[English Documentation - Final Version]

Eventus: The Unified Communication Hub

Version: 1.0.0
Created by: David Souza (twoBrackets)

1. Philosophy

Eventus is a professional, centralized communication system designed to create highly decoupled and dynamic architectures in your Unity project. It eliminates the need for direct component references, preventing "spaghetti code" and making your project cleaner, more scalable, and easier to maintain.

Eventus masterfully combines two essential communication patterns into a single, elegant API:

1. **Event Bus (Push System):** For reactive, notification-based communication. Components Publish an event (e.g., OnPlayerCrashed) to announce that something has happened, and any interested component can Subscribe to react to it. This is ideal for actions and one-time occurrences.
2. **Data Hub (Blackboard / Pull System):** For shared, real-time state. Components Write data (e.g., current motorcycle speed) to a central "blackboard," and any other component can Read that data at any time. This is perfect for values that change every frame, like UI data.

The entire system is managed through the intuitive **Eventus Hub** editor window, giving you full visual control over your project's communication channels without ever needing to manually edit the underlying code.

2. Quick Start Guide

Get Eventus running in your project in under 5 minutes.

1. **Add Eventus to Your Scene:**
   - Create an empty GameObject in your main or bootstrap scene and name it Eventus.
   - Attach the Eventus.cs script to it.
   - (Recommended) Save this object as a Prefab for easy use in other scenes.
2. **Set Script Execution Order (Crucial!):**
   - Go to Edit -> Project Settings -> Script Execution Order.
   - Click the + button and select the Eventus script.
   - Set its value to a negative number, like -100. This ensures Eventus is always ready before any other script tries to use it, preventing race conditions.
3. **Open the Eventus Hub:**

- Go to the Unity menu: Tools -> Eventus -> Open Window.
- This window is your central command for all communication channels.

4. **Create Your First Channels:**
   - Navigate to the **"Channels"** tab in the Eventus Hub.
   - In the bottom section, enter a name for your new channel (e.g., PlayerHealth).
   - Select its attributes:
     - **AsData:** For data that will be read/written (Read/Write).
     - **AsEvent:** For notifications that will be published/subscribed (Publish/Subscribe).
     - **DisallowNull:** (Only for AsData) Prevents writing null values to this channel, ensuring data integrity.
   - Click "Add" to add it to the list.
   - Click **"Save Changes"** to generate the code and recompile.
5. **Use It in Your Scripts!**

```
using EventusAsset;
using EventusAsset.Core;
using UnityEngine;

public class Player : MonoBehaviour
{
    private int _health = 100;

    void Start()
    {
        // Write initial health to the Data Hub
        Eventus.Write<int>(Channel.PlayerHealth, _health);

        // Subscribe to a damage event
        Eventus.Subscribe<int>(Channel.OnTakeDamage, TakeDamage);
    }

    private void OnDisable()
    {
        // Always unsubscribe to prevent memory leaks!
        Eventus.Unsubscribe<int>(Channel.OnTakeDamage, TakeDamage);
    }

    private void TakeDamage(int amount)
    {
        _health -= amount;
        // Update the value in the Data Hub for other systems to read
        Eventus.Write<int>(Channel.PlayerHealth, _health);
    }
}
```

3. The Eventus Hub Editor Window

Home Tab

A welcome screen with quick links to the official documentation in English and Portuguese.

Channels Tab

This is the core of the system management.

- **Channel List:** Displays all currently defined channels in alphabetical order.
- **Attributes:** Each channel shows its assigned attributes ([Event], [Data], [No Null]) for quick reference.
- **Search:** A powerful search bar that allows you to filter the list by channel name or by attribute

(e.g., typing "event" will show all event channels).

- **Add/Remove:** Easily add new channels or remove existing ones.
- **Save Changes:** Commits your changes by re-generating the Channel.cs file and recompiling your project. An "Unsaved" warning will appear in the window title if you have pending changes.
- **Revert Changes:** Discards any unsaved changes you've made, restoring the list to its last saved state.

4. API Reference

Data Hub (Blackboard System)

- Eventus.Write<T>(Channel dataKey, T value)
  - Writes or updates a value in the Data Hub.
- Eventus.Read<T>(Channel dataKey)
  - Reads a value of type T from the Data Hub. Returns default(T) if the key is not found or the type is incorrect.

Event Bus (Notification System)

- Eventus.Publish(Channel type)
  - Fires a parameter-less event.
- Eventus.Publish<T>(Channel type, T data)
  - Fires an event with a data payload of type T.
- Eventus.Subscribe(Channel type, Action listener)
  - Subscribes a method to a parameter-less event.
- Eventus.Subscribe<T>(Channel type, Action<T> listener)
  - Subscribes a method to an event with a data payload of type T.
- Eventus.Unsubscribe(Channel type, Action listener) / Eventus.Unsubscribe<T>(Channel type, Action<T> listener)
  - **Crucial:** Unsubscribes a method to prevent memory leaks. Should typically be called in OnDisable.