



# NUI Light Mod

## Developer's Guide: Cross-Language Communication via File Bridge

---

This guide explains how to build a communication link between a high-level game mod (C#) and a visual interface (HTML/JS) for the **NUI Light mod**.






### Overview

The "**File Bridge**" method relies on a shared JSON file. The C# script acts as the **Producer** (writing data), and the HTML/JavaScript acts as the **Consumer** (reading and displaying data).



### Phase 1: The C# Producer

To ensure a working bridge, your C# script must follow three rules:

-  **Frequency Control:** Do not write to the disk 144 times a second. Use a timer (like the 33ms check above) to prevent SSD wear and performance lag.
-  **Concurrency:** You must use `FileShare.ReadWrite`. Without this flag, the HTML script will receive an "Access Denied" error because the C# script "owns" the file while writing.
-  **Atomic Writing:** Use `FileMode.Create` to overwrite the old data instantly so the JSON file always contains the most recent "snapshot".

## Example C# Implementation:

```
using System;
using System.IO;
using System.Timers;
using Newtonsoft.Json;

public class NuiDataBridge
{
    private Timer _timer;
    private readonly string filePath = @"HudData/hud_data.json";

    public void Start()
    {
        _timer = new Timer(33); // ~30fps
        _timer.Elapsed += WriteGameData;
        _timer.Start();
    }

    private void WriteGameData(object sender, ElapsedEventArgs e)
```

```
{  
    var gameData = new  
    {  
        hp = Game.Player.Health,  
        max = Game.Player.MaxHealth,  
        cash = Game.Player.Money,  
        timestamp = DateTime.Now.Ticks  
    };  
  
    string json = JsonConvert.SerializeObject(gameData);  
  
    using (var fs = new FileStream(filePath, FileMode.Create, FileAccess.Write,  
        FileShare.ReadWrite))  
        using (var sw = new StreamWriter(fs))  
        {  
            sw.Write(json);  
        }  
    }  
}
```



## Phase 2: The HTML/JS Consumer

The HTML script uses a **"Polling"** method. It asks the file system for the data at regular intervals.

### Example Fetch Code:

```
async function updateHUD() {  
  try {  
    // We add ?v= + Date.now() to prevent the browser from caching the file  
    const response = await fetch('HudData/hud_data.json?v=' + Date.now());  
    const data = await response.json();  
  
    // Update your UI elements here  
    document.getElementById('hp-bar').style.width = data.hp + "%";  
  } catch (err) {  
    // Handle cases where the file is being written to exactly when we try to read  
    console.log("Read error - retrying next frame");  
  }  
  // Repeat every frame or interval  
  requestAnimationFrame(updateHUD);  
}
```

## Phase 3: Handling Game-Specific Logic

Game data is often "raw" and needs translation for the UI.

### Health Conversion:

In GTA V, player health ranges from 100 (dead) to 200 (full).

**Calculation:** `let displayHP = (data.hp - 100) / (data.max - 100) * 100;`

## Currency Formatting:

Raw integers like 1500000 look better when formatted.

**Formatting:** `data.cash.toLocaleString(); // "$1,500,000"`

```
// Complete UI update with game logic
function processGameData(data) {
    // Convert GTA health (100-200) to percentage (0-100)
    let healthPercent = ((data.hp - 100) / (data.max - 100) * 100);
    healthPercent = Math.max(0, Math.min(100, healthPercent));

    // Format currency
    let formattedCash = '$' + data.cash.toLocaleString();

    // Update UI
    document.getElementById('hp-bar').style.width = healthPercent + '%';
    document.getElementById('cash-value').textContent = formattedCash;
}
```



## Summary of the Workflow

1 **C# Script** gathers raw data → converts to JSON

string → writes to hud\_data.json with ReadWrite sharing.

2 **HTML HUD** sits on top of the game → fetches

hud\_data.json 30-60 times a second → updates CSS widths and text values.

★ The Result is a real-time, high-performance HUD that can be styled with CSS without needing to recompile the C# mod.

## 📁 Example JSON Structure (hud\_data.json)

```
{
  "hp": 185,
  "max": 200,
  "cash": 1500000,
  "timestamp": 638326584921234567
}
```

## 🎨 Basic HTML Structure

```
<!DOCTYPE html>
<html>
<head>
  <style>
```

```
.health-bar { width: 200px; height: 20px; background: #333; }  
.health-fill { height: 100%; background: #ff4444; width: 0%; }  
.cash { color: gold; font-size: 24px; }  
  
</style>  
</head>  
<body>  
  <div class="health-bar">  
    <div id="hp-bar" class="health-fill"></div>  
  </div>  
  <div class="cash">${<span id="cash-value">0</span></div>  
  
  <script src="hud.js"></script>  
</body>  
</html>
```

⚡ For optimal performance, ensure your C# write interval matches your desired HUD refresh rate (typically 30-60fps).

🚀 The file bridge method provides a clean separation between game logic and UI presentation.

🔗 Mod Source: <https://www.gta5-mods.com/scripts/nuilight>