Version 1.1.0

Introduction

Do you want to **truly** know your game can reach the internet? **Eazy NetChecker** is the right tool! Eazy NetChecker is a **light** but **reliable** and **powerful** tool for **really** determining the status of the internet connection on all devices and platforms.

Methods like Unity's `Application.internetReachability` cannot truly tell if a device is actually connected to the internet, and neither was designed to do so. Moreover, simple methods like pinging google are unreliable since devices could appear to be connected to the internet, but in reality be behind a restricted network. **Eazy NetChecker is the solution to all that!** By utilizing a technique called **Captive Portal Detection**, it can quickly and reliably determine the current internet connection.

Captive Portal Detection is a technique that is used in all major operating systems for detecting internet connectivity.

Features

- Reliable detection of internet connection status (Pending, NoConnection, WalledGarden, Connected)
- Pre-specified captive portal detection check methods (Google204, MicrosoftNCSI, AppleHotspot)
- Custom check methods
- Automatically select the best check method for each build platform
- Control over when checks are performed

    - **On Start:** Run a check when your game is loaded
    - **Continuously:** Automatically run checks on configurable intervals
    - **Manual:** Run a single check whenever you need it.
- Stats and info about internet status and connectivity
- Helpful events to listen to (OnStart, OnFinish, OnStatusChanged, OnTimeout)
- Runtime API (Fully documented)
- Easy to use Editor
- Super easy integration
- **Full C# source code**

Compatibility

- Multi-platform support (Windows, Mac, Linux, Android, iOS)
- Unity 5.6 or higher

Download

You can download the plugin from the Unity Asset store

Installation

Installation is super easy. You can get Eazy NetChecker working in just a few minutes. First, download the Unity package from the Unity Asset Store. When the download is finished, the Import window should appear, listing all the files and folders in the package. Make sure you have them all selected, and press **Import**. Unity should now start importing the assets in the package.

Content

The imported package includes all the scripts for Eazy NetChecker, as well as demo scenes for demonstrations. Some editor resources are also included. The main folder will be stored at `Assets/Hellmade/Eazy NetChecker`

- **Demo files:** This folder included example scenes and all demo files `Assets/Hellmade/Eazy NetChecker/Demo`
- **Scripts:** All the scripts required for Eazy NetChecker `Assets/Hellmade/Eazy NetChecker/Scripts`
- **Resources:** Various resources we use for Eazy NetChecker, mostly for the editor. Please do not remove this folder `Assets/Hellmade/Eazy NetChecker/Resources`
- **Common:** Various global resources we use across all of our assets, usually for the editors. Please do not remove this folder `Assets/Hellmade/Common`

Getting Started

Setting Eazy NetChecker up and using it is super easy and straightforward. It is also extremely flexible since it allows you to use either the editor to set it up, or do everything with code on runtime. Of course, using both is also possible. Everything will be explained for both ways.

Setup

Editor

The only initial setup needed for the editor, is to create the Eazy NetChecker GameObject. This can easily be done by either navigating to *GameObject > Hellmade Games* and selecting *Eazy NetChecker*. This will create a GameObject in your scene with the **Eazy NetChecker** component on it. You can always create an empty GameObject and add the **Eazy NetChecker** component yourself too.

Do not create or have more than one Eazy NetChecker in the same scene. Eazy NetChecker retains its life over different scenes, so you only need to create it once, in your first scene.
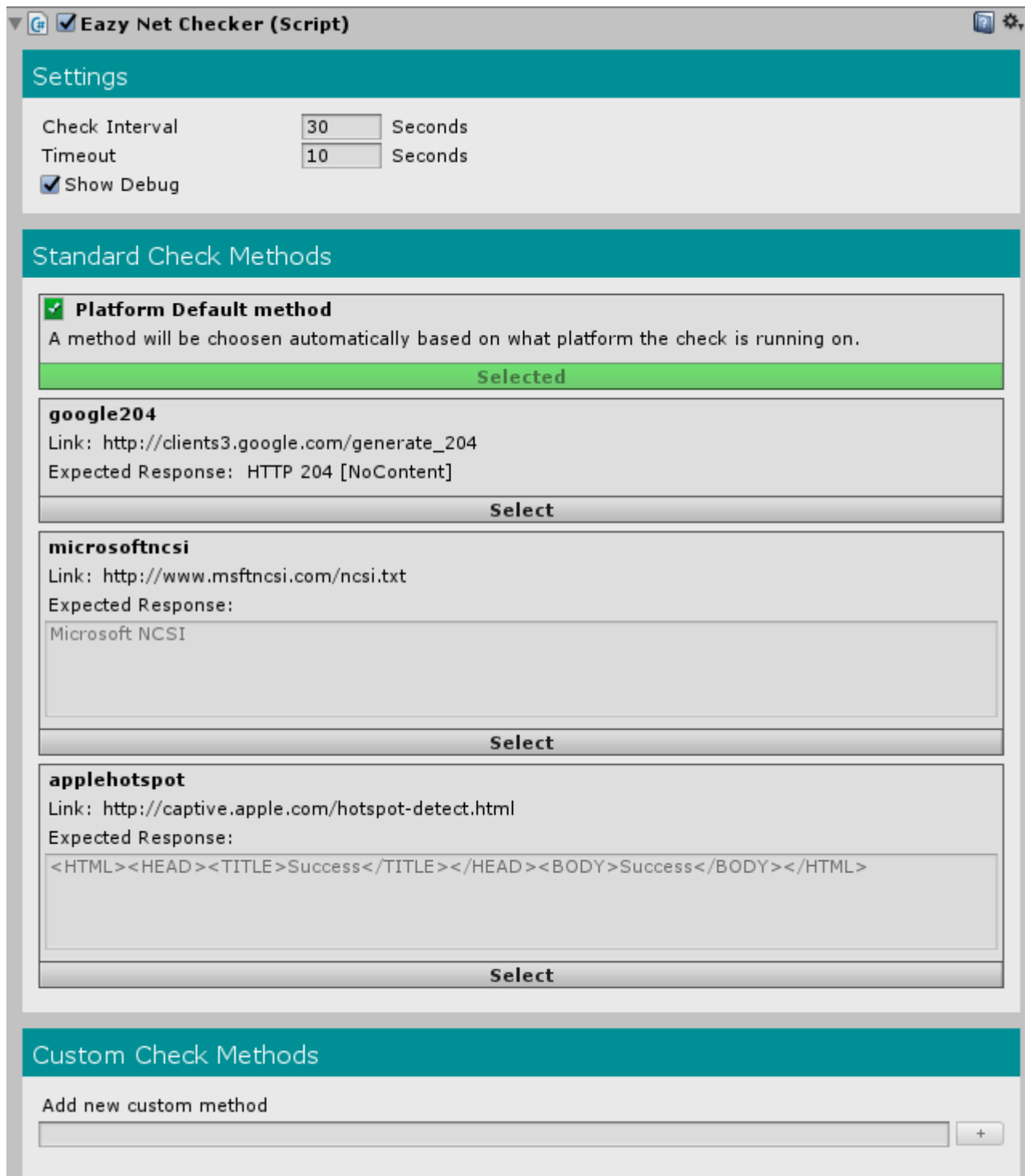
Eazy NetChecker editor

Runtime

If you want to handle everything with code, there is no reason to add an Eazy NetChecker GameObject. It will automatically be created and initialized for you as soon as you reference it for the first time in your code.

Make sure to always include the namespace Hellmade.Net when using Eazy NetChecker from code. Just use this on the top of every script that uses Eazy NetChecker:

using Hellmade.Net;

Check Internet connection

Checking for internet connection can be done in a few ways. You can manually perform a check once, start a continuous check which will be performed on a specified interval, or perform a check on start up.

Before you perform any kind of check, you need to select which [check method](#) you want to use. For the standard check methods, just use the appropriate check method selection function:

```
EazyNetChecker.UseGoogle204Method();
```

You can also select a check method in the editor before runtime
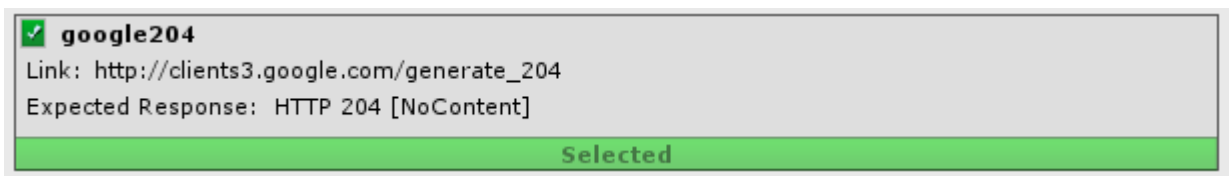
If no check method is specifically selected, one of the standard check method will be choosen automatically, based on the platform running:

- Windows (MicrosoftNCSI)
- Linux/Android (Google204)
- MacOS/iOS (AppleHotspot)

Manual Check

The simplest case would be to perform a single manual check:

```
private void Awake()
{
    EazyNetChecker.UseGoogle204Method();
    EazyNetChecker.OnCheckFinished += OnNetCheckFinished;
    EazyNetChecker.CheckConnection();
}

private void OnNetCheckFinished()
{
    Debug.Log(EazyNetChecker.Status);
}
```

This will start a process for checking the current internet connection status in the background. The check method will raise various [events](#) during its lifetime. In this example, we added a listener to the **OnCheckFinished** event, which will be raised as soon as a check is finished. Here the check is performed on awake, but of course you can perform an internet check whenever and wherever you need to.

Continuous Check

You can also start a continuous check, which will run on a specified interval, until you decide to stop it:

```
    private void Awake()
    {
        EazyNetChecker.UseGoogle204Method();
        EazyNetChecker.OnConnectionStatusChanged += OnNetStatusChanged;
        EazyNetChecker.StartConnectionCheck();
    }


    private void OnNetStatusChanged()
    {
        Debug.Log("Internet Connection Status changed to: " + EazyNetChecker.Status);
    }
```

The above example will perform a check every 20 seconds (default). It also listens to the **OnConnectionStatusChanged** event, which is raised everytime the status of the internet connection is changed (after a check is performed).

You can use whatever events you like, depending on your specific game logic. You can even not use an event, and implement your logic by checking the current detected internet connection status like the example below:

```
    private void Awake()
    {
        EazyNetChecker.UseGoogle204Method();
        EazyNetChecker.StartConnectionCheck();
    }


    private void Update()
    {
        if(EazyNetChecker.Status == NetStatus.Connected)
        {
            Debug.Log("Yeyyyy, I have internet!");
        }
        else
        {
            Debug.Log("No internet :(");
        }
    }
```

Settings

The check interval (time between continuous checks), and the timeout time can also be changed. Eazy NetChecker can be configured using both the editor and the runtime API.

Editor

Configuring the settings through the editor is very straightforward. All of them can be found in the **Settings** section

Settings section

Runtime

All settings can be set before you perform a check:

```
private void Awake()
{
    EazyNetChecker.CheckInterval = 30f;
    EazyNetChecker.Timeout = 15f;
    EazyNetChecker.ShowDebug = true;

    EazyNetChecker.UseGoogle204Method();
    EazyNetChecker.StartConnectionCheck();
}
```

Check Methods

Check methods are responsible for determining the current internet connection. They are all using the **Captive Portal Detection** technique, which is used by all major operating systems. However, there are different check methods available. While they all work in the same way, the data they use to determine whether internet connection can be established can differ. Eazy NetChecker comes with a few ready to use check methods, but is also extendable by allowing you to set custom check methods to fit your requirements.

Standard Check Methods

As already stated, Eazy NetChecker uses check methods that major operating systems are using. In particular, the following methods come ready to be used in Eazy NetChecker

Google 204

Google 204 is one of the captive portal detection methods that Google uses in its own operating systems and devices. You can use Google204 by selecting it in the editor, or by calling the following function

```
EazyNetChecker.UseGoogle204Method();
```

Google204 is just an empty page hosted by google. The method just expects an HTTP 204 status code from it.

Microsoft Connect Test

Microsoft Connect Test is a text file hosted by Microsoft, and is the method used by Microsoft for internet status detection. Select it from the editor, or use the following function:

```
EazyNetChecker.UseMicrosoftConnectTestMethod();
```

Unlike Google204, Microsoft Connect Test expects certain content to be returned from the HTTP request, instead of a status code. Specifically it expects to get the following text

```
Microsoft Connect Test
```

Apple Hotspot

Apple hotspot is very similar to Microsoft NCSI, but it is used by Apple. Use it by selecting it in the editor or by using the following function:

```
EazyNetChecker.UseAppleHotspotMethod();
```

Apple Hotspot expects to get a specific HTML code as content:

Custom Check Methods

Eazy NetChecker is fully flexible and extentable since it allows you to create your own custom check methods to specifically fit your needs. You may ask why would you want a custom one since the standard check methods are from major companies and are supposed to be reliable. Well, take for example the case where you want to download something from your own server in your game. Internet connection may be present and working fine, therefore a Google204 will detect a connected network. However, your server may actually be down and therefore your game cannot really connect to it. While general internet connection may be working fine, the connection you actually want to establish is unavailable. Custom check methods are used for cases like that.

Creating Check Method Pages

First, you need to create the check method page exactly as you like it, and upload it to your server. This will act as the check link. You will also need to decide whether you want to check against HTTP status code, or the content of the link.

For example, let's create a custom check method for Hellmade Games (You can name your check method to match your needs). First, create an empty html page and name it *connectioncheck.html*. In this example, we will use page content as the expected response. Just put a simple text in your page that will act as the expected content:

```
Connection Check
```

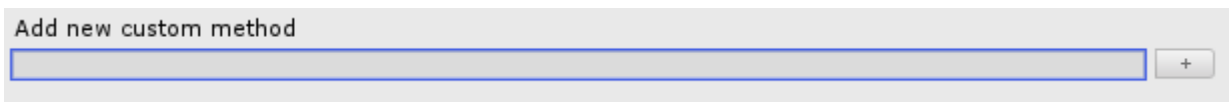That's it for the html page. Now just upload it on your server. You can find ours at http://www.hellmadegames.com/connectioncheck.html

If you are targeting Android 9+ and/or iPhone, make sure to use an HTTPS link. Cleartext HTTP traffic is not permitted on such devices, and therefore the check method will not work correctly. If you absolutely need to use an HTTP link, please check the documentation of the aforementioned devices on how to allow cleartext HTTP.

Check Method Setup

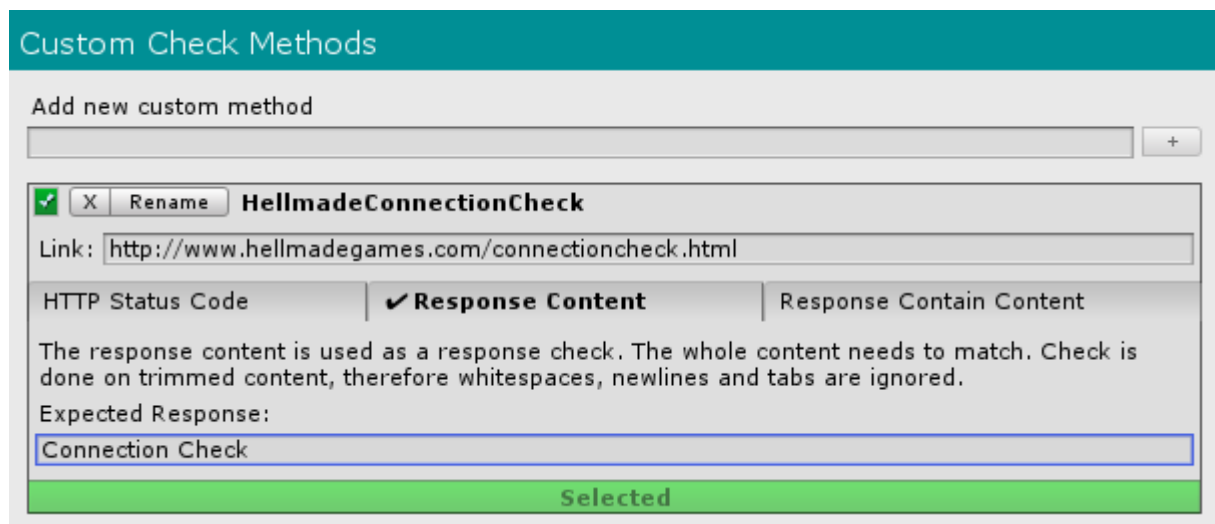Now, you just need to set it up in Eazy NetChecker. You can do it both in the editor, or during runtime from code.

Editor

All you need to do is create a new custom check method by writing its ID (name) and click the **+** button. The new check method will be added to the list. In the link field, just add the link to your uploaded html page. Then select the expected response type from the tabs below (**Response content** in our example), and just write down what the expected content is (if content is selected), or select the expected HTTP status code (if HTTP status code is selected).

Create a new check method by writing its ID and pressing the + button

You are all done. If you want to use your new method, just select it.

Runtime

Creating a custom method during runtime is still super easy:

```
string link = "http://www.hellmadegames.com/connectioncheck.html";
string expectedContent = "Connection Check";
NetCheckMethod hellmadeMethod = new NetCheckMethod( "HellmadeCheckMethod", link,
expectedContent, false);
```

If your check method is using an HTTP status code instead, you can create it in a very similar way (let's say you want an HTTP 204):

```
string link = "http://www.hellmadegames.com/connectioncheck.html";
HttpStatusCode expectedHttpStatusCode = HttpStatusCode.NoContent;
NetCheckMethod hellmadeMethod = new NetCheckMethod( "HellmadeCheckMethod", link,
expectedHttpStatusCode);
```

You can now either add it to your custom methods and use it:

```
EazyNetChecker.AddCustomMethod(hellmadeMethod, true);
```

Or just use it without adding it:

```
EazyNetChecker.UseMethod(hellmadeMethod);
```

Adding it to your custom methods will make it available to use later on if for some reason you select another check method in between.

Check methods created during runtime are not retained after execution is finished. If you want to setup check methods that are saved, create them in the editor.

Events

Various events are raised during the lifetime of an internet check. Events are very useful become they allow you to implement various different logics based on your requirements, when it comes to network status detection. You can listen to them from as many scripts as you want.

Make sure to not listen to events twice from the same script, or keep listening to them when not wanted.

OnCheckStarted

This event is raised as soon as an internet check is started. It will be raised for **every** internet check, if you have started a continuous check.

You can listen/stop listening to this event this way:

```
...

// Start listening
EazyNetChecker.OnCheckStarted += OnNetCheckStarted;

...

// Stop listening when you don't need it
EazyNetChecker.OnCheckStarted -= OnNetCheckStarted;

...

private void OnNetCheckStarted()
{
    Debug.Log("Internet check just started");
```

```
        }
```

OnCheckFinished

This event is raised as soon as an internet check is finished. It will be raised for **every** internet check, if you have started a continuous check.

You can listen/stop listening to this event this way:

```
    ...

    // Start listening
    EazyNetChecker.OnCheckFinished += OnNetCheckFinished;

    ...

    // Stop listening when you don't need it
    EazyNetChecker.OnCheckFinished -= OnNetCheckFinished;

    ...

    private void OnNetCheckFinished()
    {
        Debug.Log("Internet check just finished with status " + EazyNetChecker.Status);
    }
```

OnConnectionStatusChanged

This event is raised as soon as an internet check is finished and a different connection status is detected. You can use this event to act accordingly if the connection status changes and you want to handle connection-specific logic.

You can listen/stop listening to this event this way:

```
    ...

    // Start listening
    EazyNetChecker.OnConnectionStatusChanged += OnNetConnectionStatusChanged;

    ...

    // Stop listening when you don't need it
    EazyNetChecker.OnConnectionStatusChanged -= OnNetConnectionStatusChanged;

    ...

    private void OnNetConnectionStatusChanged()
    {
        Debug.Log("Internet status changed to " + EazyNetChecker.Status);
    }
```

OnCheckTimeout

This event is raised when an internet check times out. An internet check will time out if it is not able to detect the internet connection status within the specified timeout period.

You can listen/stop listening to this event this way:

```
...

// Start listening
EazyNetChecker.OnCheckTimeout += OnNetCheckTimeout;

...

// Stop listening when you don't need it
EazyNetChecker.OnCheckTimeout -= OnNetCheckTimeout;

...

private void OnNetCheckTimeout()
{
    Debug.Log("Check timed out! Unable to determine internet connection");
}
```

API Reference

You can access the API reference online at: http://www.hellmadegames.com/projects/eazy-netchecker/docs/api-reference

You can also find an offline version included in the package

Please use the online version whenever you can, since it will always be up to date faster.

Support

If you need help, have a question or want to recommend future features, please feel free to contact us and we will get back to you as soon as possible.

You can either send us an email, or even contact us using Facebook:

Email: support@hellmadegames.com

Facebook: https://www.facebook.com/hellmadegames

Change Log

[1.1.0] - 13-03-2019

Added

- Warning messages on editor (Invalid link & unsafe cleartext HTTP link)
- Connection statistics (Editor & API)

Changed

- Standard check methods (Secure version of google204, Miscrosoft NCSI to Microsoft Connect Test, Secure version of apple hotspot)
- NetCheckMethod implementation from WWW to UnityWebRequest

Fixed

- Non-stop debug message after time out.