# MAS SDK Integration for Game Maker Studio 2.X

# SDK Version

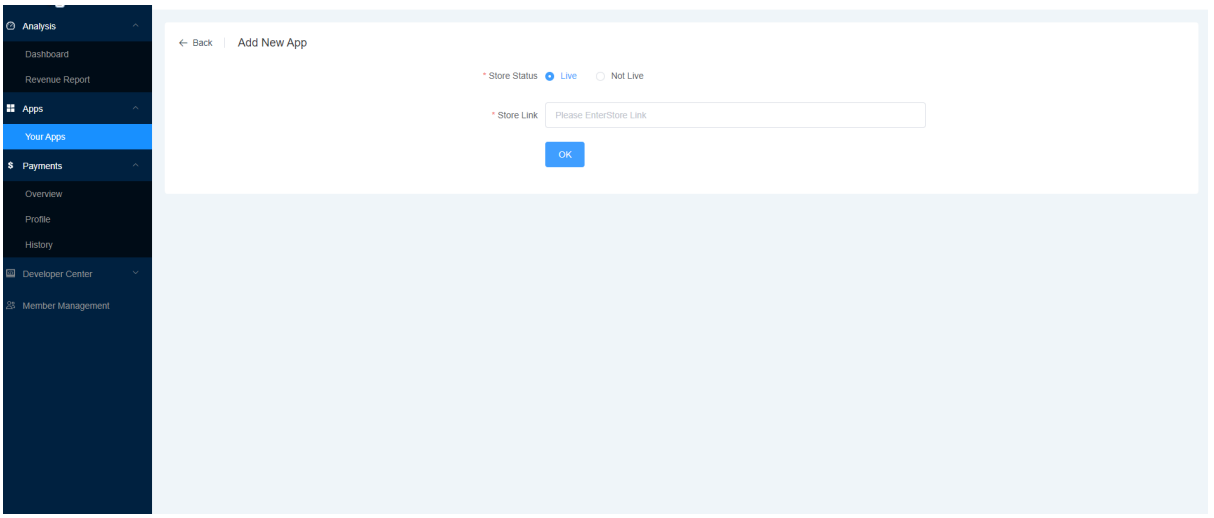
Current Version: 4.1.0
Date: 11/05/21

## Setting up

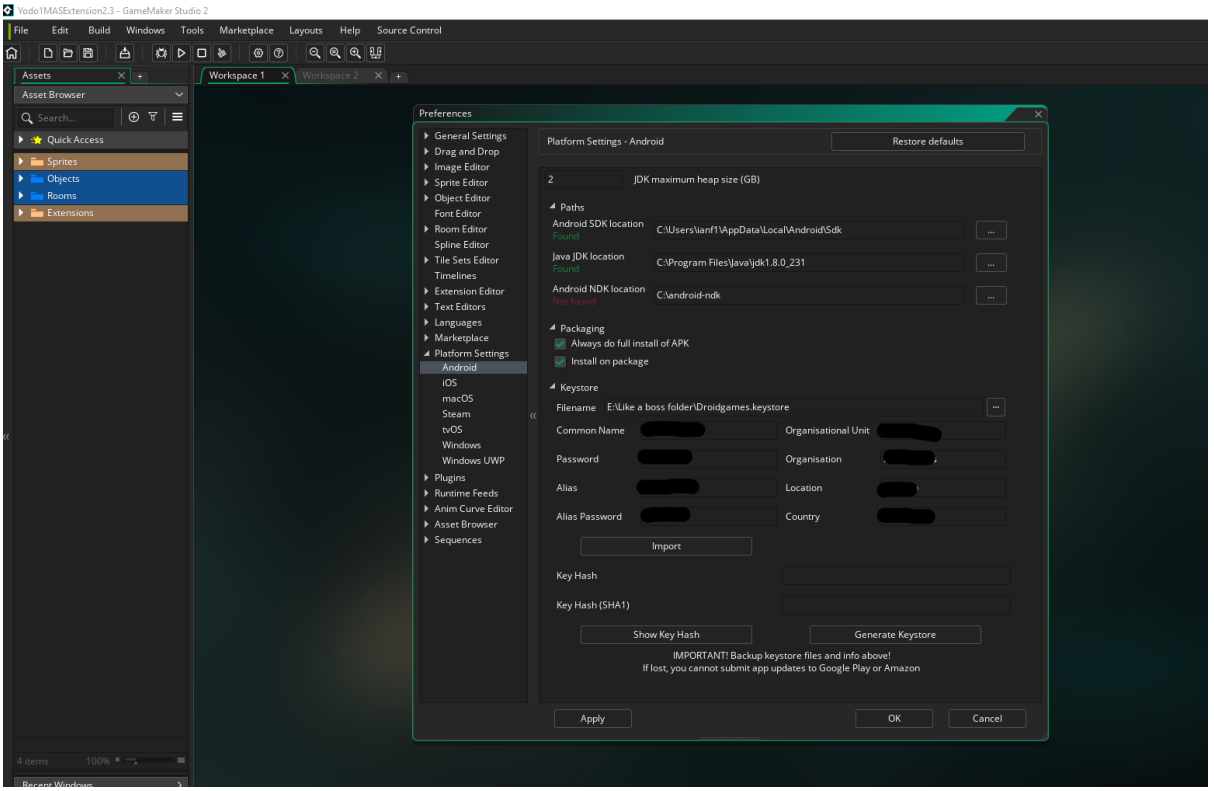
Then  implement the MAS SDK on your awesome game you need to create an account on the MAS dashboard so let's see how you can do it:

### Creating MAS account

- Got to [https://mas.yodo1.com/registration/register](https://mas.yodo1.com/registration/register)
- Create or login to the dashboard if you already have an account.
- One you create an account you can now start using the services on your Game so let's add your app on the dashboard.




- Navigate to until you reach the "Apps" section and press the "+ Add new App" Button
- And fill in the required information about the App
- There is a section about the **SHA1 key** so let's see how you can get that key from your game. Go to **File > Preferences > Platform Settings > Android** and scroll down until you see the **Show Key Hash** button and press it, now you will see your **SHA1 Key** copy and paste it.


Yodo1MASExtension2.3 - GameMaker Studio 2
File Edit Build Windows Tools Marketplace Layouts Help Source Control
Assets
Asset Browser
Search...
Quick Access
Sprites
Objects
Rooms
Extensions
Workspace 1
Workspace 2
Preferences
General Settings
Drag and Drop
Image Editor
Sprite Editor
Object Editor
Font Editor
Room Editor
Spline Editor
Tile Sets Editor
Timelines
Extension Editor
Text Editors
Languages
Marketplace
Platform Settings
Android
iOS
macOS
Steam
tvOS
Windows
Windows UWP
Plugins
Runtime Feeds
Anim Curve Editor
Asset Browser
Sequences
Platform Settings - Android
Restore defaults
2
JDK maximum heap size (GB)
Paths
Android SDK location
Found
C:\Users\ianf1\AppData\Local\Android\Sdk
Java JDK location
Found
C:\Program Files\Java\jdk1.8.0_231
Android NDK location
Not found
C:\android-ndk
Packaging
Always do full install of APK
Install on package
Keystore
Filename
E:\Like a boss folder\Droidgames.keystore
Common Name
Organisational Unit
Password
Organisation
Alias
Location
Alias Password
Country
Import
Key Hash
Key Hash (SHA1)
Show Key Hash
Generate Keystore
IMPORTANT! Backup keystore files and info above!
If lost, you cannot submit app updates to Google Play or Amazon
Apply
OK
Cancel
4 items
100%

# SDK Setting Up Android

The SDK is really easy to integrate, but first we need to have some considerations in mind before importing it.

> Delete all the previous ad SDKs if you already have one on your game before implementing the MAS SDK

After those considerations let's rock it!

## Modify root files of GMS 2.3.X

In order to make the extension compile on the GameMaker Studio we need to add a little parameter to the AndroidManifest.xml from the root files of GMS. If you don't make this step you will get a compile error like this one:

```
FAILURE: Build failed with an exception.

* What went wrong:
Execution failed for task ':com.yodo1.MasSDK:processDebugManifest'.
> Manifest merger failed : Attribute application@allowBackup value=(false) from
AndroidManifest.xml:29:179-206
        is also present at [com.yodo1.mas.mediation:admob:4.0.3.0]
AndroidManifest.xml:15:9-35 value=(true).
        Suggestion: add 'tools:replace="android:allowBackup"' to <application> element at
AndroidManifest.xml:29:3-104:17 to override.
```

> **NOTE:**
> **You'll need to change the AndroidManifest.xml each runtime update of GMS**

To modify the root AndroidManifest.xml you will need to go to next path of your pc:

C:\ProgramData\GameMakerStudio2\Cache\runtimes\runtime-2.3.X.X\android\runner\ProjectFiles\src\main

You'll need to change the runtime version depending on yours, if you are not sure what your runtime version is, just copy that path before **\runtime-2.3.X.X** and then follow the path manually.

Then you will find the file called AndroidManifes.xml and open it with a text editor or source code editor like Atom. Scroll down until you see the application tag <application ...>.....
and under the application tag properties find the one called “**tools:replace="android:label"**”

```
<!-- application -->
<application android:name="${YYAndroidPackageName}.RunnerApplication" android:label="@string/app_name" android:icon="@drawable/icon" tools:replace="android:label"
  <activity android:name="${YYAndroidPackageName}.RunnerActivity" android:theme="@android:style/Theme.NoTitleBar.Fullscreen" android:label="@string/app_name"
            android:launchMode="singleTask" android:alwaysRetainTaskState="true"    android:configChanges="orientation|keyboardHidden|screenSize|screenLayout|smalle
    <intent-filter>
      <action android:name="android.intent.action.MAIN"/>
      <category android:name="android.intent.category.LAUNCHER"/>
      <category android:name="android.intent.category.LEANBACK_LAUNCHER"/>
      ${YYAndroidExtraCategories}
    </intent-filter>
  ${YYAndroidManifestActivityInject}
  </activity>
```

Inside the double quotes of that property add the next one separate with coma: **android:allowBackup** so the final look of that property could be like this: **tools:replace="android:label, android:allowBackup"**

```
application -->
lication android:name="${YYAndroidPackageName}.RunnerApplication" android:label="@string/app_name" android:icon="@drawable/icon" tools:replace="android:label, android:allowBackup"
ctivity android:name="${YYAndroidPackageName}.RunnerActivity" android:theme="@android:style/Theme.NoTitleBar.Fullscreen" android:label="@string/app_name"
       android:launchMode="singleTask" android:alwaysRetainTaskState="true"    android:configChanges="orientation|keyboardHidden|screenSize|screenLayout|smallestScreenSize" tools:r
<intent-filter>
  <action android:name="android.intent.action.MAIN"/>
  <category android:name="android.intent.category.LAUNCHER"/>
  <category android:name="android.intent.category.LEANBACK_LAUNCHER"/>
  ${YYAndroidExtraCategories}
</intent-filter>
YYAndroidManifestActivityInject}
```

## Adding the SDK from the Marketplace into the project

Let's start by importing the SDK on your game

- Go to the Yoyo Marketplace and download the SDK and import it. We recommend to do it inside the Game Maker Studio by right click on the **Extension** folder and then click on **Add Existing from my Library**

- Then a screen will appear asking for what resources do you want to import
  There are some resources that are "**Examples**" but you can import them and also use them and will work perfectly fine!

- The main resource to import is the **Yodo1MASExtension** and the object **obj_yodo1core**. This object has some premade codes that will help you to implement the SDK faster, but you can create it manually and follow the guide below.

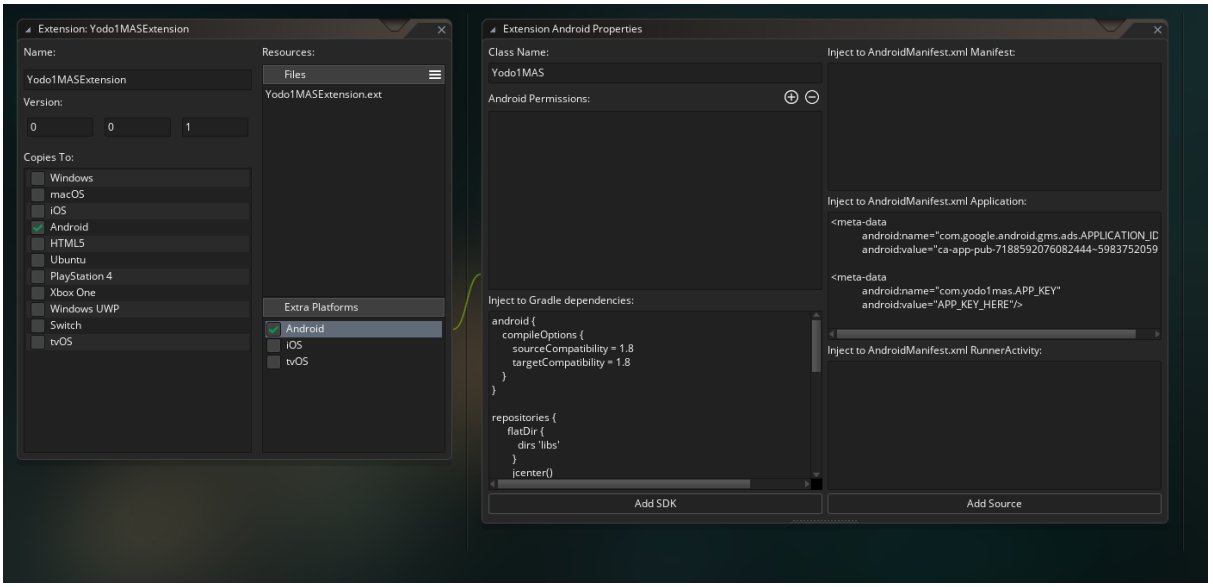
- Then let's double click the extension and go to Extra Platforms, Click Android and you will see another window with some info. Here we need to add the **AppKey** (this AppKey is on the details of your app on the MAS Dashboard) and the **Admob App ID**.

- Copy the Key from the MAS Dashboard and find the section **"Inject to AndroidManifest.xml Application"** and you will see a code like this:

```
<meta-data
        android:name="com.google.android.gms.ads.APPLICATION_ID"
        android:value="ADMOD_APP_ID"/>

<meta-data
        android:name="com.yodo1mas.APP_KEY"
        android:value="APP_KEY_HERE"/>
```

Just replace the text **APP_KEY_HERE** with your **AppKey** and the **ADMOD_APP_ID** with the **Admob App ID**




And that's it from the Setting Up the extension now let's see **how to Interstitial and Rewarded Videos!**

# Using the Extension

## Initializing the extension

In order to use the extension we need to create a **persistent object** and we can name it **obj_yodo1core** and place it in your **first room** of your game.

Before initializing the extension we need to **comply with some agreements** to avoid **revenue loss** and Google Play/AppStore **bans**. The next functions need to placed on the **Game Start Event** of the **obj_yodo1core:**

The functions **Yodo1MAS_COPPA**, **Yodo1MAS_CCPA** and **Yodo1MAS_GDPR** are the functions you need to use depending on the agreement you need to comply.

**NOTE:**
You can use the macro **yodo1mas_agree** or **yodo1mas_disagree** to accept or decline the agreements depending on the case.

- For **COPPA** you can use the next code:

```
Yodo1MAS_COPPA(yodo1mas_agree or yodo1mas_disagree);
```

- For **CCPA** you can use the next code:

```
Yodo1MAS_CCPA(yodo1mas_agree or yodo1mas_disagree);
```

- For **GDPR** you can use the next code:

```
Yodo1MAS_GDPR(yodo1mas_agree or yodo1mas_disagree);
```

After adding the needed agreement functions add the next function to initialize the sdk below the agreements ones on the **Game Start Event** in the **obj_yodo1core:**

```
Yodo1MAS_Initialize();
```

## Adding Interstitials

The interstitial is the easiest one to implement and use with MAS SDK Extension! so let's have a look on how to implement it.

- Whenever you want to display an interstitial ad just add this function on **any object**:

```
Yodo1MAS_ShowInterstitialAd();
```

Additionally you could add the async callbacks on your **obj_yodo1core**, on the **Asynchronous Event > Social** you can add the next code:

**NOTE:**
In case you already implemented the Asynchronous logic you can only add the **cases** inside the **switch** that you actually need.

```
//getting the id of the callback
var id_async = async_load[? "id"];

///checking if was from Yodo1 MAS
if(id_async == Yodo1MAS_ASyncEvent){

	switch(async_load[? "type"]){
		#region Interstitial Cases
			case yodo1mas_interstitialclosed:
				show_debug_message("YODO1 Closed");
				break;


			case yodo1mas_interstitialopened:
				show_debug_message("YODO1 Opened");
				break;

			case yodo1mas_interstitialerror:
				show_debug_message("YODO1 Error");
				break;
		#endregion

	}
}
```

## Adding Rewarded Videos

This popular type of ad is really easy to implement too but will required a little bit more code The extension also triggers an **Async Event** to get the response of the ad and check if the user ends to watch the ad and therefore give the reward.

Ok let's keep going:

- Add this code on the object that you want to show the interstitial:

```
Yodo1MAS_ShowRewardAd();
```

- Add this code on the **Async - Social Event**

**NOTE:**
In case you already implemented the Asynchronous logic you can only add the **cases inside the #region Rewarded Cases** inside the **switch**.

```
//getting the id of the callback
var id_async = async_load[? "id"];

///checking if was from Yodo1 MAS
if(id_async == Yodo1MAS_ASyncEvent){

	///now, what was it?
	switch(async_load[? "type"]){

		#region Rewarded Cases
			case yodo1mas_videoshow:
				show_debug_message("YODO1: Reward  Opened");
				break;

			case yodo1mas_videoshowfailed:
				show_debug_message("YODO1: Reward  Failed");
				break;

			 case yodo1mas_videoclosed:
				//The extension returns 0 if the video was closed
				//The extension returns 1 if it was Finished.
				var isFinished = async_load[? "isFinished"];
				if(isFinished == 1){
				   ///Your awesome reward!
				   show_debug_message("YODO1: Reward  claimed");
				}
```

```
                        global.isBannerReady = false;
                        break;

                #endregion

        }
}
```

You can add the reward code below the comment ///Your awesome reward! and that 's it!

## Final steps

- Submit the SDK-integrated build to App Store/Play Store.
- Once the SDK-integrated build has gone live on the App Store/Play Store, let us know by clicking the release button on the Dashboard.