## Android

Al compilar aplicaciones para Android existen distintas alternativas de **empaquetamiento**. La clásica es compilar las aplicaciones a paquetes **.apk** (**Android Application Package**). Los paquetes .apk contienen todos los recursos de la aplicación, tales como los distintos assets, idiomas, textos, interfaces, etc. El problema principal de los .apk es que incluyen todos los recursos necesarios para que la aplicación pueda ejecutarse en distintas plataformas Android, como distintos smartphones, tablets, con su variedad de tamaños e idiomas. Esto implica que al momento de descargar la aplicación, esta será de un mayor tamaño que el necesario. Además, cuando una aplicación supera un cierto tamaño, por ejemplo si incluye una gran cantidad de assets como imagenes o videos, es necesario incluir estos assets en un paquete secundario del tipo **.obb** (**Opaque Binary Blob**).

En 2018 Google presentó los **.aab** (**Android App Bundle**), y desde Agosto del 2021 las aplicaciones que se suban a **Google Play** tendrán que estar empaquetadas en formato .aab. Además, las aplicaciones que superen el tamaño de 150MB deberán hacer uso de **Play Feature Delivery** o **Play Asset Delivery** (ambas plataformas para obtención de paquetes de Google Play). Una de las ventajas para el desarrollador que poseen los .aab consiste en que al momento de compilar el paquete, este automáticamente genera todas las versiones posibles para las distintas plataformas y las incluye en el .aab, a diferencia de los .apk, en los que era necesario compilar para distintas plataformas y luego incluirlas manualmente. Además, para el usuario final, al momento de descargar la aplicación de Google Play, la plataforma solo descarga aquellos recursos necesarios para su plataforma particular, reduciendo la utilización de espacio en memoria por la aplicación.

## Unity y los AssetBundle

Unity posee la capacidad de empaquetar conjuntos de assets en archivos de tipo **AssetBundle**.
Los AssetBundles tienen como función simplificar la obtención de assets desde una red (CDN).
Estos assets contenidos en AssetBundles pueden cargarse a la aplicación en tiempo de ejecución.
Por esta razón es común que se utilicen en aplicaciones con **Contenido Descargable** (**DLC**).

Play Asset Delivery hace uso del sistema de AssetBundles de Unity y lo potencia entregando funciones automatizadas de obtención de paquetes desde la red de Google Play mediante la API de Play Asset Delivery.

## Play Asset Delivery (PAD)

PAD es una alternativa entregada por Google para reemplazar los antiguos paquetes .obb de expansión, permitiendo aplicaciones que superen los 150MB de tamaño. PAD permite generar múltiples paquetes de assets secundarios y marcarlos para distintas modalidades de entrega, actualizaciones automáticas, compresión de assets, parches delta, etc. Otra ventaja es que con los antiguos .obb, era el desarrollador el encargado de suministrar el paquete de expansion .obb a los usuarios mediante una **Red de Distribución de Contenidos** (**CDN**) externa, mientras que con PAD, los paquetes son alojados en la red de Google Play, facilitando la obtención de estos paquetes. Los paquetes pueden marcarse con 3 modalidades de entrega diferentes: **install-time**, **fast-follow** y **on-demand**. Esto permite desarrollar una aplicación que posea una gran cantidad de contenidos, los que pueden ir entregándose por partes o capítulos, manteniendo el tamaño de la aplicación bajo.

**Creación de AssetBundles**

Para poder generar paquetes de tipo AssetBundle se deben seleccionar los Assets y marcarlos con una etiqueta que denota el AssetBundle al que pertenecerá. Luego, Unity nos entrega la función BuildPipeline.BuildAssetBundles(), con la que se pueden generar AssetBundles en un script. Para facilitar el proceso de creación de AssetBundles se agregará una opción en el Menú de Unity mediante un script, de esta forma solo será necesario marcar los Assets en Unity con la etiqueta de AssetBundle correspondiente y ejecutar esta nueva opción en el Menú de Unity. Esto creará todos los AssetBundles en la carpeta "Assets/AssetBundles" dentro del proyecto. Para esto se deben seguir los siguientes pasos:

1) Crear una carpeta llamada "Editor" dentro de la carpeta "Assets" del proyecto en Unity.
2) Dentro de esta carpeta, se debe crear el siguiente Script C# llamado "CreateAssetBundles":

```
---------------------------------------------------------------------------------------------------------------------
using UnityEditor;
using System.IO;

public class CreateAssetBundles
{
    [MenuItem("Assets/Build AssetBundles")]
    static void BuildAllAssetBundles()
    {
        string assetBundleDirectory = "Assets/AssetBundles";
        if(!Directory.Exists(assetBundleDirectory))
        {
            Directory.CreateDirectory(assetBundleDirectory);
        }

BuildPipeline.BuildAssetBundles(assetBundleDirectory,
BuildAssetBundleOptions.UncompressedAssetBundle |
BuildAssetBundleOptions.ForceRebuildAssetBundle,
BuildTarget.Android);
    }
}

---------------------------------------------------------------------------------------------------------------------
```

3) Desde el Project View en Unity deben marcarse los Assets con la etiqueta de AssetBundle al que se quiere agregar, para esto, se selecciona el Asset y se revisa el Inspector de Unity. En la parte inferior del Inspector hay un menú llamado AssetBundle, aquí se debe seleccionar el AssetBundle al que se desea agregar el Asset seleccionado. Este menú permite seleccionar de una lista de etiquetas de AssetBundles existentes o crear nuevas etiquetas.

4) Una vez se hayan marcado todos los Assets necesarios en sus AssetBundles respectivos, solo es necesario ir a: [Menú de Unity → Assets → Build AssetBundles] . Esto ejecutará el script de creación de AssetBundles y una vez terminado los AssetBundles se encontrarán en la carpeta "AssetBundles".

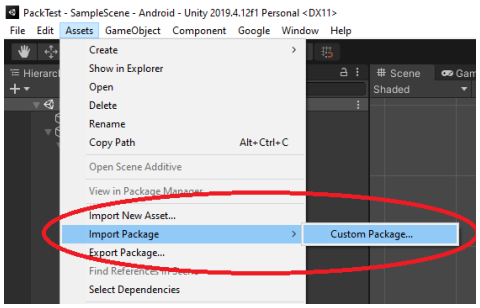
5) Por último, Unity entrega funciones para cargar estos AssetBundles de distintas formas, ya sea desde una dirección en memoria (usando AssetBundle.LoadFromFile()) o incluso descargándolo. En este caso sera PlayAssetDelivery el encargado de obtener los AssetBundles.

**Integración de PAD a Unity**

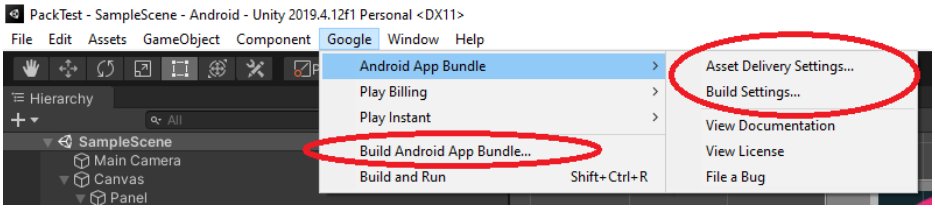
Para integrar PAD a Unity es necesario instalar la librería **Play Core**, esta hace de interfaz entre la aplicación y todas las funcionalidades disponibles de la **Google Play Store**. La librería se debe descargar del siguiente link:

**https://github.com/google/play-unity-plugins/releases**

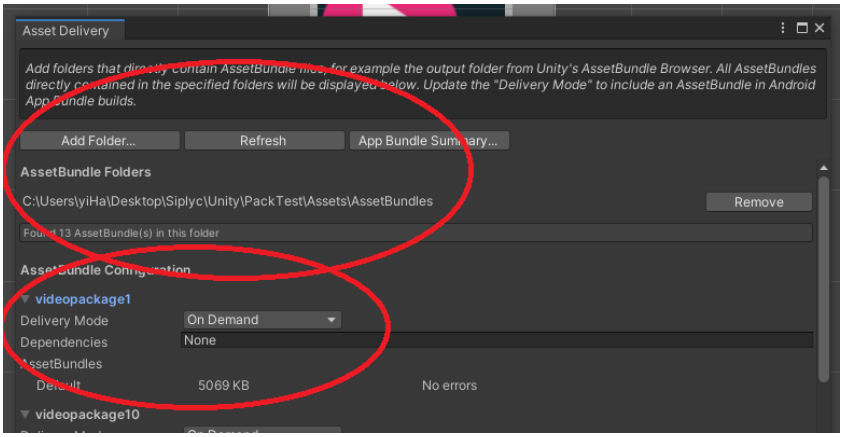
La librería viene en la forma de un archivo **.unitypackage**. Estos paquetes se cargan desde [Assets → Import Package → Custom Package].




Una vez instalada la librería, debiese de aparecer una nueva opción en el menú de Unity dedicada a las opciones de Google.

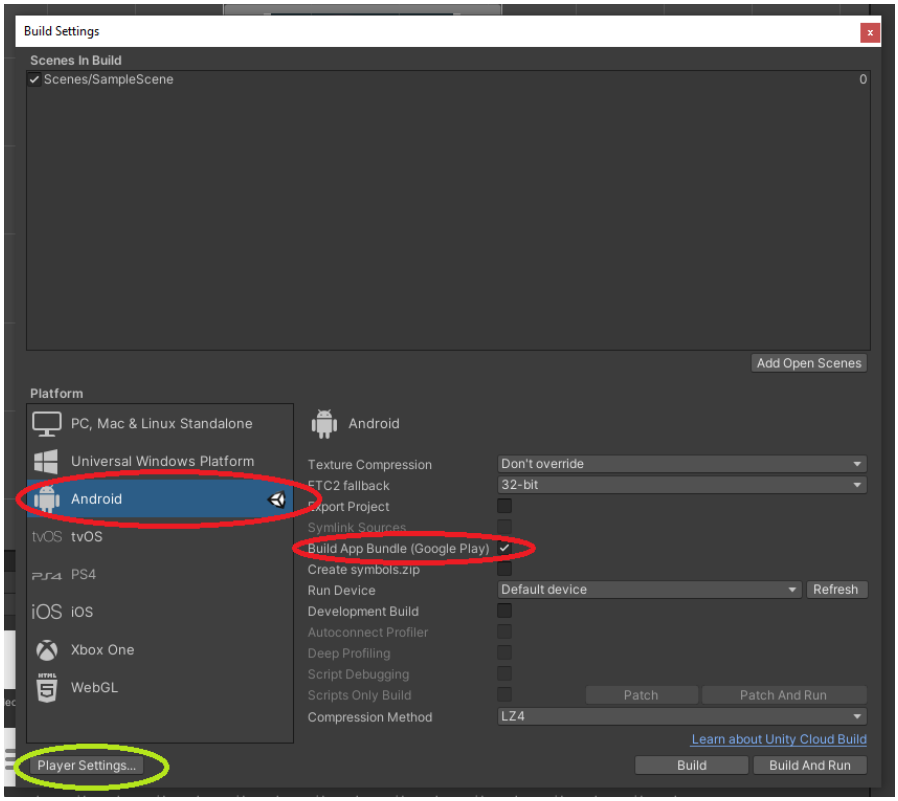



Por último, se debe ir a la opción de “Asset Delivery Settings”. Aquí debe especificarse la carpeta donde se encuentran los AssetBundles previamente creados. Luego, se debe seleccionar la modalidad de entrega para cada uno de los AssetBundles (Install-time, Fast-follow, On-demand).

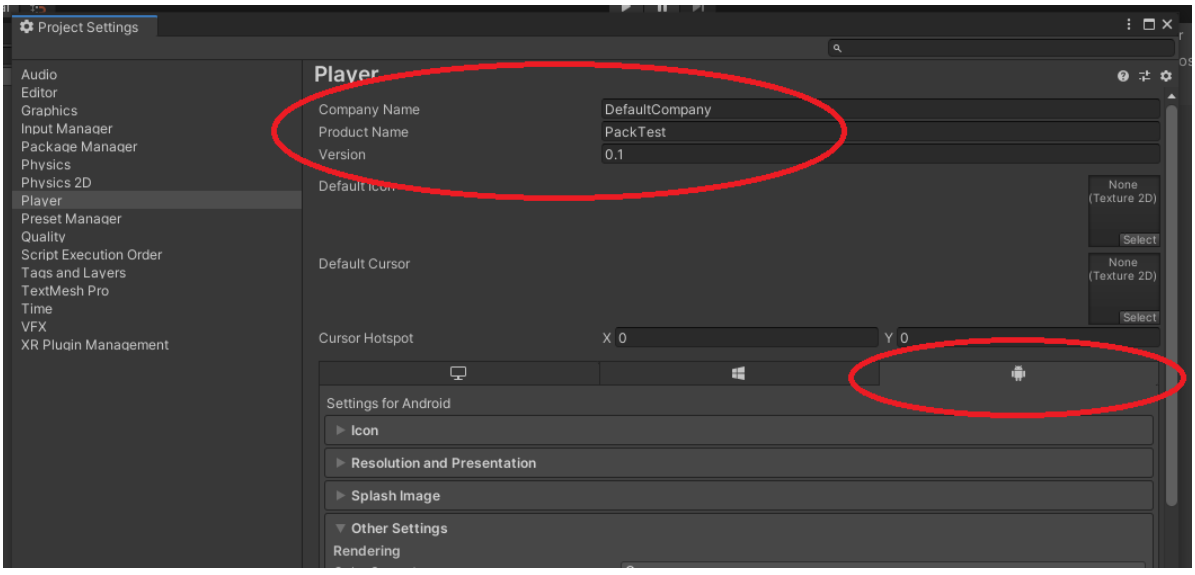


**Configurar Unity para Android**

1) Ir a: [Menu de Unity → File → Build Settings] y configurar el Target Platform a Android. Luego, se activa el check de “Build App Bundle (Google Play)”, así el paquete compilado por Unity será del tipo .aab y no un .apk.

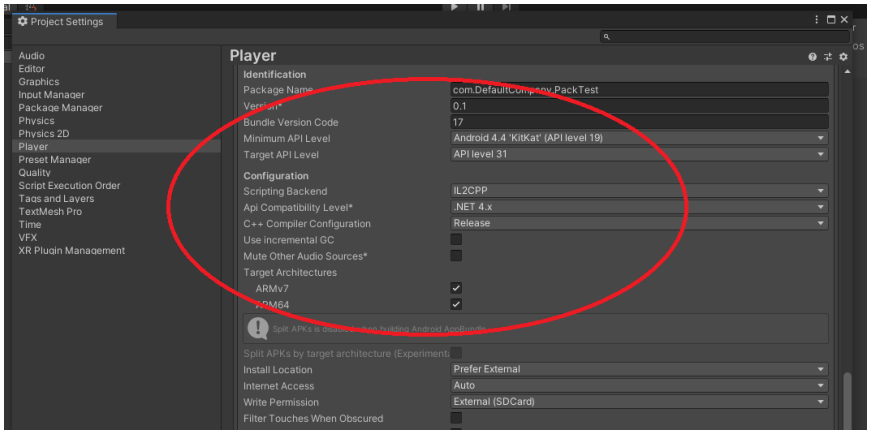



2) Apretar en “Player Settings”, esto abrirá la ventana de “Project Settings”. Aquí, se deberán configurar múltiples parámetros del proyecto, comenzar por nombre de la empresa, nombre de la aplicación y versión.

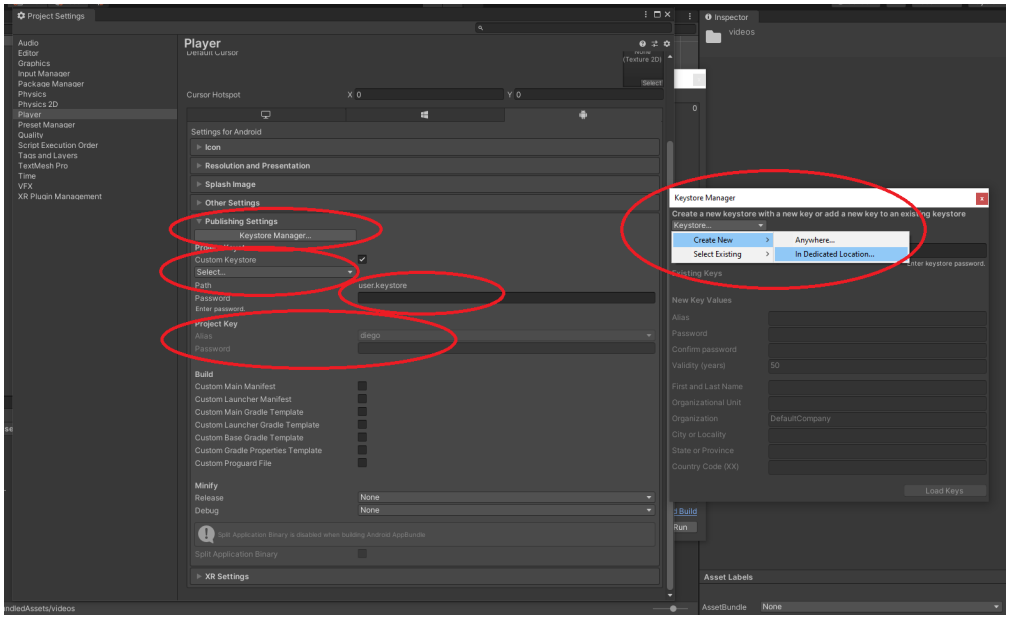


3) Es importante configurar los siguientes parámetros:

* Bundle Version Code (cada vez que se suba una nueva versión de la App a Google Play Console, deberá aumentarse este valor!).
* Minimum API Level (define la versión mínima de Android a la que se le permite ejecutar la aplicación).
* Target API Level (desde Agosto 2021 el Target como mínimo debe ser 30!).
* Scripting Backend (IL2CPP).
* API Compatibility Level (.NET 4.X).
* C++ Compiler Configuration (Release).




4) Por último, se debe ir a la sección de Publishing Settings y apretar en Keystore Manager. Esto abrirá la ventana que se ve en la derecha de la imagen. Esto nos permitirá crear una Key para la aplicación, que será una especie de licencia o identificador. Esta Key se guarda como un archivo y no se debe perder. Además, cada vez que se abra Unity, si se desea compilar la aplicación y subirla a Google Play Console, será necesario ingresar los datos de la Key en la parte inferior (ingresar la password, seleccionar el Alias de la Key e ingresar la Password de la Key). Esto lo exige Google Play.



**Configurar GooglePlayConsole**

- password / firma de aplicación

**Plugin AVCPRO y preparar videos para assetbundles y PAD (.bytes)**

// #SCRIPT PARA TRANSFORMAR .mp4 a .mp4.bytes!!!!

---

```
for file in *.mp4; do mv "$file" "${file/.mp4/.mp4.bytes}"; done
```

---

// #agregar en archivo de texto, ejecutar como sh desde cmd.

asdf

**Links de Interés**

https://docs.unity3d.com/Manual/AssetBundles-Workflow.html
https://developer.android.com/guide/playcore/asset-delivery/integrate-unity?language=plugin
https://docs.huihoo.com/unity/5.3/Documentation/en/Manual/AssetBundlesIntro.html
https://answers.unity.com/questions/1714181/av-pro-how-to-play-video-from-asset-bundle.html
https://learn.unity.com/tutorial/introduction-to-asset-bundles#6028be40edbc2a112d4f4fe5
https://stackoverflow.com/questions/45580717/playing-large-video-files-in-gear-vr-in-unity-project?noredirect=1#comment78162170_45580717
https://forums.oculusvr.com/t5/Unity-Development/Expansion-File-Bundled-Scene-Not-Loading-Video-Go/td-p/667827
https://answers.unity.com/questions/1521925/how-do-you-add-text-to-a-scroll-view-programatical.html