Flexible Internet Manager - Unity SDK Documentation

Overview

`FlexibleInternetManager` is a cross-platform Unity SDK that provides **robust, configurable internet connectivity detection** using multiple strategies. It is designed to work seamlessly across **WebGL, Android, iOS, PC, and Editor**, balancing **speed, accuracy, bandwidth usage, and battery efficiency**.

The SDK supports **instant checks**, **reliable HTTP verification**, **event-driven browser callbacks**, and **two-stage hybrid strategies**.

Key Features

- ✅ **Multiple checking strategies** (Browser, Unity API, HTTP, Hybrid)
- 🌍 **Platform-aware optimization** (WebGL, Mobile, Standalone)
- ⚡ **Instant detection** where possible
- 🎯 **High accuracy** with HTTP verification
- 🔋 **Battery & bandwidth efficient** hybrid strategies
- 🔁 **Continuous monitoring** support
- 📣 **Events for connection changes**
- 🧩 **Singleton-based global access**

Supported Platforms

Platform	Supported
WebGL	✅ Full (Browser + HTTP)

Android	✅
iOS	✅
Windows / macOS	✅
Unity Editor	✅

Architecture Summary

The SDK works by selecting one **Checking Strategy**, either manually or automatically:

1. **Instant check** (Browser or Unity API)
2. **Optional HTTP verification** (HEAD request)
3. **Event-based or polling-based monitoring**
4. **Status change propagation via events**

Checking Strategies

1. BrowserAPI (WebGL only)

Uses `navigator.onLine` via JavaScript interop.

- ⚡ Instant
- 🎯 Good accuracy
- 📡 Zero bandwidth
- 🔋 Minimal battery usage
- ❗ WebGL only

---

2. UnityAPI

Uses `Application.internetReachability`.

- ⚡ Instant
- 🎯 Medium accuracy (can give false positives)
- 📡 Zero bandwidth
- 🌐 All platforms

---

3. HTTP

Sends HTTP `HEAD` requests to multiple URLs.

- ⚡ Slower (100ms–10s)
- 🎯 Excellent accuracy
- 📡 ~1–3 KB per check
- 🔋 Higher battery usage

---

4. UnityAndHTTP (Recommended for Mobile)

Two-stage strategy:

1. Unity API (instant)
2. HTTP verification (only if network exists)

- ⚡ Fast when offline
- 🎯 Excellent accuracy
- 📡 Minimal bandwidth usage
- 🔋 Battery efficient

---

5. BrowserAndHTTP (Recommended for WebGL)

Two-stage WebGL strategy:

1. Browser API (event-driven)
2. HTTP verification

- ⚡ Instant detection
- 🎯 Excellent accuracy
- 📡 Minimal bandwidth
- 🌐 WebGL only

Singleton Usage

FlexibleInternetManager.Instance

The manager persists across scenes using `DontDestroyOnLoad`.

Inspector Configuration

Strategy Selection

- **Strategy**: Manual strategy selection
- **Auto Optimize For Platform**: Automatically selects best strategy

General Settings

- **Auto Check On Start**: Runs a check on startup
- **Continuous Monitoring**: Enables periodic checking
- **Check Interval**: Interval between checks (seconds)
- **Timeout**: HTTP request timeout (seconds)

HTTP Settings

- **Check URLs**: List of URLs used for HTTP verification

Debug Settings

- **Show Debug Logs**: Enable logs
- **Show Detailed Logs**: Enable verbose logs

Public Properties

bool IsConnected

bool IsChecking

CheckingStrategy CurrentStrategy

bool IsWebGLBuild

float CheckInterval

Public Methods

CheckInternetConnection()

Checks internet using the active strategy.

```
FlexibleInternetManager.Instance.CheckInternetConnection();
```

SetStrategy(CheckingStrategy newStrategy)

Change strategy at runtime.

```
FlexibleInternetManager.Instance.SetStrategy(
    FlexibleInternetManager.CheckingStrategy.HTTP
);
```

StartMonitoring() / StopMonitoring()

Enable or disable continuous monitoring.

```
FlexibleInternetManager.Instance.StartMonitoring();
FlexibleInternetManager.Instance.StopMonitoring();
```

CheckConnectionWithUrl(string url, Action)

Manual URL-based connection check.

```
FlexibleInternetManager.Instance.CheckConnectionWithUrl(
    "https://example.com",
    result => Debug.Log(result)
);
```

Events

OnInternetConnected

Triggered when connection is restored.

OnInternetDisconnected

Triggered when connection is lost.

OnConnectionStatusChanged(bool isConnected)

Triggered on any connection state change.

```
FlexibleInternetManager.Instance.OnInternetDisconnected += () =>
{
    Debug.Log("Internet Lost");
};
```

Utility Methods

```
NetworkReachability GetUnityReachability()
```

```
bool GetBrowserOnlineStatus()
```

string GetConnectionType() // WiFi / Mobile Data / None

bool IsMeteredConnection()

WebGL JavaScript Integration

This SDK expects a `.jslib` file implementing:

- `IsOnline()`
- `RegisterOnlineCallback()`
- `RegisterOfflineCallback()`
- `UnregisterCallbacks()`

Used for `navigator.onLine` and browser events.

Recommended Strategy by Platform

Platform	Recommended Strategy
WebGL	BrowserAndHTTP
Android	UnityAndHTTP
iOS	UnityAndHTTP
PC / Mac	HTTP

Best Practices

- ✅ Enable **Auto Optimize For Platform** for most projects
- ✅ Use **UnityAndHTTP** for mobile games
- ✅ Use **BrowserAndHTTP** for WebGL builds
- ❌ Avoid pure HTTP checks at very small intervals

Example Use Case

- Show **"No Internet" popup** when disconnected
- Pause gameplay on loss of connectivity
- Resume game automatically when internet returns

Versioning & Extensibility

The SDK is designed to be easily extended:

- Add new strategies
- Plug into analytics
- Add latency checks
- Add captive portal detection

Summary

`FlexibleInternetManager` is a **production-ready, extensible, and platform-aware Unity internet connectivity SDK**.

⚠️ CORS Warning (WebGL Only)

When using **HTTP-based strategies** (`HTTP`, `UnityAndHTTP`, `BrowserAndHTTP`) in **WebGL**, browser **CORS (Cross-Origin Resource Sharing)** rules apply.

Browsers block HTTP requests to domains that **do not allow cross-origin access**, which can cause **false "No Internet" results** even when the user is online.

❌ URLs that will NOT work in WebGL (CORS blocked)

`https://www.google.com`

`https://www.youtube.com`

`https://www.facebook.com`

`https://www.amazon.com`

`https://www.microsoft.com`

`https://www.cloudflare.com`

✅ Recommended approach

- Prefer **BrowserAPI** or **BrowserAndHTTP** for WebGL
- Use **your own backend URL** (e.g. `/ping`, `/health`) with CORS enabled

Ensure server headers:

```
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET, HEAD
```

- ⚠️ **Note:** CORS is a browser security restriction, not a Unity limitation.
  This SDK cannot bypass CORS.

Author: Abhishek Sahu
Category: Connectivity / Network Utility