



[Download from the Unity Asset Store](#) | [Discord Support](#) | [Mail Contact](#) |

# **ImaginationOverflow**

**2018**

Ana Correia

Diogo Cardoso

## Contents

Getting Started .....	6
Using the Plugin .....	6
Configuring the Plugin .....	8
Global Configuration.....	10
Per Platform Configuration .....	11
Testing .....	13
Editor .....	13
iOS .....	13
Android.....	13
Windows 10.....	14
MacOS .....	14
Linux .....	14
Standalone Caveats .....	15
Build for MacOS.....	15
Linux and Windows.....	16
Notes .....	16
Domain Association .....	17
iOS - Universal Links .....	17
Android - App Links.....	19
Windows UWP - Uri Handling.....	21
Notes .....	22
Steam Integration.....	23
Configuring Steam Integration.....	23
Mac and Steam Integration .....	24
Solution Suggestion .....	24
Inside the Plugin .....	27
Overview .....	27
Runtime.....	28
Android.....	28
iOS .....	28
Windows UWP (Windows Store Games) .....	28

Linux and Windows .....	28
Mac.....	29
Showcase.....	30
Sudoku Zenkai .....	30
Super Bunny Laser Spikes .....	30
About.....	31
Authors.....	31
Links.....	31
Acknowledgements .....	31

ImaginationOverflow Universal Deep Linking plugin enables your games to use Deep Linking and Domain Association in iOS, Android, Windows 10 (UWP), Standalone (Linux, Mac, and Windows).

Deep Linking "consists of using a uniform resource identifier (URI) that links to a specific location within a mobile app rather than simply launching the app."  
([Mobile deep linking @ Wikipedia](#))

Domain Association is a similar feature but instead of using custom URIs (mycustomuri://somecontent) it utilizes Web URIs (www.mydomain/somecontent) to directly open your games on a specific location

Each mobile platform has a specific name for this feature, iOS calls it [Universal Links](#), Android has its [App links](#) and Windows 10 named it [Uri handling](#), throughout this documentation we will refer the ability to associate a web URI with your game as Domain Association.

We will be using our game [Sudoku Zenkai](#) ([iOS](#), [Android](#), [Windows 10](#) and [Steam](#)) as a use case throughout this documentation. Anyone that purchases the plugin is entitled to a free Steam key to check a live example of this plugin usage, to get it just contact us on our [Discord channel](#) or via [mail](#)

Universal Deep Linking plugin enables your games to use Deep Linking on iOS, Android, Windows (UWP), Windows, Mac, Linux and Steam (Linux, Mac, and Windows).

Currently, the plugin supports Domain Association to iOS, Android and Windows UWP (Mobile, Tablet, and Desktop).

## Getting Started

- [How to Use](#)
- [Configuring the Plugin](#)
- [Testing](#)

## Domain Association

- [iOS](#)
- [Android](#)
- [Windows UWP \(Windows Store Apps\)](#)

## Steam Integration

- [Configuring the Plugin](#)
- [Mac integration details](#)

## Inside the Plugin

- [Overview](#)
- [Details on all supported platforms](#)

## Showcase

- [Games using the Plugin](#)

# Getting Started

## Using the Plugin

The plugin uses a single event where you need to register in order to receive Deep Linking or Domain Association activations:

```
void Start()
{

    ImaginationOverflow.UniversalDeepLinking.DeepLinkManager.Instance.LinkActivated += Instance_LinkActivated;
}

private void
Instance_LinkActivated(ImaginationOverflow.UniversalDeepLinking.LinkActivations)
{
    //
    // my activation code
    //
}
```

Never forget to remove your event registration when the **GameObject** where you registered it is **destroyed**:

```
void OnDestroy()
{

    ImaginationOverflow.UniversalDeepLinking.DeepLinkManager.Instance.LinkActivated -= Instance_LinkActivated;
}
```

The **LinkActivated** event will be triggered when your game is **started** or **resumed** by a Deep Link or Domain activation.

The **LinkActivated** event single argument **LinkActivation** contains the Uri that triggered the game activation, a string with the raw **Query String** and a Dictionary with the query string already processed:

```
public class LinkActivation
{
    public string Uri { get; private set; }

    public string RawQueryString { get; private set; }

    public Dictionary<string, string> QueryString { get; private set; }
}
}
```

Consider the following example:

```
private void
Instance_LinkActivated(ImaginationOverflow.UniversalDeepLinking.LinkAc
tivation linkActivation)
{
    var url = linkActivation.Uri;
    var querystring = linkActivation.RawQueryString;
    var qParameter = linkActivation.QueryString["q"];
}
```

If Sudoku Zenkai is activated with the following link:

<https://sudokuzenkai.imaginationoverflow.com/dailychallenge?q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw4TDhMOEw4TDhMOEw4TCIA==>

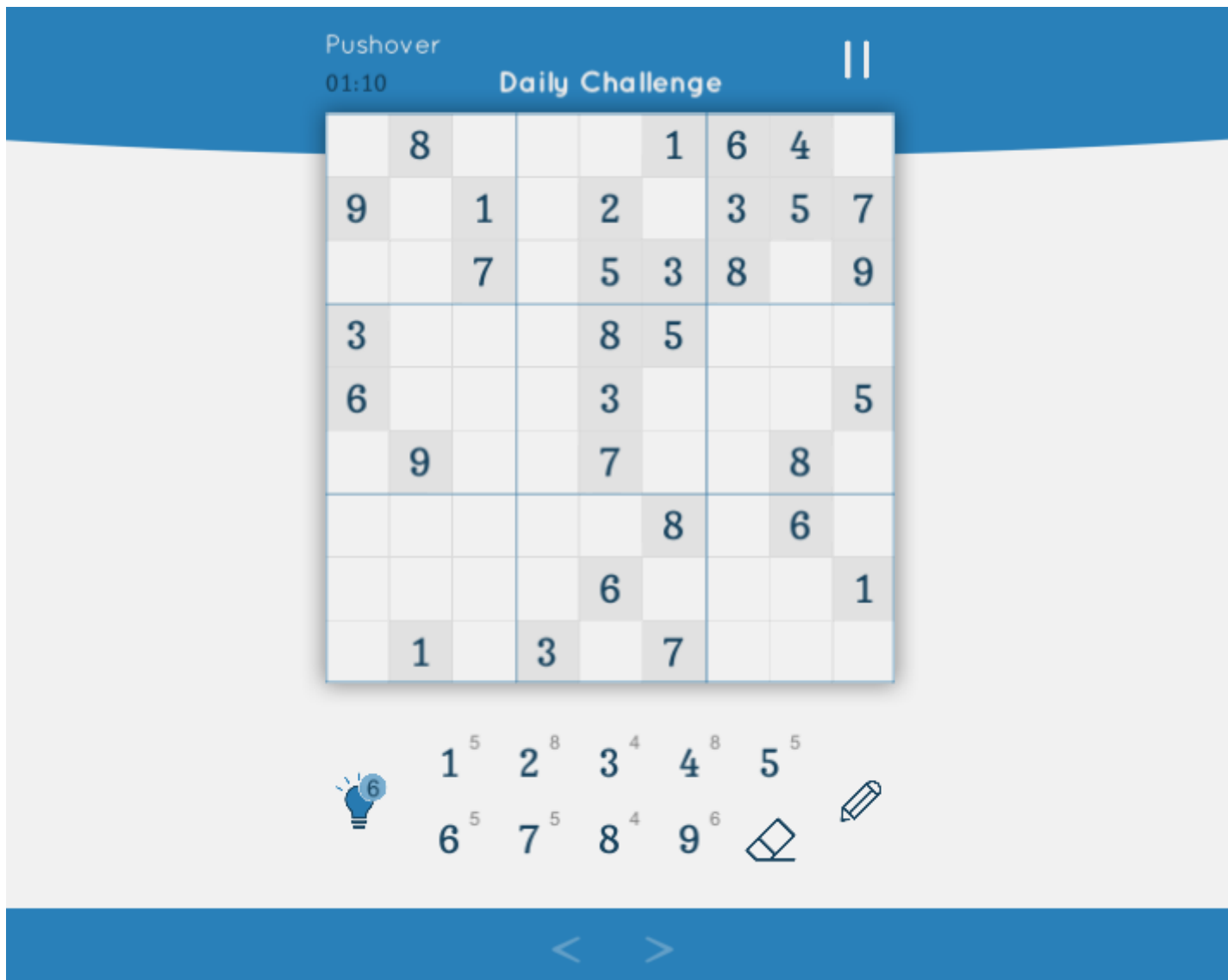
The variables values would be:

- **url** - <https://sudokuzenkai.imaginationoverflow.com/dailychallenge?q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw4TDhMOEw4TDhMOEw4TCIA==>
- **querystring** - q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw4TDhMOEw4TDhMOEw4TCIA==
- **qParameter** - ZMORw4TDhcOKw4fDisOKw4jDhMOFw4bDjMOEw4TDhMOEw4TDhMOEw4TCIA==

If you have the game installed on your device right now, instead of using the site URI you can use the deep linking URI instead:

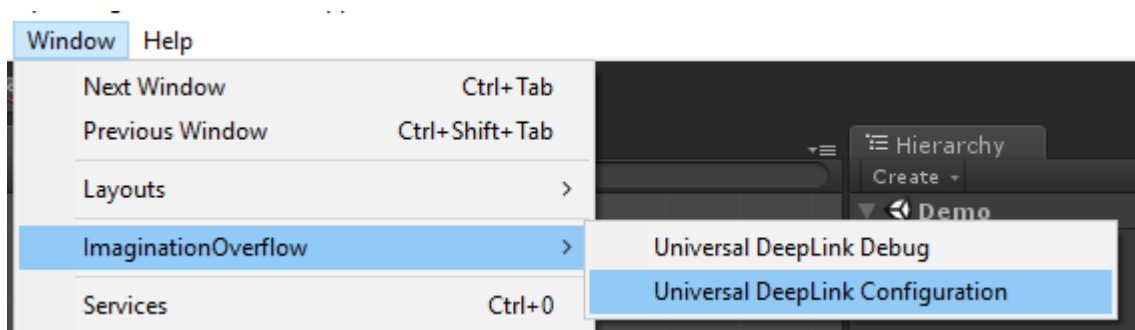
[sudokuzenkai://dailychallenge?q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw4TDhMOEw4TDhMOEw4TCIA==](https://sudokuzenkai://dailychallenge?q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw4TDhMOEw4TDhMOEw4TCIA==)

You should be seeing the exact puzzle as seen below.

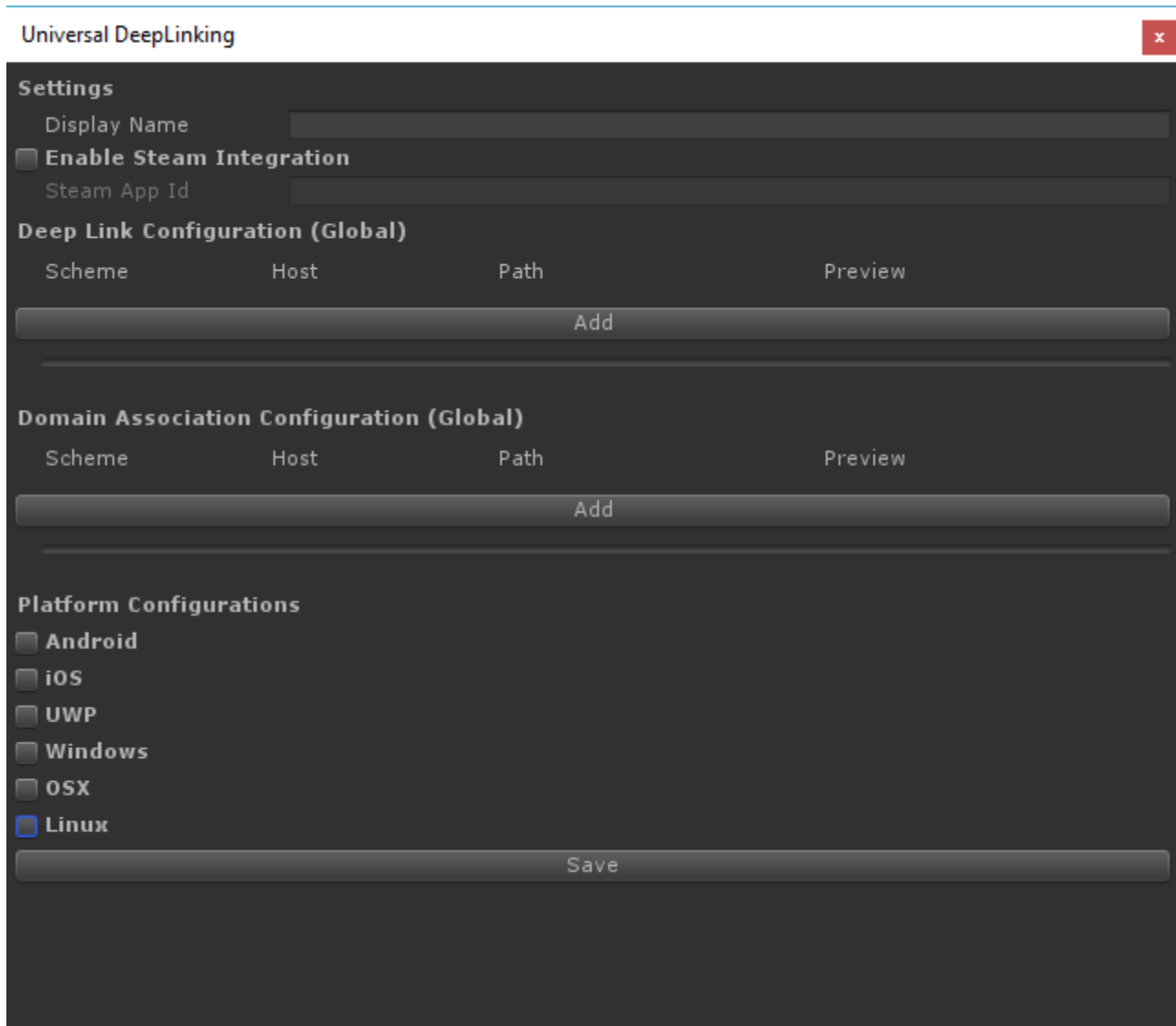


## Configuring the Plugin

The configuration interface is under **Window** -> **ImaginationOverflow** -> **Universal DeepLink Configuration**







On some platforms, the operating system asks the player what app she wishes to use after clicking a custom URI, the **Display Name** allows you to configure what name will the OS show on that occasion. The **Steam App Id** is for Steam only games, you can read about Steam integration on its [section](#). The plugin allows you to configure the deep linking and domain association globally or per platform.

## Global Configuration

### Universal DeepLinking

**Settings**

Display Name

**Enable Steam Integration**

Steam App Id

**Deep Link Configuration (Global)**

Scheme	Host	Path	Preview
<input type="text" value="sudokuzenkai"/>	<input type="text"/>	<input type="text"/>	<input type="button" value="Delete"/> sudokuzenkai://*

**Domain Association Configuration (Global)**

Scheme	Host	Path	Preview
<input type="text" value="http"/>	<input type="text" value="sudokuzenkai.imaginationoverflow.com"/>	<input type="text"/>	<input type="button" value="Delete"/> http://sudokuzenkai.imaginationoverflow.com/*
<input type="text" value="https"/>	<input type="text" value="sudokuzenkai.imaginationoverflow.com"/>	<input type="text"/>	<input type="button" value="Delete"/> https://sudokuzenkai.imaginationoverflow.com/*

**Platform Configurations**

**Android**

**iOS**

**UWP**

**Windows**

**OSX**

**Linux**

If you are using the same custom URIs and Domains for every platform, you only need to setup it once and the plugin will automatically propagate all data to all platforms as you build for them.

To configure a Deep Link or a Domain Association you need to provide:

- **Scheme** - it can be anything you wish, remember to check the Stores for other apps or games that are also using an URI you wish to adopt (when configuring for deep linking), for domain association it's usually **http**, if your site supports **https** add another entry to the list.
- **Host** - this parameter is usually only used for domain association, here you should put the host of your website
- **Path** - some platforms support which **paths** the application should be activated, but usually this is accomplished using other configuration files see [DOMAIN ASSOCIATION CONFIG FILES].

## Per Platform Configuration

When you wish to override any configuration for a specific platform you can do that by clicking on the specific platform checkbox. After that, you just need to fill out the Deep Linking and Domain Association data or leave it empty if you do not wish to support these features for that specific platform. Note that by checking a platform none of the global configurations will be used for that specific platform.

**Settings**

Display Name

**Enable Steam Integration**

Steam App Id

**Deep Link Configuration (Global)**

Scheme	Host	Path	Preview
<input type="text" value="sudokuzenkai"/>	<input type="text"/>	<input type="text"/> <input type="button" value="Delete"/>	sudokuzenkai://*

**Domain Association Configuration (Global)**

Scheme	Host	Path	Preview
<input type="text" value="http"/>	<input type="text" value="sudokuzenkai.imaginationoverflow.com"/>	<input type="text"/> <input type="button" value="Delete"/>	http://sudokuzenkai.imaginationoverflow.com/*
<input type="text" value="https"/>	<input type="text" value="sudokuzenkai.imaginationoverflow.com"/>	<input type="text"/> <input type="button" value="Delete"/>	https://sudokuzenkai.imaginationoverflow.com/*

**Platform Configurations**

**Android**

**Deep Link Configuration (Android)**

Scheme	Host	Path	Preview
<input type="text" value="sudokuzenkaidroid"/>	<input type="text"/>	<input type="text"/> <input type="button" value="Delete"/>	sudokuzenkaidroid://*

**Domain Association Configuration (Android)**

Scheme	Host	Path	Preview
--------	------	------	---------

- iOS**
- UWP**
- Windows**
- OSX**
- Linux**

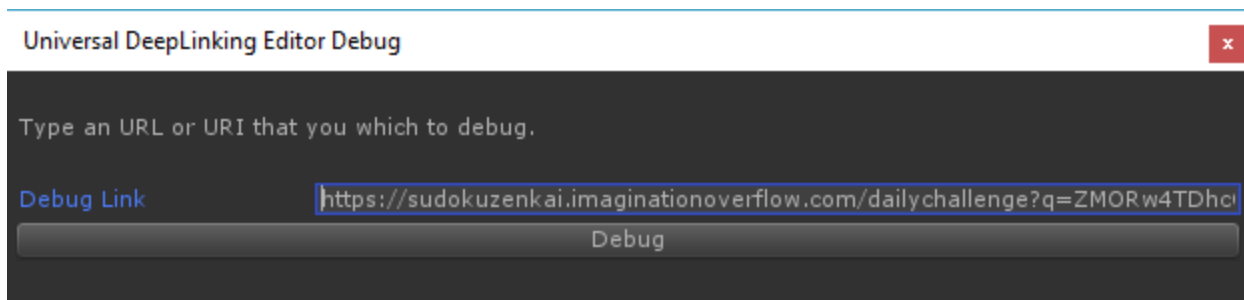
In the example above, we've changed the **Android Deep Linking scheme** to **sudokuzenkaidroid** and disabled the Domain Association capability. The remaining platforms will continue to use the global configurations.

## Testing

After your game is deployed to a device, you can test the integration simply by clicking on a configured URI on any app or website. As an example, you can send yourself a message (on any messaging app) or an email with the proper URI.

### Editor

We included a simple interface where you can test your deep linking activation without deploying, you can insert any valid and invalid URIs in order to test your integration.



When you press Debug, the LinkActivated will be triggered and your callback called if the application is running in the editor.

### iOS

#### *Simulator*

After your game is installed on a simulator, open a terminal and run the following command:

```
xcrun simctl openurl booted "[MY_URI_HERE]"
```

### Examples

```
xcrun simctl openurl booted
"sudokuzenkai://dailychallenge?q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw
4TDhMOEw4TDhMOEw4TC1A=="
xcrun simctl openurl booted
"https://sudokuzenkai.imaginationoverflow.com/dailychallenge?q=ZMORw4T
DhcOKw4fDisOKw43DhcOFw4TDiMOEw4TDhMOEw4TDhMOEw4TC1A=="
```

### Android

Open a terminal on your development machine, open a terminal (console or powershell) and run the command:

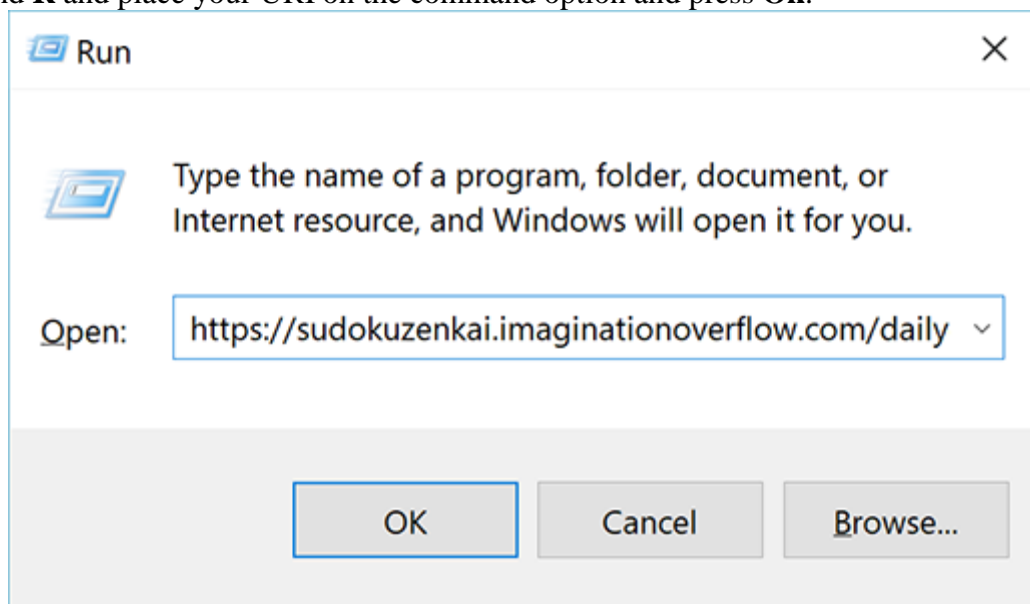
```
adb shell am start -W -a android.intent.action.VIEW -d "[MY_URI_HERE]"
```

## Examples

```
adb shell am start -W -a android.intent.action.VIEW -d
"sudokuzenkai://dailychallenge?q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw4
4TDhMOEw4TDhMOEw4TC1A=="
adb shell am start -W -a android.intent.action.VIEW -d
"https://sudokuzenkai.imaginationoverflow.com/dailychallenge?q=ZMORw4T
DhcOKw4fDisOKw43DhcOFw4TDiMOEw4TDhMOEw4TDhMOEw4TC1A=="
```

## Windows 10

You can use windows run program to test your integration, just click on the **Windows Button** and **R** and place your URI on the command option and press **Ok**.



## MacOS

Open up a terminal and use the **Open** command:

```
open [MY_URI_HERE]
open
sudokuzenkai://dailychallenge?q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw4
TDhMOEw4TDhMOEw4TC1A==
```

## Linux

On Linux you can use the xdg-open command to test your integration, open up a terminal and run the following command.

```
xdg-open [MY_URI_HERE]
xdg-open
sudokuzenkai://dailychallenge?q=ZMORw4TDhcOKw4fDisOKw43DhcOFw4TDiMOEw4
TDhMOEw4TDhMOEw4TC1A==
```

# Standalone Caveats

## Build for MacOS

If you make your Mac builds on MacOS the plugin will automatically configure everything that is required in order for it to work correctly. But if you make your MacOS build on Windows you will need to do some extra steps in order to fully configure your build to receive deep link activations. Due to Unity limitations, you **will need a Mac or a MacOS VM** in order to completely use the plugin capabilities.

### *Building for MacOS on Windows*

When the build is completed you will note that the build process created an extra folder named **UniversalDeepLinkingScripts**. This folder contains all tools required to correctly finish up the plugin configuration:

1. Copy the **UniversalDeepLinkingScripts** and your deliverable (**you\_game.app**) to your Mac or VM.
2. Open a Terminal and navigate to the copied items location.
3. Run the setup.sh script:
4. `./setup.sh`

If everything goes as expected you should see something similar to the following image:



```
UniversalDeepLinkingScripts — -bash — 80x24
Last login: Mon Jul 30 16:01:38 on ttys003
[Diogos-Mac:UniversalDeepLinkingScripts diogocardoso$ ls
optool          setup.sh
[Diogos-Mac:UniversalDeepLinkingScripts diogocardoso$
[Diogos-Mac:UniversalDeepLinkingScripts diogocardoso$ ./setup.sh
Found FAT Header
Found thin header...
Found thin header...
Inserting a LC_LOAD_DYLIB command for architecture: x86
Successfully inserted a LC_LOAD_DYLIB command for x86
Inserting a LC_LOAD_DYLIB command for architecture: x86_64
Successfully inserted a LC_LOAD_DYLIB command for x86_64
Writing executable to ../macapp.app/Contents/MacOS/macapp...
[Diogos-Mac:UniversalDeepLinkingScripts diogocardoso$
```

For more information on why you need to do this extra step check our [Inside the Plugin Section](#).

## Linux and Windows

On Windows and Linux standalone builds, the Deep Linking is only configured when the game runs the first time, so even if the player has the game installed on his machine, if he didn't play at least one time, the Deep Linking connection won't activate the game.

Windows and Linux builds also only allow players to start your game via a deep link if the player clicks on a deep link after your game is already open the plugin won't react to this new activation.

For further details on why this happens, you can check our [Inside the Plugin Section](#).

## Notes

- If you don't own a mac you can always create a [Mac VM](#)



# Domain Association

The documentation below is just a quick setup guide so that you can quickly integrate the Domain Association capability of the ImaginationOverflow Universal Deep Linking plugin if you wish to know more about this feature you can consult the official documentation:

- [Universal Links - iOS](#)
- [App Links - Android](#)
- [Uri Handling - Windows UWP](#)

**Note** that the Universal Deep Linking plugin **already handles all registration and configuration** to support Domain Association, you won't need to change any configurations on your manifest or configuration files on your Unity project.

The plugin **does not**, however, create or configure the required files that you need to host on your website, nor the dev account configurations required to enable this capability on some platforms.

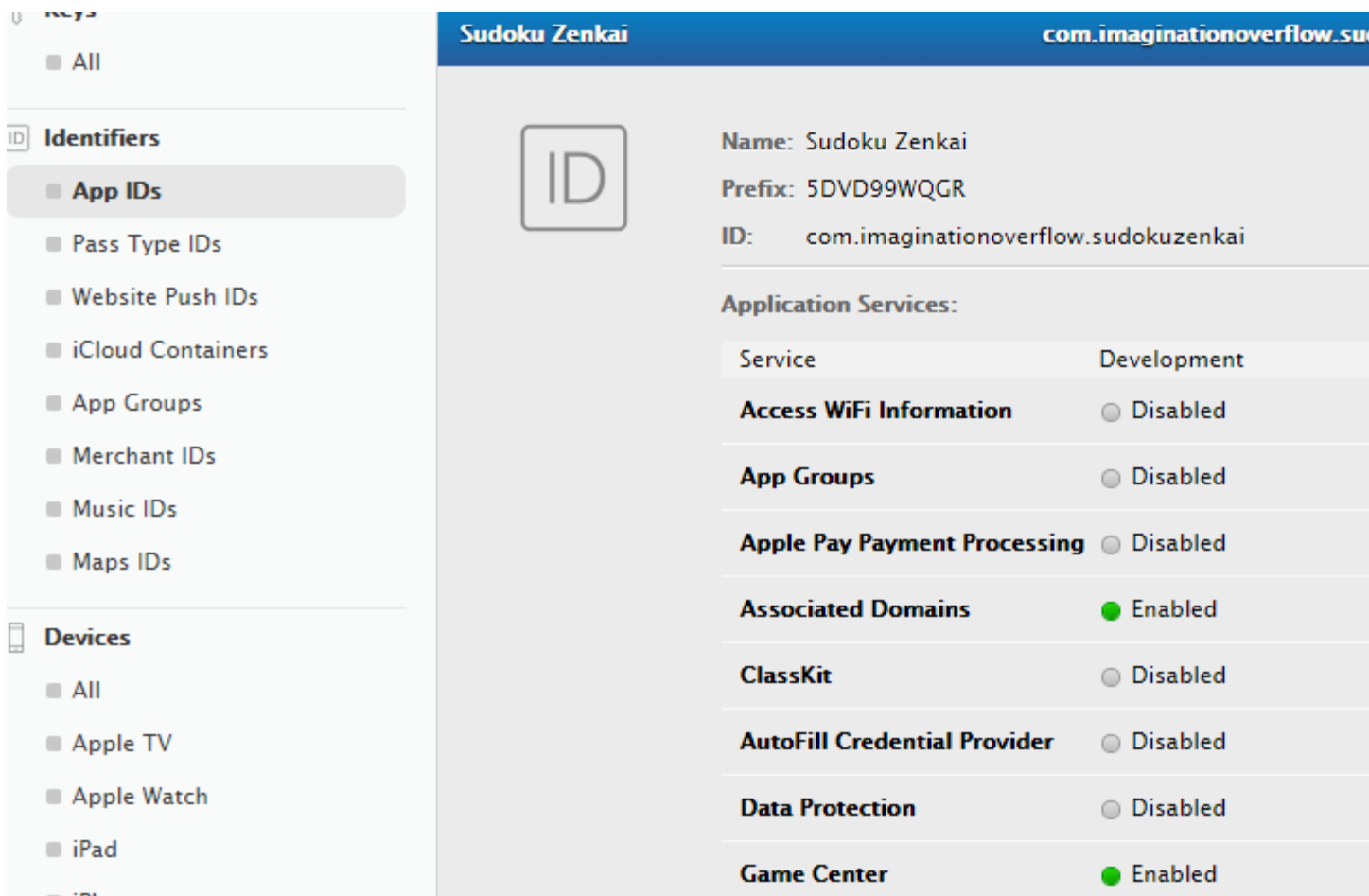
The following configuration steps are only required if you wish to use the Domain Association feature, if you are only using Deep Linking, you can skip these steps.

## iOS - Universal Links

The first thing you will need to do is activate the **Associated Domains** service on your app id:

1. Login on your Apple developer account in <https://developer.apple.com/>
2. In Identifiers -> App ID's, create or edit the app you wish to add Domain Association.
3. Check the **Associated Domains** checkbox
4. Save the changes

After you added the Associated Domain service, your page should look like the following:



From the above image, set-aside the **Prefix** and the **ID**. In Sudoku Zenkai case the Prefix is *5DVD99WQGR* and the ID *com.imaginationoverflow.com.sudokuzenkai*.

Now you need to create a json file named **apple-app-site-association** (no extension) with the following contents:

```
{
  "applinks": {
    "apps": [],
    "details": [
      {
        "appID": "[PREFIX].[ID]",
        "paths": [ "*" ]
      }
    ]
  }
}
```

With the file created you need to upload it to your website. The file should be accessible on either, the root or the **.well-known** directory:

[https://<your\\_domain>/well-known/apple-app-site-association](https://<your_domain>/well-known/apple-app-site-association)

or

`https://<your_domain>/apple-app-site-association`

You can check Sudoku Zenkai and Loyca (other of our projects) association files on:<http://sudokuzenkai.imaginationoverflow.com/apple-app-site-association>

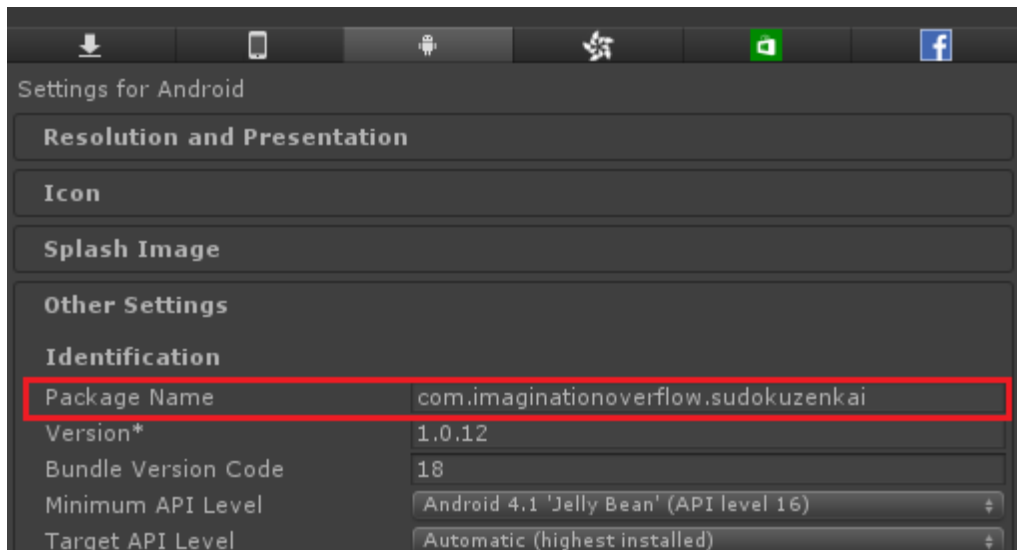
<http://loyca.imaginationoverflow.com/.well-known/apple-app-site-association>

With the above configuration, you are telling the iOS that all links to your website should be opened in your game, you may wish to configure only a few paths, to do that you can check the [official documentation](#).

Finally, your site must be using **HTTPS**, if you don't have it this feature won't work!

## Android - App Links

The first thing you need to configure the Domain Association is a **package name**, to configure that on your Unity Project click on **Edit -> Project Settings -> Player**. In the Android options you will find the **Package Name** setting:



For more info on how to choose a package name checkout [Google Documentation](#).

The second thing you need to have is a keystore, you need to sign your game before submitting it to the store, you can get more info on [Google documentation](#) about this topic.

After creating your keystore, you need to get **SHA-256 certificate fingerprint**, to do that just run the following command on your terminal (note the keytool comes with the Java SDK) :

```
keytool -list -v -keystore mystorekeystore.keystore
```

Running this command should yield something similar to the following image:

```
Your keystore contains 1 entry

Alias name: sudokuzenkai
Creation date: Mar 4, 2018
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: L=Lisbon, O=ImaginationOverflow
Issuer: L=Lisbon, O=ImaginationOverflow
Serial number: 25c367c4
Valid from: Sun Mar 04 14:54:39 GMT 2018 until: Mon Feb 20 14:54:39 GMT
Certificate fingerprints:
    MD5: 6D:6A:06:48:C4:3B:E5:0F:F3:6E:76:7C:EC:CA:B2:80
    SHA1: 50:14:E5:13:EA:A5:82:F0:FE:1E:41:17:1A:A3:C9:A3:C8:F3:27:
    SHA256: 2B:C9:10:E2:75:C8:EA:5D:8B:1A:03:7E:08:03:51:81:74:43:6
Signature algorithm name: SHA1WITHRSA
Subject Public Key Algorithm: 1024-bit RSA key
Version: 3
```

With the package name and SHA-256 certificate fingerprint, you can finally create the [digital asset link](#) file, the filename should be **assetlinks.json**

```
[{
  "relation": ["delegate_permission/common.handle_all_urls"],
  "target": {
    "namespace": "android_app",
    "package_name": "[MY_APP_PACKAGE_NAME]",
    "sha256_cert_fingerprints": ["[SHA256_FINGERPRINT_FROM_KEYTOOL]"]
  }
}]
```

Finally you need to host the file on your website, it needs to be available on the **root** or **.well-known** directory.

```
https://<your_domain>/.well-known/assetlinks.json
```

or

```
https://<your_domain>/assetlinks.json
```

You can check Sudoku Zenkai and Loyca digital asset link on the following links:

<https://sudokuzenkai.imaginationoverflow.com/assetlinks.json>

<https://loyca.imaginationoverflow.com/.well-known/assetlinks.json>

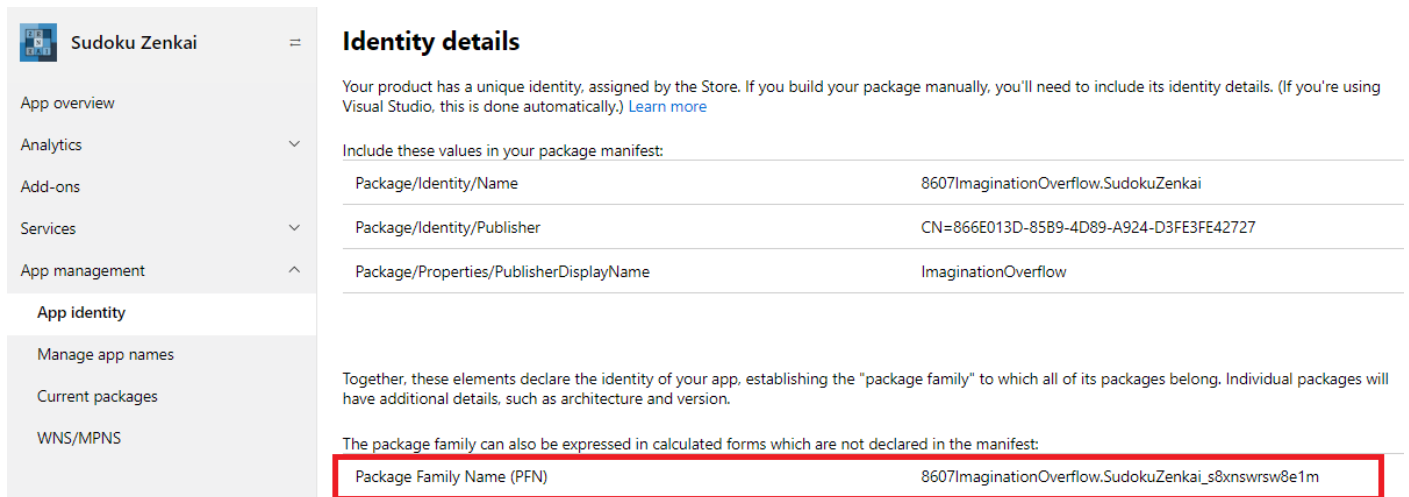
These files have optional configurations that you can use, for more info about these configurations check the [official documentation](#).

Note: You can get the production SHA-256 certificate fingerprint on your Google Play Console account, just navigate to Release Management -> App signing and

## Windows UWP - Uri Handling

To enable Domain Association on UWP you need to reserve a game name on the Windows Dev Center dashboard, to do that you can follow the [official documentation](#).

After you reserve your game name, you will need to go into the game settings under the Windows Dev Center dashboard, **App Management** -> **App identity**:



The screenshot shows the 'App identity' settings for 'Sudoku Zenkai'. The left sidebar lists navigation options: App overview, Analytics, Add-ons, Services, App management, App identity (selected), Manage app names, Current packages, and WNS/MPNS. The main content area is titled 'Identity details' and contains the following information:

Your product has a unique identity, assigned by the Store. If you build your package manually, you'll need to include its identity details. (If you're using Visual Studio, this is done automatically.) [Learn more](#)

Include these values in your package manifest:

Package/Identity/Name	8607ImaginationOverflow.SudokuZenkai
Package/Identity/Publisher	CN=866E013D-85B9-4D89-A924-D3FE3FE42727
Package/Properties/PublisherDisplayName	ImaginationOverflow

Together, these elements declare the identity of your app, establishing the "package family" to which all of its packages belong. Individual packages will have additional details, such as architecture and version.

The package family can also be expressed in calculated forms which are not declared in the manifest:

Package Family Name (PFN)	8607ImaginationOverflow.SudokuZenkai_s8xnsrsw8e1m
---------------------------	---

From there you can check your game **Package Family Name**, just as seen in the picture above.

With it, you need to create a json file named **windows-app-web-link** (no extension) and upload it to your website, under the **root** or **.well-know** directory.

```
[{
  "packageFamilyName": "[YOUR_PACKAGE_FAMILY_NAME]",
  "paths": [ "*" ],
  "excludePaths" : [ ]
}]
```

For some examples check out Sudoku Zenkai and Loyca **windows-app-web-link** files:

<https://sudokuzenkai.imaginationoverflow.com/windows-app-web-link>

<https://loyca.imaginationoverflow.com/windows-app-web-link>

You can set up other configurations on windows-app-web-link for more info check the [official documentation](#).

Finally, it's mandatory that your website supports **HTTPS**, because the windows will make an HTTPS request to try to retrieve the link.

## Notes

- The domain association feature only works for iOS, Android, and Windows (UWP).
- At ImaginationOverflow we are using [Cloud Flare](#) free tier service to enable **HTTPS** on all our domains.

# Steam Integration

Regular Standalone Deep Linking opens up your game when someone clicks on a custom defined URI. But if your game is on Steam you are probably using its SDK or DRM capabilities to further enhance your players' experience.

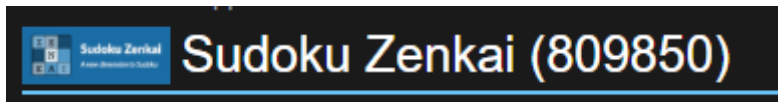
If you open a Steam game from its installation folder, since it wasn't open via steam it probably won't be able to communicate with valve software in order to report achievements, stats, etc, or it won't simply launch due to the DRM feature.

So using the regular Deep Linking mechanism where we associate a custom URI to a game executable wouldn't work since the game would possibly lose functionality or simply wouldn't start.

To avoid this issue, instead of registering your game to your custom URI, we configure the target systems to open steam instead of your game, but we parameterize Steam to open your game as soon as it's initialized.

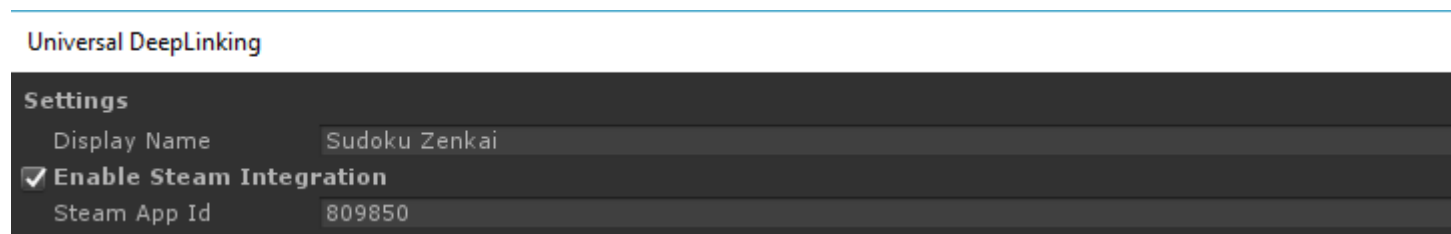
## Configuring Steam Integration

To enable the Steam capability of ImaginationOverflow Deep Linking Plugin you first need to have a valid app id, for that you need to get access to [Steam Direct](#). After that valve will attribute to your game an ID.



As an example, Sudoku Zenkai id is **809850**.

Then you need to configure the plugin with this id:



The final step of the configuration to enable the Steam integration is setting the *IsSteamBuildProperty* under **DeepLinkManager**:

```
ImaginationOverflow.UniversalDeepLinking.DeepLinkManager.Instance.IsSteamBuild = true;
```

If your game is on Steam there is a good chance that you offer DRM-free versions of it on other stores, so don't forget to turn off this option when making DRM-free standalone builds, a good way to do this automatically is using [compilation flags](#).

```
#if STEAM_BUILD
ImaginationOverflow.UniversalDeepLinking.DeepLinkManager.Instance.IsSteamBuild = true;
#else
ImaginationOverflow.UniversalDeepLinking.DeepLinkManager.Instance.IsSteamBuild = false;
#endif
```

## Mac and Steam Integration

Mac Deep Linking works a little different from the Windows and Linux, you have to take special care when integrating the plugin with Mac and Steam. On Linux and Windows, the plugin handles the registration of the Deep Link when the game is first open. This enables the plugin to register whatever it wants on these platforms. On Mac, the Deep Link feature is fully controlled by the OS and the plugin can't change the default configuration, which opens your game directly.

The OS itself doesn't know that is running a Steam game, so when the user clicks on a custom URI, the OS opens the game executable. This means that you need to explicitly delay the enforcement of the DRM until you know if the game was activated via Deep Linking or not.

The plugin is configured in a way that when it's running a Steam game on MacOS **it will always** fire the LinkActivated event even if the game wasn't activated via Deep Link. This way is possible to store the Deep Link before enforcing any DRM in the game.

### Solution Suggestion

Below is the code that we use on Sudoku Zenkai to handle this tricky issue, for Steam integration we are using [Steamworks.NET](#).

The user flow for this situation is the following:

1. User clicks on a Deep Link.
2. The game opens
3. The game enforces DRM and launches the game on Steam.
4. Steam launches (if not already running).
5. Steam launches the game.

To ensure that we process the Deep Link that the user initially clicked we need to save it before enforcing the DRM, in the code below we do exactly that:

1. The game is activated via Deep Link.



2. The LinkActivated event is triggered.
3. Save the link information (since it's impossible to have the game launch on steam via Deep Link)
4. Check if the game has steam access, if not restart.
5. If we are already on steam, load the previously saved Deep Link
6. Resume Deep Link activation.

This behavior is only possible because **the plugin always triggers** the LinkActivated event (on Steam Mac builds) regardless if it was activated via a Deep Link or not.

```

    public void RegisterForActivation()
    {
#ifdef UNITY_STANDALONE_OSX && STEAM_BUILD
        DeepLinkManager.Instance.LinkActivated += SteamOsxActivation;
#else
        DeepLinkManager.Instance.LinkActivated +=
Instance_LinkActivated;
#endif
    }

    private void SteamOsxActivation(LinkActivation s)
    {
        //
        // On Steam OSX builds the plugin triggers the LinkActivated
with the
        // deep link data or with a null Uri if it wasn't activated
        // via Deep Linking
        //
        if (string.IsNullOrEmpty(s.Uri) == false)
            YourGameStorage.SaveDeepLinkActivation(s);

        //
        // Enforce DRM
        //
        if (Steamworks.SteamAPI.RestartAppIfNecessary(new
Steamworks.AppId_t([YourAppId])))
        {
            Application.Quit();
            return;
        }

        //
        // We are already running on Steam, so load any saved deep
linking
        // activations
        //
        s = YourGameStorage.LoadDeepLinkActivation();

        if (s == null)
            return;

```

```
        //
        // Clear the activation ensuring that the game won't be
activated again
        // with the same uri this depends on your storage
infrastructure.
        //
        YourGameStorage.ClearDeepLinkActivation();

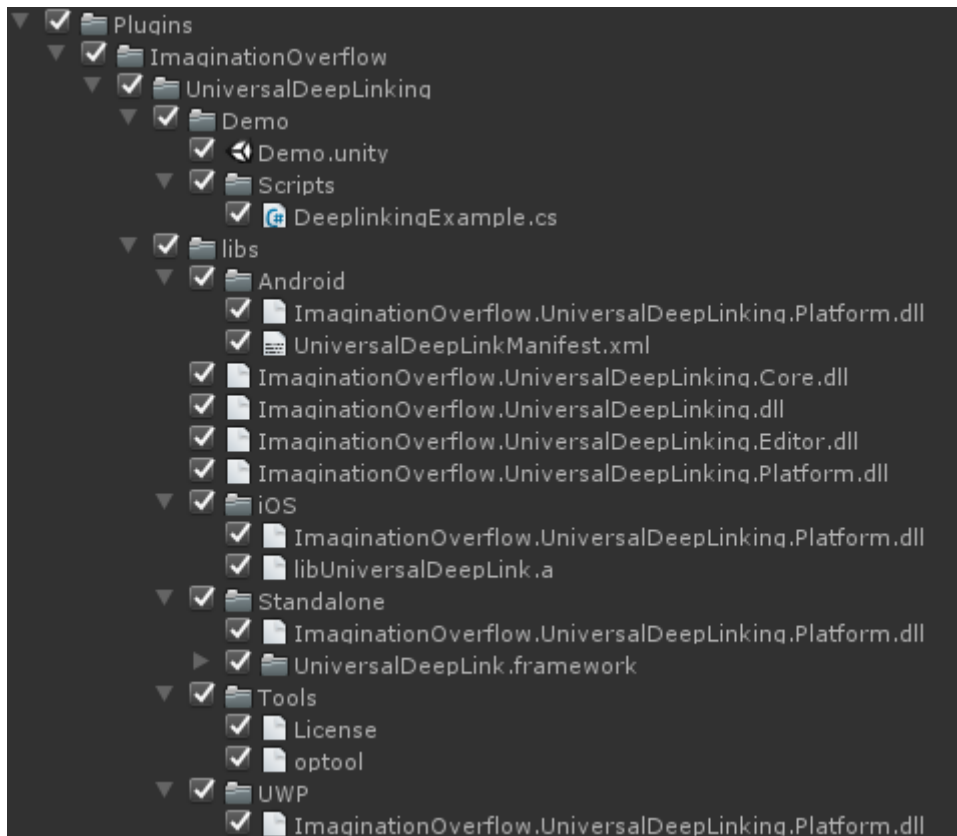
        //
        // Process deep link activation
        //
        Instance_LinkActivated(s);
    }
```

This behavior only happens when the game is running on Mac and you set the Steam flag to true, on the other combinations the plugin only fires the LinkActivated event in case of an actual activation.

# Inside the Plugin

On this section, we talk a little on how the plugin works on each platform and explain why it has some caveats on some platforms.

## Overview



The plugin content can be found inside the folder **Plugins/ImaginationOverflow/UniversalDeepLinking**, making easy to install and update when required.

We included a simple **Demo** scene that allows you to test and play with the plugin before integrating it in your game.

Inside the **libs** folder os all the required dlls in order for the plugin to work:

- *ImaginationOverflow.UniversalDeepLinking.dll* - Contains the public API of the plugin
- *.Core.dll* - Contains the core elements of the plugin
- *.Editor.dll* - Contains the windows and user interface of the plugin

- *.Platform.dll* - Contains the specific implementation for each platform, Android, iOS, UWP and Standalone.

The **Tools** folder contains external tools that the plugin requires in order to work correctly, right now, the only external tool we are using is [optool](#), required for MacOS builds.

The plugin configurations are saved under **Resources/ImaginationOverflow/UniversalDeepLink**, facilitating the use of source control systems on development. This file is also required on some build targets at runtime, reason why it's directly under the Resources folder.

## Runtime

At runtime the plugin creates a single **GameObject** and adds it to your current scene, this GameObject purpose is to ensure that all activation callbacks are called inside Unity main thread as well as propagate the **pause** events of the game to the plugin. The latter enables the plugin to refresh the activation data on some mobile platforms.

## Android

The plugin handles all manifest registrations when you make a build, the plugin **doesn't** override the default activity enabling it to work with the most used plugins in the Asset Store. The Deep Linking and Domain Association activation is checked everytime the game is opened or resumed.

## iOS

The iOS integration uses a static library **libUniversalDeepLink.a** in order to receive information about the app activation. Just like on Android the plugin handles all registrations on **info.plist** and **entitlements** files. The library included in the plugin notifies it everytime the game was opened via deep link or domain association activation.

## Windows UWP (Windows Store Games)

Just like the previous mobile platforms, the plugin automatically configures the **Package.appxmanifest** file with your configuration.

In order for the plugin to work it edits the OnActivated event under *App.xaml.cs*, *App.xaml.cpp* or *App.cpp* depending on what **Build Type** (Xaml or D3D) and **Scripting Backend** (.NET or IL2CPP) you configure. The plugin should also work on **Xbox UWP** games but it was impossible to test on an actual console in order to get confirmation.

## Linux and Windows

Linux the deep link registration is done the **first time the player opens the game**. To accomplish this the plugin creates a [Desktop File](#) on the player machine, enabling the operating system to set up the game as a target of a custom protocol.

Windows, the game writes in the registry the information necessary to enable the OS to open the game every time the player clicks on a configured deep link URI.

The protocol registration is also done everytime the [Application.version](#) is changed, enabling you to change the configuration with an update.

If the game build is for Steam, the plugin configures Steam to be the target of your custom URI instead of the game (this is done to work around DRM) but configures Steam to launch your game with the Uri that opened steam.

Linux and Windows builds (Steam or Standalone) can't be activated via a Deep Link **after the game is already running**. This is because the link activation information is passed via argument on the **main** function, making it impossible (right now at least) to get information of the activation link after the game is already running.

## Mac

All manifest registrations are handled by the plugin, the deep linking activation is deferred from our library into the game as it happens, so MacOs builds won't have the caveats that Windows and Linux have.

In order to support Deep Linking we had to make a **library** (*UniversalDeepLink.framework*) that would intercept the activation events of the application itself since Unity doesn't allow the plugin to automatically link a library on the build process it must be done after the build. To make that possible the plugin includes the tool **optool**.

optool allow us to inject the library into the game and collect all the activation events. If you build your game on MacOs the plugin **will automatically** call optool and inject the library. If you make the build on any other OS you will need to make an extra step, just has explained on the [Getting Started Section](#).

This requirement exists because optool was made for MacOs and the team couldn't in useful time port it to Windows.

For more info about how the library injection works check [here](#).

# Showcase

Did you integrate the Universal Plugin in your game? Let us know and we will add it to the site and this list.

## Sudoku Zenkai



|| [Android](#) || [iOS](#) || [Microsoft Store](#) || [Steam](#) ||

|| [Website](#) || [Twitter](#) || [Facebook](#) ||

## Super Bunny Laser Spikes



|| [Android](#) || [iOS](#) || [Microsoft Store](#) ||

|| [Website](#) || [Twitter](#) || [Facebook](#) ||

# About

ImaginationOverflow is a polyvalent group of people that come together in order to make software. They specialize in Apps made with Xamarin and Unity games. As indies, they published dozen of apps and games.

## Authors

**Diogo Cardoso** - [Twitter](#) --- [LinkedIn](#)

**Ana Correia** - [LinkedIn](#)

## Links

[Website](#)

[Twitter](#)

[Facebook](#)

[Steam Page](#)

[Play Store](#)

[App Store](#)

[Microsoft Store](#)

[Discord Server](#)

[Contact Email](#)

[Blog](#)

## Acknowledgements

- The portuguese gamedev community for suggestions and support.
- [Geri Borbás](#) for his awesome articles regarding unity plugins and library injection on MacOS
- [Alex Zielenski](#) for creating optool and open it for the community
- Our Patrons, Eric T, Tom T, Michael, Abdullah T

