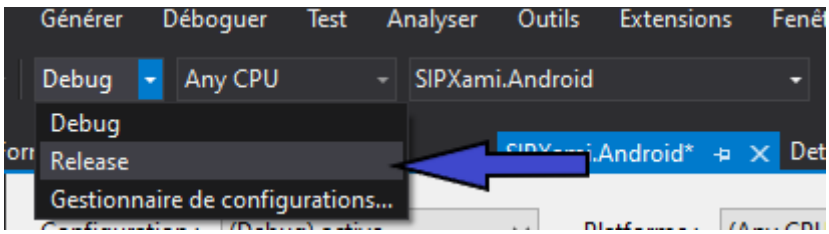
Déployer un projet Xamarin.Forms sur Android

Ce guide s'adresse aux développeurs ayant une application codée et testée, et qui souhaitent préparer un paquet application Android pour la distribution.

Par souci de simplicité, j'ai décidé de ne garder que les informations essentielles pour ce guide. Cependant, si vous voulez des détails sur des options que je ne traiterai pas, je vous conseille la documentation de Microsoft qui est très bien rédigée (en anglais), voir source à la fin.

Afin de ne pas avoir de confusion entre le mode **Debug** du projet et le mode **Release**, il est conseillé de passer son projet en mode Release avant de commencer ce guide.

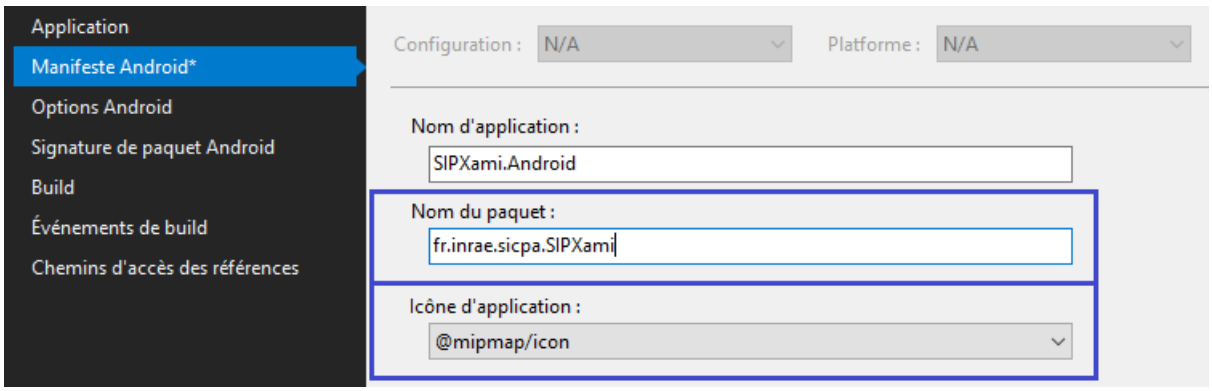


Etape 1: Nom du paquet et Icone de l'application

Le nom du paquet est le nom qu'aura notre paquet APK ainsi que le nom qui sera affiché dans informations détaillées de l'application. Pour nos applications, il conviendra de mettre : fr.inrae.sicpa.<Nom_Du_Programme>

Il est recommandé d'avoir une icône pour l'application. Les applications de distribution de type "marketplace" (ex. Google Play) empêchent de publier une application qui n'en a pas. La propriété **Icon** de l'attribut **Application** permet de définir une icône pour un projet Android.

Sur Visual Studio 2019, en allant dans les propriétés du projet *<Nom_Du_Projet>.Android*.



Pour les icones, il est recommandé par l'équipe Android d'utiliser les dossiers ***mipmap*** fournis comprenant chacun deux fichiers : **icon.png** et **launcher_foreground.png**. Chaque dossier correspond à une densité d'écran (dpi). Heureusement, il n'est pas obligé de tout changer manuellement et il existe un outil mis à disposition par les développeurs Android.

Plus d'informations sur le lien suivant :
https://developer.android.com/studio/write/image-asset-studio#create-legacy .

Note : Le dossier *mipmap-anydpi-v26* est utilisé pour avoir des icones adaptatives (au format XML) sur des appareils utilisant un niveau SDK supérieur a 26, ce qui ne nous concerne pas car nous visons un niveau d'APK 25.

Etape 2: Version de l'application

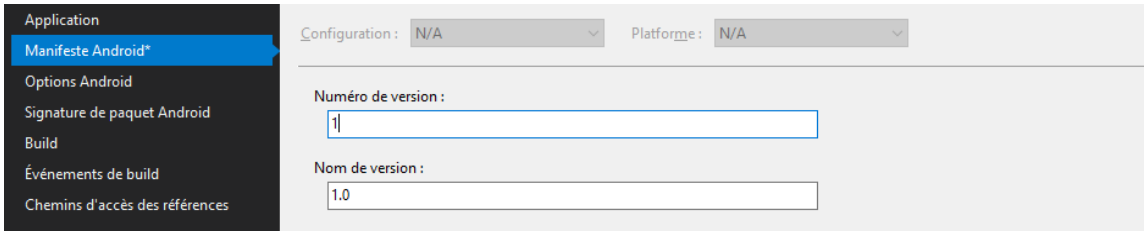
Le versionnage est important pour la maintenance et la distribution des applications Android. Sans une certaine forme de gestion des versions, il est difficile de déterminer si et comment une application doit être mise à jour. Pour faciliter la gestion des versions, Android reconnaît deux types d'informations différentes :

Numéro de version : une valeur *integer* qui représente la version de l'application. Ce numéro s'incrémente de 1 à chaque nouvelle version et ne doit pas être diffusé aux utilisateurs. Il est

indépendant du Nom de version (ci-dessous). Cette valeur est utilisée en interne par Android pour gérer les conflits de version, par exemple.

Nom de version : un *string* qui est utilisé pour informer l'utilisateur de la version de l'application. C'est le nom qui sera affiché aux utilisateur (ex. "v1.2.0"). Cette valeur est purement indicative est n'est pas utilisée en interne par Android.

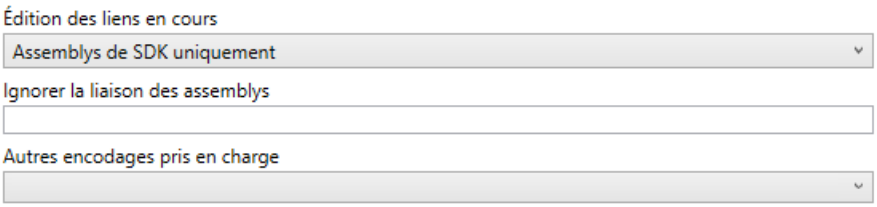
De la même façon que pour l'icône, on définit ces paramètres en allant dans les propriétés du projet (plus bas):



Etape 3: Optimisation de la taille de l'APK

La taille de l'APK Xamarin.Android peut être réduite grâce à la combinaison du *linker* (édition de liens) Xamarin.Android, qui supprime le code inutile. Le processus de construction utilise le *linker* Xamarin.Android pour optimiser l'application au niveau du code géré (C#).

Le mode Release désactive le runtime partagé, qui est utilisé pour le débogage (en mode Debug donc). Cela peut entraîner une réduction significative de la taille du paquet.
En passant dans l'onglet "Options Android" des propriétés, les dernières options affichent :



Le menu déroulant Linking fournit les options suivantes pour contrôler l'éditeur de liens :

- None : Cette option désactive l'éditeur de liens ; aucune liaison ne sera effectuée.

- Assemblys de SDK uniquement : Cette option permet de lier uniquement les assemblages requis par Xamarin.Android. Les autres assemblages ne seront pas liés. **On utilisera celui-ci dans notre cas**.

- Assemblys de SDK et assembly d'utilisateur : Cette option permet de lier tous les ensembles requis par l'application, et pas seulement ceux requis par Xamarin.Android.

Important : La liaison peut produire certains effets secondaires involontaires, il est donc important de tester à nouveau l'application en mode **Release** sur un appareil physique.

Etape 4: Protéger l'application

Il est important de désactiver toujours l'état de débogage dans une application publiée car il est possible (via JDWP) d'obtenir un accès complet au processus Java et d'exécuter du code arbitraire dans le contexte de l'application si cet état de débogage n'est pas désactivé. Une bonne pratique est d'ajouter les lignes de code suivantes dans **AssemblyInfo.cs** (*<Nom_du_projet>.Android* **>** *Properties*)**:**

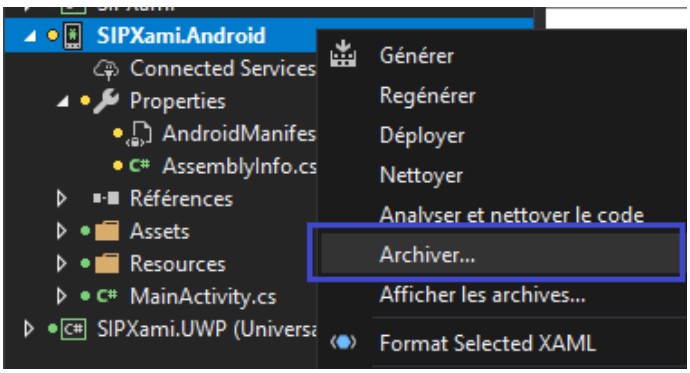
```
#if DEBUG
[assembly: Application(Debuggable=true)]
#else
[assembly: Application(Debuggable=false)]
#endif
```

Etape 5: Compiler

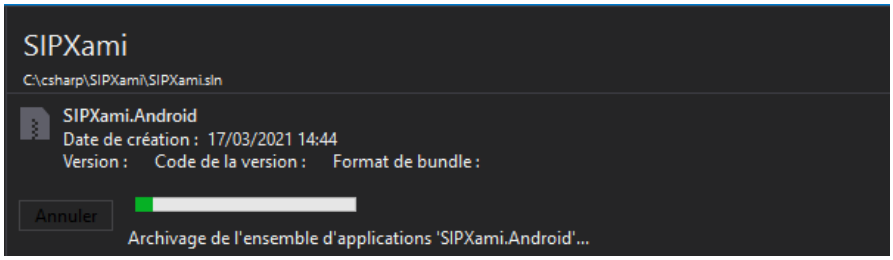
Après avoir complété toutes ces étapes, il est judicieux de recompiler l'application (Onglet **Générer** -> **Regénérer la solution**) pour s'assurer que tout le *build* s'exécute avec succès en mode **Release**. Cette étape ne génère pas encore un APK.

Etape 6: Archiver pour distribution

On accède au processus de distribution en cliquant droit sur le projet Android dans l'explorateur de solutions et en sélectionnant "Archiver…" :



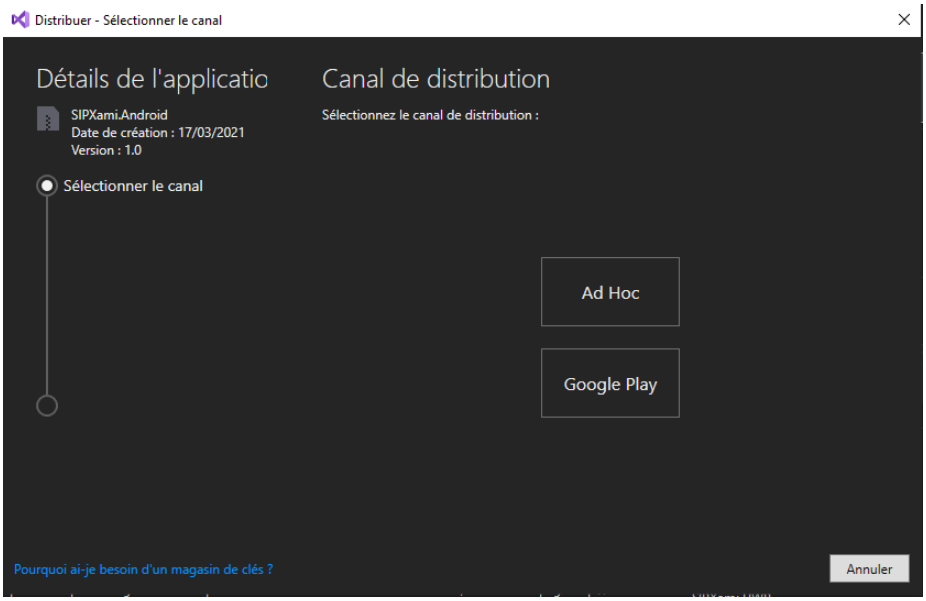
"Archiver…" lance le **Gestionnaire d'archives** et commence le processus d'archivage du bundle d'applications :



Etape 7: Distribution

Lorsqu'une version archivée de l'application est prête à être publiée, sélectionnez l'archive dans le gestionnaire d'archives et cliquez sur le bouton “Distribuer…” :

La boîte de dialogue **Canal de distribution** présente des informations sur l'application, une indication de la progression du *workflow* de distribution et un choix de canaux de distribution. Lors de la première exécution, deux choix sont présentés :



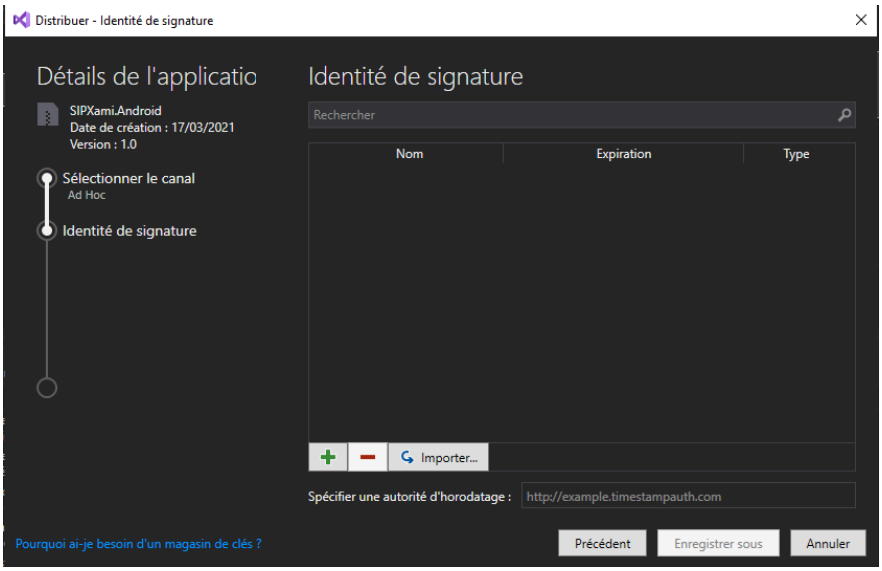
Ad-Hoc: Enregistre sur le disque un APK signé qui peut être chargé sur des appareils Android. C'est un bon moyen de créer un APK pour les tests. **On utilisera cette option** car on ne souhaite pas passer par Google Play.

Google Play: Publie un APK signé sur Google Play.

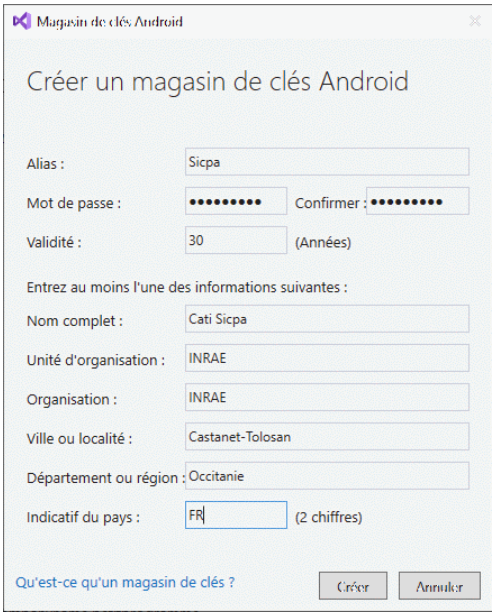
Etape 8: Signer le paquet d'application Android

En utilisant la méthode *ad hoc*, l'APK résultant peut être chargé dans les appareils Android sans passer par une application de distribution.

En sélectionnant l'option "Ad Hoc", Visual Studio ouvre la page **Identité de Signature** de la boîte de dialogue, comme le montre la capture d'écran suivante. Pour publier le .APK, il doit d'abord être signé avec une clé de signature (également appelée certificat).



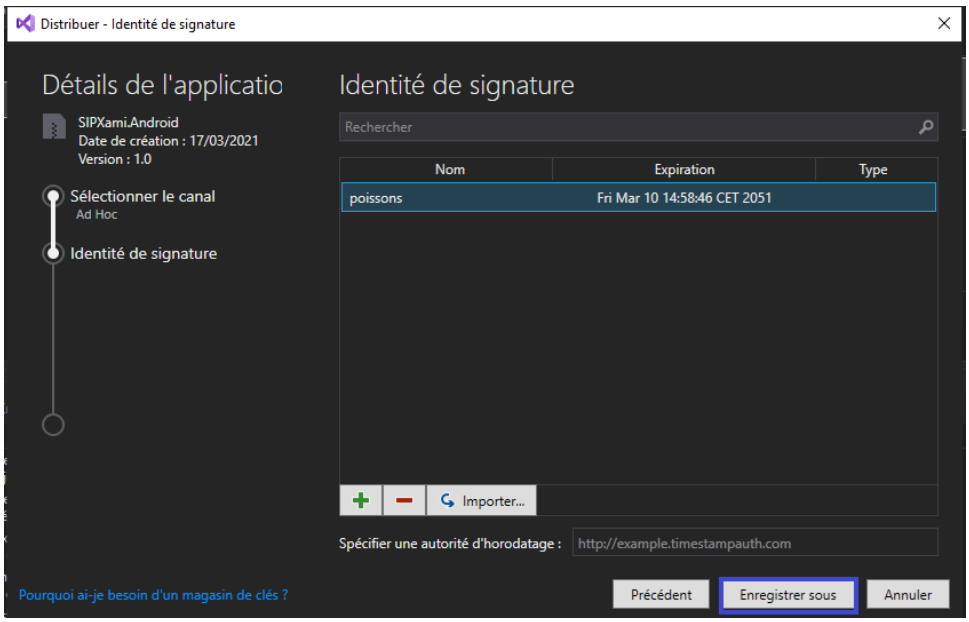
Un certificat existant peut être utilisé en cliquant sur le bouton Importer puis en procédant à la signature de l'APK. Sinon, cliquez sur le bouton + pour créer un nouveau certificat. La boîte de dialogue **Créer un magasin de clés Android** s'affiche ; utilisez cette boîte de dialogue pour créer un nouveau certificat de signature qui pourra être utilisé pour signer des applications Android. Saisissez les informations requises comme l'indique l'exemple suivant :



Le keystore obtenu se trouve à l'emplacement suivant :

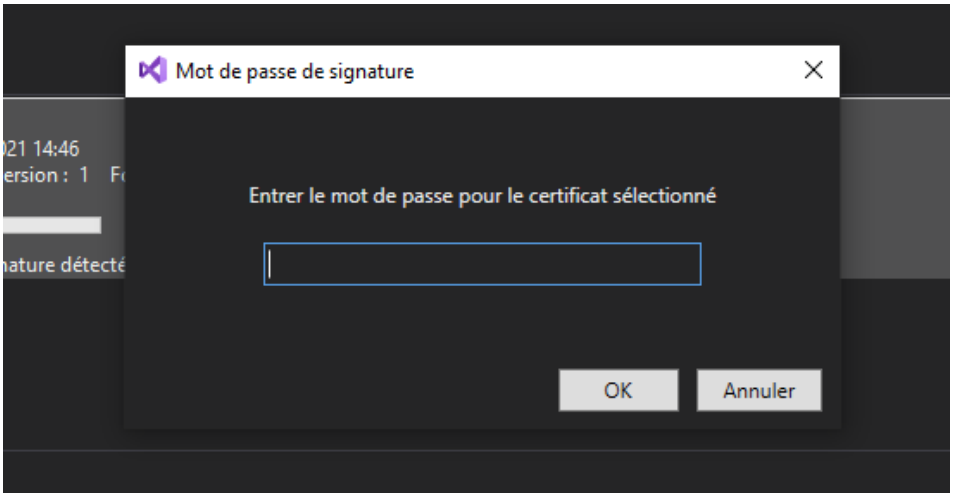
```
C:\Users\USERNAME\AppData\Local\Xamarin\Mono for Android\Keystore\ALIAS\ALIAS.keystore
```

Lorsque vous cliquez sur **Créer**, un nouveau magasin de clés (contenant un nouveau certificat) est enregistré et répertorié sous **Identité de signature**, comme le montre la capture d'écran suivante. Pour publier *ad hoc*, sélectionnez l'identité de signature à utiliser pour la signature et cliquez sur **Enregistrer sous** pour publier l'application pour une distribution indépendante.



Ensuite, le gestionnaire d'archives affiche la progression de la publication. Lorsque le processus de publication est terminé, la boîte de dialogue **Enregistrer sous** s'ouvre pour demander l'emplacement où le fichier .APK généré doit être stocké.

Placez-vous jusqu'à l'emplacement souhaité et cliquez sur **Enregistrer**. Si le mot de passe de la clé est inconnu, la boîte de dialogue **Mot de passe de signature** s'affiche pour demander le mot de passe du certificat sélectionné :



Une fois le processus de signature terminé, cliquez sur Ouvrir la distribution.

L'Explorateur Windows ouvre alors le dossier contenant le fichier APK généré. À ce stade, Visual Studio a compilé l'application Xamarin.Android en un APK prêt à être distribué.

Source: https://docs.microsoft.com/en-us/xamarin/android/deploy-test/release-prep/?tabs=windows