.
4. Ensure a `RealFPSSaveManager` exists in the gameplay scene and **Auto Find Save Manager** is enabled. Create a valid save slot before testing the Load Last Save button.
5. Leave **Override Health Auto Respawn** enabled if this component owns recovery, preventing the legacy `FPSHealth` automatic respawn from competing with it.
6. Test both buttons after player death in Play mode and in a build.

The save-loading button calls the Save Manager's `HasSlot` and `LoadFromSlot` methods. If loading succeeds, the Save Manager restores the player and world as described in **Loading and scene changes**. The two systems each have an optional fade: Death Recovery fades before the recovery action; the Save Manager can display its loading overlay during restoration.

Other recovery modes

Mode	Result
Restart Scene	Automatically reloads the active scene after Recovery Delay .
Respawn Point	Moves the player to the assigned Respawn Point , or to the player's initial position if Use Initial Transform As Fallback is enabled. Restore Full Health On Respawn controls health recovery.
Load Last Save	Automatically selects a save slot and starts loading after Recovery Delay .
Show Death Menu	Waits for a button selection; no automatic recovery action is started.

The public `RespawnAtRecoveryPoint()` method can also be wired to a menu button.

`RecoverUsingConfiguredMode()` executes the configured mode immediately; when that mode is **ShowDeathMenu**, it chooses **LoadLastSave**. The delay applies to automatic modes; button methods start immediately.

How the save slot is selected

Set **Save Slot Id** for a fixed slot. When the field is empty, the script checks, in order: the Save Manager's last saved slot, its last loaded slot, the slot remembered for this play session, then its default slot. A missing slot causes loading to fail; the system does not create a save automatically on death.

With **Restart Scene On Failure** enabled, a failed save load restarts the active scene. Disable it if you want to handle the failure through the **On Save Load Failed** UnityEvent. The component also exposes **On Recovery Requested**, **On Recovery Started** and **On Local Respawn Completed** events.

Fade and validation

With **Auto Create Fade UI** enabled, Death Recovery builds a full-screen CanvasGroup if none is assigned. You can instead assign **Fade Canvas Group** and adjust **Fade Duration**; transitions use unscaled time. Verify the death panel, buttons, fade order and cursor behavior in the project's actual UI. For troubleshooting, inspect the runtime **Resolved Save Slot** and **Last Status** fields and temporarily enable **Verbose Logs**.



Troubleshooting

Symptom	First checks
A player reference is missing	Assign it explicitly on <code>RealFPSSaveManager</code> ; check Auto Find Player References and the active player in the scene.

Symptom	First checks
An inventory item does not return	Register its <code>ItemData</code> in Known Items or confirm that the automatic registry can discover it.
Two world objects restore incorrectly	Check their <code>RealFPSSaveableEntity</code> IDs for duplicates and confirm each has the intended saveable component.
A scene reload does not apply the saved state	Check Auto Restore After Scene Load , the saved scene name and whether the scene can be loaded.
The slot list appears empty	Check Save Manager , Slots List Parent , Slot Button Prefab and Refresh Slot List On Open on the UI manager.
Equipped energy is wrong just after load	Check Restore Equipped Device Energy , the item registry and Equipped Energy Restore Frames .

Turn on **Verbose Logs** temporarily while diagnosing; turn it back off if the Console becomes noisy.

CONTENTS

 Filter components...

-  Table of Contents
-  Overview
-  Installation and first setup
-  Your first save and load
-  Save slots and the Save UI
-  What the system saves
-  Saveable components and persistent IDs
-  Optional integrations
-  Loading and scene changes
-  Death Recovery
-  Troubleshooting
-  Components (20)

BLINFORMATIQUE.REALFPS

SaveablePersistentObjectState

BLINFORMATIQUE.REALFPS.SAVESYSTEM

RealFPSSaveableEntity

RealFPSSaveManager

RealFPSSaveSlotButtonUI

RealFPSSaveUIFreezeBridge

RealFPSSaveUIManager

SaveableComputerUIState

SaveableDestructibleObject

SaveableDoor

SaveableEnemy

SaveableEnviro3State

SaveableExplosionEntity

SaveableGeoDayNightCycleProState

SaveableMissionManager

SaveableMissionObjectiveNotifier

SaveableMissionTriggerZone

SaveableMovingPlatform

SaveablePickup

SaveableToggleEvent

SaveableTransform

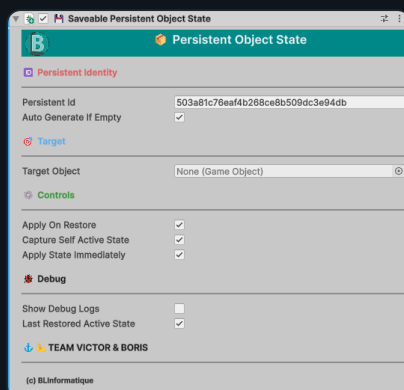
Components

Tooltips are in **English**. Missing tooltips can be completed automatically from field names. Tables list documented fields. Optional notes (Markdown) and images are shown below each component when present.

▼ **SaveablePersistentObjectState** MonoBehaviour

BLInformatique.RealFPS

AddComponentMenu: BLInformatique/Real FPS/Save/📄 Saveable Persistent Object State • Requires: RealFPSSaveableEntity

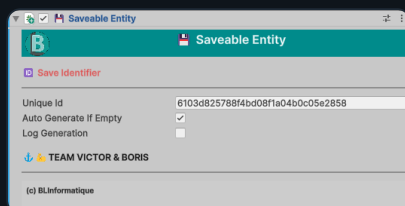


Field	Type	Tooltip	Hints
↓			
<code>persistentId</code>	<code>String</code>	Unique persistent ID used by the save system. Generate once and never duplicate.	—
<code>autoGenerateIfEmpty</code>	<code>Boolean</code>	Automatically generate an ID if the field is empty.	—
↓			
<code>targetObject</code>	<code>GameObject</code>	Target <code>GameObject</code> whose active state is saved. If null, uses this <code>GameObject</code> .	Object Reference
↓			
<code>applyOnRestore</code>	<code>Boolean</code>	Apply saved active state automatically on restore.	—
<code>captureSelfActiveState</code>	<code>Boolean</code>	If true, captures the current active state of the target.	—
<code>applyStateImmediately</code>	<code>Boolean</code>	If true, restores the active state on Awake from the current scene state when possible.	—
↓			
<code>showDebugLogs</code>	<code>Boolean</code>	Shows debug logs in Console.	—
<code>lastRestoredActiveState</code>	<code>Boolean</code>	Last restored active state.	—

▼ RealFPSSaveableEntity MonoBehaviour

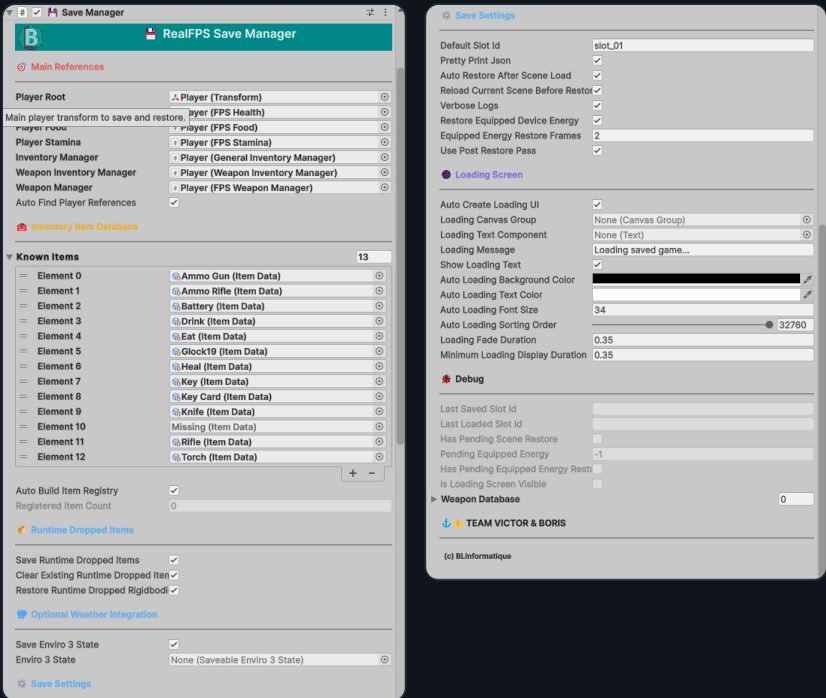
BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/ Saveable Entity



Field	Type	Tooltip	Hints
↓			
<code>uniqueId</code>	<code>String</code>	Unique persistent identifier used by the save system.	Private Serialized
<code>autoGenerateIfEmpty</code>	<code>Boolean</code>	Automatically generate an ID if the field is empty.	—
<code>logGeneration</code>	<code>Boolean</code>	If true, logs ID generation in the Console.	—

AddComponentMenu: BLInformatique/Real FPS Save System/Save Manager



Field	Type	Tooltip	Hints
playerRoot	Transform	Main player transform to save and restore.	Object Reference
playerHealth	FPSHealth	Player health component.	Object Reference
playerFood	FPSFood	Player food component.	Object Reference
playerStamina	FPSStamina	Player stamina component.	Object Reference
inventoryManager	GeneralInventoryManager	Player inventory component.	Object Reference
weaponInventoryManager	WeaponInventoryManager	Player weapon inventory component.	Object Reference
weaponManager	FPSWeaponManager	Player weapon manager component.	Object Reference

Field	Type	Tooltip	Hints
autoFindPlayerReferences	Boolean	Automatically try to find missing player references on Awake.	—
↓			
knownItems	ItemData[]	All ItemData assets that can be restored by the save system.	List / Array
autoBuildItemRegistry	Boolean	Automatically builds a runtime item registry from all loaded ItemData assets, scene pickups and the current inventory.	—
registeredItemCount	Int32	Number of unique ItemData assets currently available to the save system.	—
↓			
saveRuntimeDroppedItems	Boolean	If true, runtime dropped items are captured and respawned on restore.	—
clearExistingRuntimeDroppedItemsBeforeRestore	Boolean	If true, existing runtime dropped items are cleared before respawning saved ones.	—

Field	Type	Tooltip	Hints
restoreRuntimeDroppedRigidbodybodies	Boolean	If true, rigidbody velocity is restored for runtime dropped items when possible.	—
↓			
saveEnviro3State	Boolean	If true, the save system tries to capture and restore Enviro 3 when the optional bridge is present.	—
enviro3State	SaveableEnviro3State	Optional Enviro 3 save bridge. If empty, it is found automatically at runtime.	Object Reference
↓			
defaultSlotId	String	Default slot ID used by quick save and quick load.	—
prettyPrintJson	Boolean	If true, JSON files are written with formatting for easier debugging.	—
autoRestoreAfterSceneLoad	Boolean	If true, scene-changing loads are restored automatically after scene load.	—

Field	Type	Tooltip	Hints
reloadCurrentSceneBeforeRestore	Boolean	Reloads the current scene before applying a save, ensuring all runtime objects, AI states, animations and coroutines start from a clean scene state.	—
verboseLogs	Boolean	If true, logs save and load operations in the Console.	—
restoreEquippedDeviceEnergy	Boolean	If true, tries to restore equipped device energy after inventory restore.	—
equippedEnergyRestoreFrames	Int32	How many frames to wait before applying restored equipped energy.	—
usePostRestorePass	Boolean	If true, calls all post-restore listeners after world object restoration.	—
↓			
autoCreateLoadingUI	Boolean	Automatically creates a persistent full-screen loading UI when no Loading Canvas Group is assigned.	—

Field	Type	Tooltip	Hints
loadingCanvasGroup	CanvasGroup	Optional full-screen CanvasGroup displayed while a saved game is being restored.	Object Reference
loadingTextComponent	Text	Optional UI Text used to display the loading message.	Object Reference
loadingMessage	String	Message displayed while a saved game is being restored.	—
showLoadingText	Boolean	Shows the loading message while the loading screen is visible.	—
autoLoadingBackgroundColor	Color	Background color used by the automatically created loading screen.	—
autoLoadingTextColor	Color	Text color used by the automatically created loading message.	—
autoLoadingFontSize	Int32	Font size used by the automatically created loading message.	—

Field	Type	Tooltip	Hints
autoLoadingSortingOrder	Int32	Sorting order used by the automatically created loading canvas. • Range [0..32767]	Range
loadingFadeDuration	Single	Duration of the loading screen fade-out in unscaled seconds.	—
minimumLoadingDisplayDuration	Single	Minimum time the loading screen remains fully visible, in unscaled seconds.	—
↓			
lastSavedSlotId	String	Last saved slot ID.	—
lastLoadedSlotId	String	Last loaded slot ID.	—
hasPendingSceneRestore	Boolean	True while a scene load restore is pending.	—
pendingEquippedEnergy	Single	Runtime restored energy waiting to be applied to the equipped device.	—
hasPendingEquippedEnergyRestore	Boolean	True if an equipped device energy restore is pending.	—
isLoadingScreenVisible	Boolean	True while the save loading screen is visible.	—

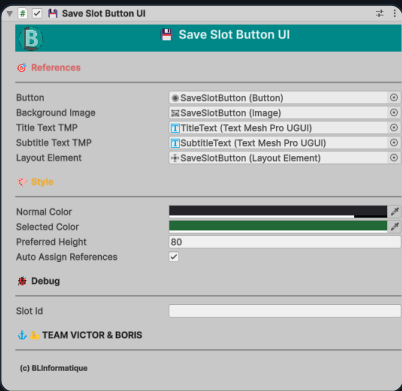
Field	Type	Tooltip	Hints
weaponDatabase	ArmsAndWeaponData[]	Weapon database used to resolve saved weapon names when loading a save.	List / Array

▼

RealFPSSaveSlotButtonUI
MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/UI/  Save Slot Button UI • Requires: RectTransform, Button, Image



Field	Type	Tooltip	Hints
↓			
slotId	String	Current slot ID represented by this UI entry.	Private Serialized
↓			
button	Button	Main clickable button for this save slot entry.	Object Reference
backgroundImage	Image	Background image used to show selected state.	Object Reference
titleTextTMP	TMP_Text	TMP text used to display the slot title.	Object Reference
subtitleTextTMP	TMP_Text	TMP text used to display the slot subtitle.	Object Reference
layoutElement	LayoutElement	Optional layout element used to define a stable row height.	Object Reference
↓			

Field	Type	Tooltip	Hints
normalColor	Color	Background color when the slot is not selected.	—
selectedColor	Color	Background color when the slot is selected.	—
preferredHeight	Single	Preferred row height used by the layout system.	—
autoAssignReferences	Boolean	If true, missing references are auto-assigned from this prefab hierarchy.	—

▼ RealFPSSaveUIFreezeBridge MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/UI/  Save UI Freeze Bridge

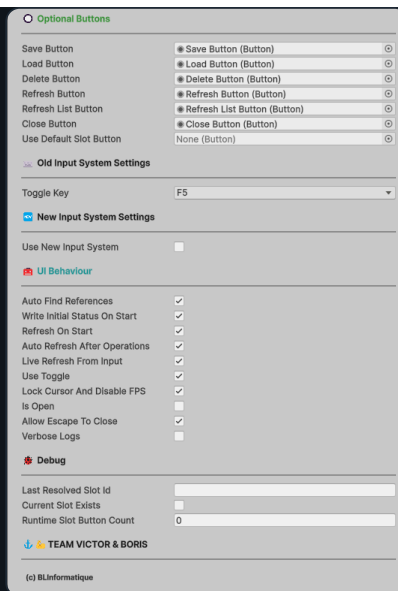
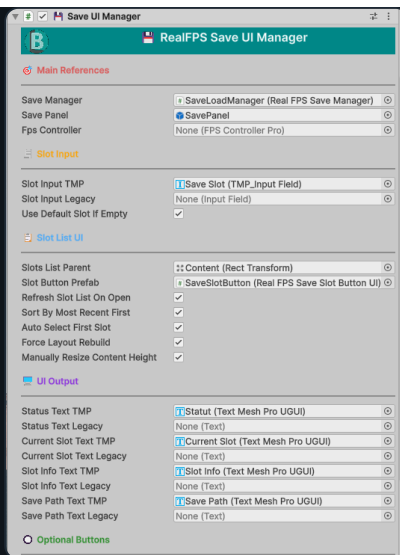


Field	Type	Tooltip	Hints
↓			
freezeOnEnable	Boolean	If true, gameplay is frozen automatically when this GameObject is enabled.	—
releaseOnDisable	Boolean	If true, gameplay is released automatically when this GameObject is disabled.	—
releaseOnDestroy	Boolean	If true, gameplay freeze is also released when this component is destroyed.	—
↓			
verboseLogs	Boolean	If true, logs freeze state changes in the Console.	—

▼ RealFPSSaveUIManager MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/UI/  Save UI Manager



Field	Type	Tooltip	Hints
↓			
saveManager	RealFPSManager	Main save manager used for save, load and delete operations.	Object Reference
savePanel	GameObject	Panel root that contains the whole save/load UI.	Object Reference
fpsController	FPSControllerPro	Optional FPS controller disabled while the save UI is open.	Object Reference
↓			
slotInputTMP	TMP_InputField	TMP input field used to type or edit the current slot ID.	Object Reference
slotInputLegacy	InputField	Legacy input field used to type or edit the current slot ID.	Object Reference
useDefaultSlotIfEmpty	Boolean	If true, uses the default slot from the save manager when the input field is empty.	—
↓			
slotsListParent	Transform	Parent transform where all slot buttons are instantiated.	Object Reference
slotButtonPrefab	RealFPSManagerSlotButtonUI	Prefab used to render one save slot entry in the list.	Object Reference
refreshSlotListOnOpen	Boolean	If true, the slot list is refreshed every time the panel opens.	—

Field	Type	Tooltip	Hints
sortByMostRecentFirst	Boolean	If true, the slot list is sorted by most recent save date first when possible.	—
autoSelectFirstSlot	Boolean	If true, automatically selects the first visible slot after rebuilding the list.	—
forceLayoutRebuild	Boolean	If true, forces a layout rebuild after the slot list is generated.	—
manuallyResizeContentHeight	Boolean	If true, recalculates the Content height manually after rebuilding the slot list.	—
↓			
statusTextTMP	TMP_Text	TMP text used to display status messages.	Object Reference
statusTextLegacy	Text	Legacy text used to display status messages.	Object Reference
currentSlotTextTMP	TMP_Text	TMP text used to display the resolved slot ID.	Object Reference
currentSlotTextLegacy	Text	Legacy text used to display the resolved slot ID.	Object Reference
slotInfoTextTMP	TMP_Text	TMP text used to display save slot information.	Object Reference
slotInfoTextLegacy	Text	Legacy text used to display save slot information.	Object Reference
savePathTextTMP	TMP_Text	TMP text used to display the physical save file path.	Object Reference
savePathTextLegacy	Text	Legacy text used to display the physical save file path.	Object Reference
↓			
saveButton	Button	Optional button used to save the current slot.	Object Reference
loadButton	Button	Optional button used to load the current slot.	Object Reference
deleteButton	Button	Optional button used to delete the current slot.	Object Reference
refreshButton	Button	Optional button used to refresh the selected slot information.	Object Reference

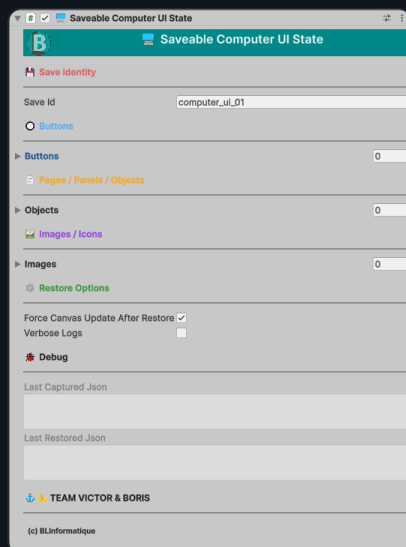
Field	Type	Tooltip	Hints
refreshListButton	Button	Optional button used to refresh the full slot list.	Object Reference
closeButton	Button	Optional button used to close the panel.	Object Reference
useDefaultSlotButton	Button	Optional button used to force the default slot name in the input.	Object Reference
↓			
toggleKey	KeyCode	Key used to open or close the save UI when using the old Input System.	Enum
↓			
useNewInputSystem	Boolean	Enable this if you want to use the new Input System.	—
↓ 🎮 New Input System Actions			
keypressEvent	InputAction	Input action used as the main key press for toggling or opening the save UI.	—
keyDownEvent	InputAction	Input action used when the key is pressed down in hold mode.	—
keyUpEvent	InputAction	Input action used when the key is released in hold mode.	—
↓			
autoFindReferences	Boolean	If true, tries to find missing references automatically on Awake.	—
writeInitialStatusOnStart	Boolean	If true, writes an initial status message on Start.	—
refreshOnStart	Boolean	If true, refreshes slot information automatically on Start.	—
autoRefreshAfterOperations	Boolean	If true, refreshes slot information after save, load and delete operations.	—
liveRefreshFromInput	Boolean	If true, refreshes slot information live when the slot input changes.	—

Field	Type	Tooltip	Hints
useToggle	Boolean	If true, pressing the key toggles between open and close.	—
lockCursorAndDisableFPS	Boolean	If true, unlocks the cursor and disables the FPS controller while the UI is open.	—
isOpen	Boolean	True if the save UI is currently open.	—
allowEscapeToClose	Boolean	If true, pressing Escape closes the save UI when using the old Input System.	—
verboseLogs	Boolean	If true, logs UI operations in the Console.	—
↓			
lastResolvedSlotId	String	Last resolved slot ID used by the UI.	—
currentSlotExists	Boolean	True if the current slot exists on disk.	—
runtimeSlotButtonCount	Int32	Current instantiated slot button count.	—

▼ SaveableComputerUIState MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Saveables/ Saveable Computer UI State



Field	Type	Tooltip	Hints
↓			

Field	Type	Tooltip	Hints
saveld	String	Unique save ID used by the RealFPS Save System. Keep this value stable between saves.	—
↓			
buttons	ButtonStateEntry[]	Buttons whose interactable state will be saved and restored.	List / Array
↓			
objects	ObjectStateEntry[]	GameObjects whose active state will be saved and restored.	List / Array
↓			
images	ImageStateEntry[]	Images whose sprite state will be saved and restored using the configured sprite list.	List / Array
↓			
forceCanvasUpdateAfterRestore	Boolean	If true, Canvas.ForceUpdateCanvases is called after restoring the UI state.	—
verboseLogs	Boolean	If true, logs save and restore operations in the Console.	—
↓			
lastCapturedJson	String	Last captured JSON state. • TextArea	TextArea
lastRestoredJson	String	Last restored JSON state. • TextArea	TextArea

SaveableDestructibleObject MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/ Saveable Destructible Object • Requires: RealFPSSaveableEntity, RealFPSDestructibleObject

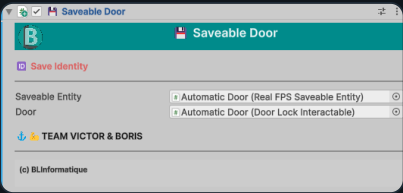


Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSSaveableEntity	Reference to the saveable entity that provides the unique persistent ID.	Object Reference
↓			
destructibleObject	RealFPSDestructibleObject	Destructible object whose state is saved and restored.	Object Reference
↓			
verboseLogs	Boolean	Display save and restore messages in the Console.	—

▼ **SaveableDoor** MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/ Saveable Door • Requires: RealFPSSaveableEntity



Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSSaveableEntity	Reference to the saveable entity that provides the unique persistent ID.	Object Reference
door	DoorLockInteractable	Door component to save and restore.	Object Reference

▼ **SaveableEnemy** MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/ Saveable Enemy • Requires: RealFPSSaveableEntity

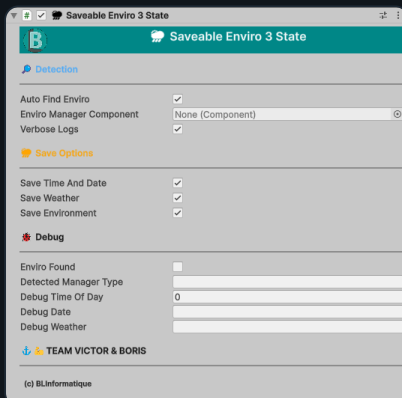


Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSSaveableEntity	Reference to the saveable entity that provides the unique persistent ID.	Object Reference
enemyHealth	EnemyHealth	Enemy health component to save and restore.	Object Reference
↓			
savePosition	Boolean	If true, saves and restores world position.	—
saveRotation	Boolean	If true, saves and restores world rotation.	—

▼ SaveableEnviro3State MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Integrations/Saveable Enviro 3 State



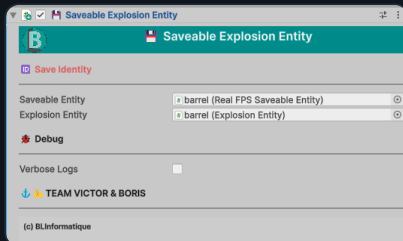
Field	Type	Tooltip	Hints
↓			
autoFindEnviro	Boolean	If true, the script searches Enviro 3 automatically at runtime.	—
enviroManagerComponent	Component	Optional Enviro Manager component reference. Leave empty to use EnviroManager.instance.	Object Reference
verboseLogs	Boolean	If true, logs Enviro 3 detection, capture and restore operations.	—
↓			
saveTimeAndDate	Boolean	If true, saves and restores Enviro time and date.	—
saveWeather	Boolean	If true, saves and restores Enviro active weather.	—

Field	Type	Tooltip	Hints
saveEnvironment	Boolean	If true, saves and restores Enviro environment values like wetness, snow and temperature.	—
↓			
enviroFound	Boolean	True when EnviroManager.instance has been found.	—
detectedManagerType	String	Detected Enviro Manager type name.	—
debugTimeOfDay	Single	Last captured Enviro time of day.	—
debugDate	String	Last captured Enviro date.	—
debugWeather	String	Last captured Enviro weather.	—

SaveableExplosionEntity MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Saveable Explosion Entity • Requires: RealFPSSaveableEntity

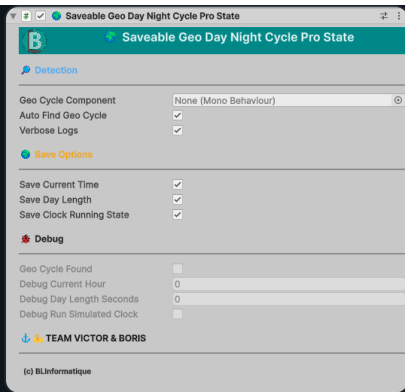


Field	Type	Tooltip	Hints
saveableEntity	RealFPSSaveableEntity	Reference to the saveable entity that provides the unique persistent ID.	Object Reference
explosionEntity	ExplosionEntity	Explosion entity to save and restore.	Object Reference
↓			
verboseLogs	Boolean	If true, logs save and restore fallback operations in the Console.	—

SaveableGeoDayNightCycleProState MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Integrations/Saveable Geo Day Night Cycle Pro State



Field	Type	Tooltip	Hints
↓			
geoCycleComponent	MonoBehaviour	Optional reference to the GeoSunCyclePro component. If empty, it will be found automatically.	Object Reference
autoFindGeoCycle	Boolean	Automatically finds GeoSunCyclePro when the reference is missing.	—
verboseLogs	Boolean	Displays GeoDayNightCyclePro capture and restore logs.	—
↓			
saveCurrentTime	Boolean	Saves and restores the current GeoDayNightCyclePro time.	—
saveDayLength	Boolean	Saves and restores the configured day length.	—
saveClockRunningState	Boolean	Saves and restores whether the simulated clock is running.	—
↓			
geoCycleFound	Boolean	True when a compatible GeoSunCyclePro component is available.	—
debugCurrentHour	Single	Last captured GeoDayNightCyclePro hour.	—
debugDayLengthSeconds	Single	Last captured GeoDayNightCyclePro day length in seconds.	—
debugRunSimulatedClock	Boolean	Last captured simulated clock running state.	—

▼ **SaveableMissionManager** MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Bridge/  Saveable Mission Manager • Requires: RealFPSSaveableEntity



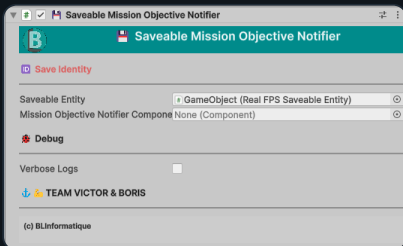
Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSSaveableEntity	Reference to the saveable entity that provides the persistent save ID.	Object Reference
missionManagerComponent	Component	Optional mission manager component resolved at runtime if the Mission System addon is installed.	Object Reference
↓			
verboseLogs	Boolean	If true, logs bridge operations in the Console.	—

SaveableMissionObjectiveNotifier

MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Bridge/Saveable Mission Objective Notifier • Requires: RealFPSSaveableEntity



Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSSaveableEntity	Reference to the saveable entity that provides the persistent save ID.	Object Reference
missionObjectiveNotifierComponent	Component	Optional mission objective notifier component resolved at runtime if the Mission System addon is installed.	Object Reference

Field	Type	Tooltip	Hints
↓			
verboseLogs	Boolean	If true, logs bridge operations in the Console.	—

SaveableMissionTriggerZone

MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Bridge/Saveable Mission Trigger Zone • Requires: RealFPSSaveableEntity



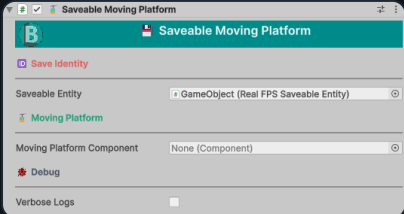
Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSSaveableEntity	Reference to the saveable entity that provides the persistent save ID.	Object Reference
missionTriggerZoneComponent	Component	Optional mission trigger zone component resolved at runtime if the Mission System addon is installed.	Object Reference
↓			
verboseLogs	Boolean	If true, logs bridge operations in the Console.	—

SaveableMovingPlatform

MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Bridge/Saveable Moving Platform • Requires: RealFPSSaveableEntity



Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSSaveableEntity	Saveable entity that provides the unique persistent ID.	Object Reference
↓			
movingPlatformComponent	Component	Optional RealFPSMovingPlatform component resolved without creating a compile dependency.	Object Reference
↓			
verboseLogs	Boolean	Writes capture, restore and integration messages to the Console.	—

▼ **SaveablePickup** MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/  Saveable Pickup • Requires: RealFPSSaveableEntity



Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSSaveableEntity	Reference to the saveable entity that provides the unique persistent ID.	Object Reference
↓			
verboseLogs	Boolean	If true, logs save and restore fallback operations in the Console.	—

▼ **SaveableToggleEvent** MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/  Saveable Toggle Event • Requires: RealFPSSaveableEntity

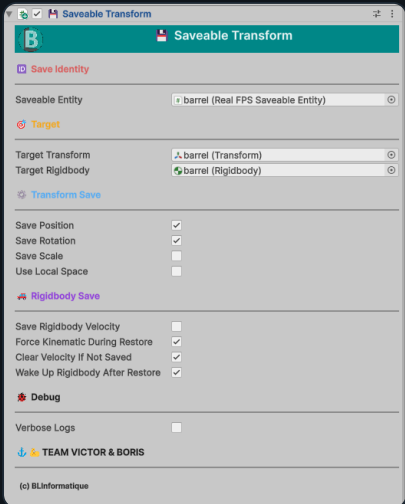


Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSaveableEntity	Reference to the saveable entity that provides the persistent save ID.	Object Reference
toggleEvent	ToggleEventInteractable	Toggle component to save and restore.	Object Reference

SaveableTransform MonoBehaviour

BLInformatique.RealFPS.SaveSystem

AddComponentMenu: BLInformatique/Real FPS Save System/Saveable Transform • Requires: RealFPSaveableEntity



Field	Type	Tooltip	Hints
↓			
saveableEntity	RealFPSaveableEntity	Reference to the saveable entity that provides the unique persistent ID.	Object Reference
↓			
targetTransform	Transform	Transform to save and restore. If empty, this GameObject transform is used.	Object Reference
targetRigidbody	Rigidbody	Optional Rigidbody used to save and restore physics velocity.	Object Reference

Field	Type	Tooltip	Hints
↓			
savePosition	Boolean	If true, saves and restores the transform position.	—
saveRotation	Boolean	If true, saves and restores the transform rotation.	—
saveScale	Boolean	If true, saves and restores the transform local scale.	—
useLocalSpace	Boolean	If true, uses local position and local rotation instead of world space.	—
↓			
saveRigidbodyVelocity	Boolean	If true, saves and restores Rigidbody linear and angular velocity.	—
forceKinematicDuringRestore	Boolean	If true, temporarily sets the Rigidbody to kinematic while restoring transform values.	—
clearVelocityIfNotSaved	Boolean	If true, clears Rigidbody velocities when velocity restore is disabled.	—
wakeUpRigidbodyAfterRestore	Boolean	If true, wakes up the Rigidbody after restore.	—
↓			
verboseLogs	Boolean	If true, logs capture and restore operations in the Console.	—