flutter_native_splashのより詳細な使い方はpub.dev[注6]を参照してください。

アプリのID

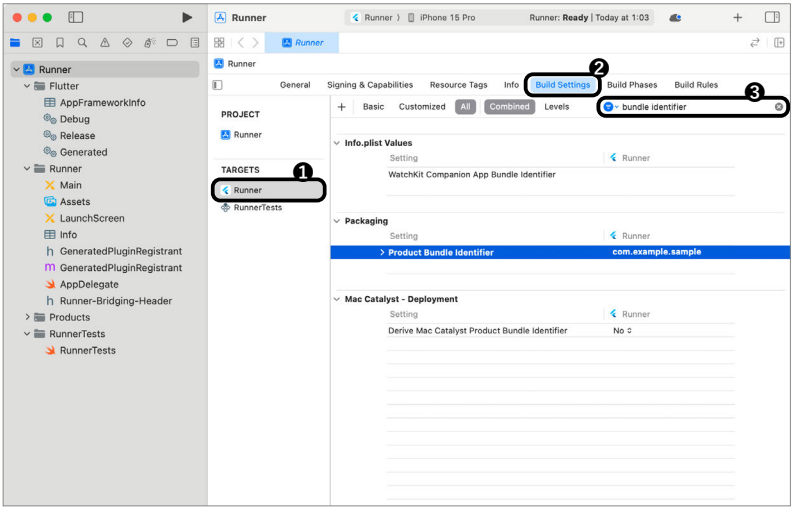
iOSもAndroidもそれぞれアプリを一意に識別するIDがあります。iOSはBundle ID、AndroidはApplication IDと呼ばれます。このIDは他のアプリと重複しないように、自分が所有するドメインを逆順にしたものを使用するのが一般的です。たとえば、筆者が所有するドメインが`example.com`であれば、`com.example.myapp`のようになります。

プログラムがまったく同じアプリでもアプリIDが異なると別のアプリとして扱われます。これを逆手にとり、検証環境のサーバに接続するアプリと本番環境のサーバに接続するアプリを1台の端末に共存させることができます。

iOSのアプリIDを設定する

iOSのアプリIDはプロジェクトファイルに記述されています。Xcodeで編集するのが無難です。`ios/Runner.xcworkspace`をXcodeで開き、TARGETSの「Runner」を選択して(❶)「Build Settings」のタブを開きます(❷)。「bundle identifier」でフィルタをかけると(❸)、すぐに設定項目が見つかります(**図11.6**)。

図11.6 **bundle identifierの設定の様子**



注6 https://pub.dev/packages/flutter_native_splash

Androidのアプリ IDを設定する

Androidのアプリ IDはアプリのビルド構成ファイル`android/app/build.gradle`に記述します。`android`エントリ内`defaultConfig`の`applicationId`に設定します。

```
./android/app/build.gradle
android {
    defaultConfig {
        applicationId "com.example.myapp"
    }
}
```

11.3 アプリの配布とコード署名

App StoreやGoogle Play Storeでアプリを配布するには、アプリにデジタル署名をする必要があります。筆者の考えるモバイルアプリの鬼門はこのコード署名です。

コード署名とは、アプリの開発元が正しく、配布されたアプリが改ざんされていないことを証明するしくみです。秘密鍵を使ってアプリに署名し、その秘密鍵とペアになる公開鍵の証明書をアプリの中に埋め込みます。こうすることで、アプリが改ざんされていないか検証することができるのです。iOSとAndroidとで証明書の取り扱いがまったく異なり、Flutterエンジニアにとっては敷居が高いものとなっています。

本節では、iOSとAndroidのコード署名のしくみ、App StoreやGoogle Play Storeでアプリを配布するための署名の方法を解説します。なお、AppleとGoogleの開発者アカウントが必要となりますが、これらを事前に取得していることを前提として解説します。Appleの開発者アカウントとGoogleの開発者アカウントはそれぞれ以下から登録できます。

- `https://developer.apple.com/jp/programs/`
- `https://developer.android.com/distribute/console`

料金が発生しますのでよく確認してから登録してください。

また、本章で利用するアカウントはApple Developer Programに個人登録したApple IDを想定しています。組織登録した場合やApple Developer Enterprise

Programに登録した場合などではアカウントがアプリの登録、アプリID（AppID）やプロビジョニングプロファイルの作成、証明書の作成に必要な権限を付与されている必要があります。

iOSのコード署名

iOSのコード署名と検証のプロセスには、プロビジョニングプロファイルというファイルが関わってきます。プロビジョニングプロファイルはアプリIDと証明書が含まれたファイルです。アプリIDと証明書はAppleの開発者アカウントに紐付いています。

Xcodeプロジェクトは以下の4つの設定項目があり、すべてが正しく設定されていなければ、アプリの署名は成功しません。

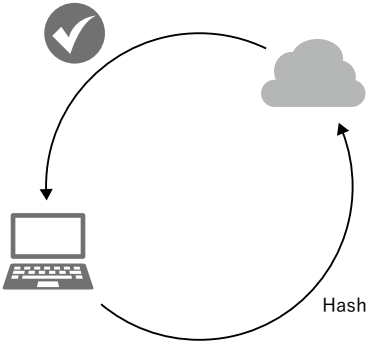
- **Appleの開発者アカウント（Team ID）**
- **アプリID**
- **証明書**
- **プロビジョニングプロファイル**

管理の難しい秘密鍵

iOSのコード署名で管理が難しいのが秘密鍵です。証明書を発行したMacにのみ秘密鍵が存在し、他のMacに移す際は証明書とともにエクスポートする必要があります。秘密鍵を紛失した場合は、証明書を再発行する必要があります。また、証明書には有効期限があり、有効期限が切れた場合もまた再発行する必要があります。

これらの管理の手間を軽減する手段としてクラウド管理対象証明書というしくみが導入されました。秘密鍵や証明書はAppleが管理し、ビルド環境には必要ありません。任意の環境でビルドしたアプリのハッシュ値をAppleに送信し、署名情報を取得してアプリに署名するというしくみです（**図11.7**）。

図11.7 クラウド管理対象証明書のしくみ

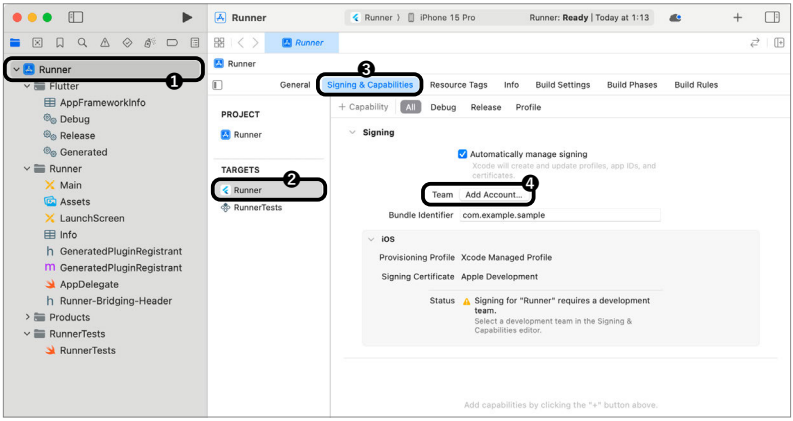


クラウド管理対象証明書を使うかどうかは選択可能です。本書ではこのしくみを使ってアプリを署名する方法を解説します。

アプリに署名する

それでは、クラウド管理対象証明書を使ってアプリに署名する手順を解説します。`ios/Runner.xcworkspace`をXcodeで開きます。**図11.8**のようにナビゲーションペインの「Runner」を選択し(❶)、TARGETSの「Runner」を選択します(❷)。

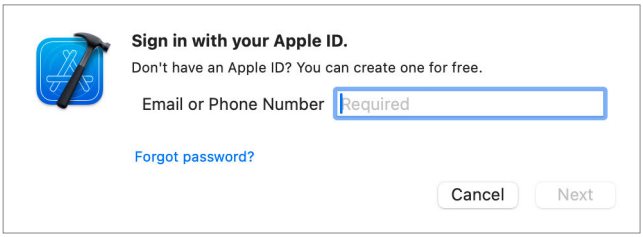
図11.8 「Signing & Capabilities」のタブを表示



続いて、「Signing & Capabilities」のタブを選択します(❸)。

「Add Account...」のボタンをクリックする(❹)とログイン画面が表示されますので、Appleの開発者アカウントでログインします(**図11.9**)。

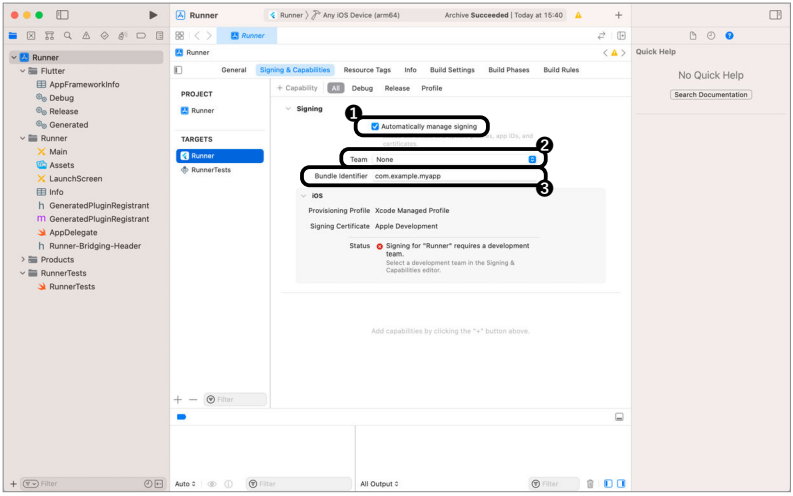
図11.9 **ログインダイアログ**



ログインが完了したら設定画面を閉じます。

図11.10の画面が表示されたら、まず「Automatically manage signing」のチェックボックスがオンになっていることを確認します(❶)。次に「Team」のドロップダウンリストからAppleの開発者アカウントを選択します(❷)。最後にアプリIDを設定します[注7](❸)。アプリIDの変更は「iOSのアプリIDを設定する」の項で解説した方法か、この「Signing & Capabilities」のタブからも行えます。

図11.10 **Team、アプリIDの設定の様子**



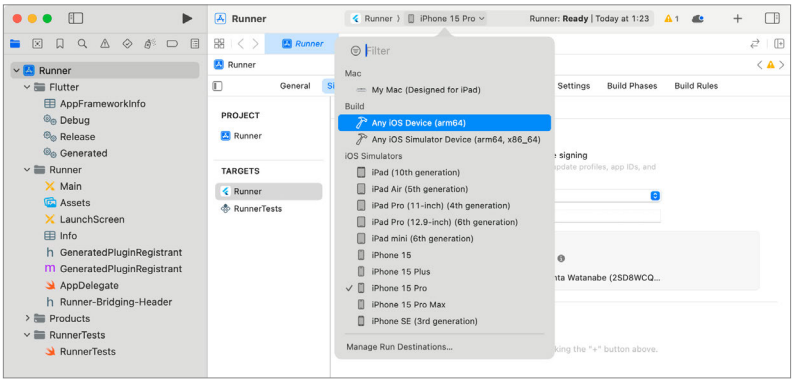
クラウド管理対象証明書を使う場合、設定は以上です。

いよいよアプリの署名を開始します。ビルド対象を「Any iOS Device」に設

注7 図ではアプリIDが「com.example...」となっていますが、アプリIDとしては無効な文字列です。検証される際は有効なアプリIDを設定してください。

定します(**図11.11**)。

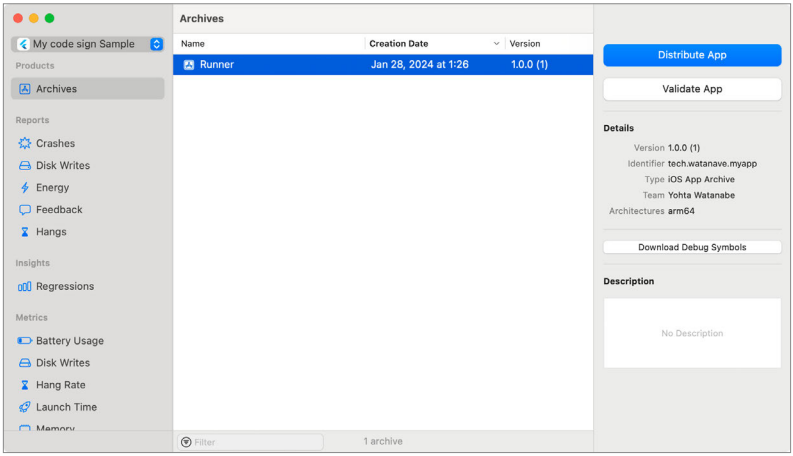
図11.11 **ビルド対象の選択**



Xcodeのメニューから「Product」➡「Archive」を選択します。アーカイブとは、アプリの成果物をひとまとめにしたバンドルを作成することです。

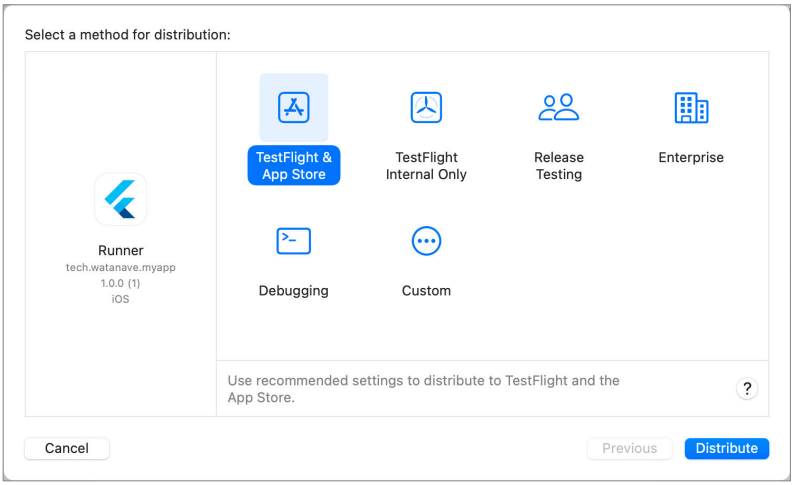
完了するとオーガナイザーが開きます。通常、先ほどの操作で作成したアーカイブが選択されていますので、そのまま「Distribute App」をクリックします(**図11.12**)。

図11.12 **オーガナイザーウィンドウ**



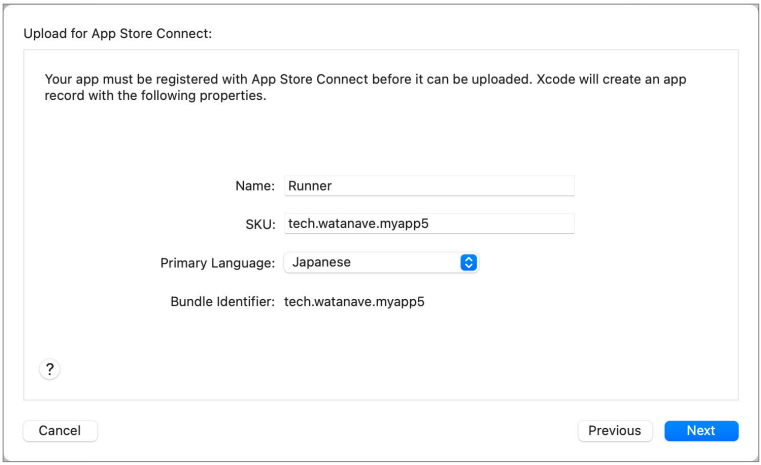
配布先の選択画面が表示されますので、「TestFlight & App Store」を選択します(**図11.13**)。

図11.13 配布先の選択画面



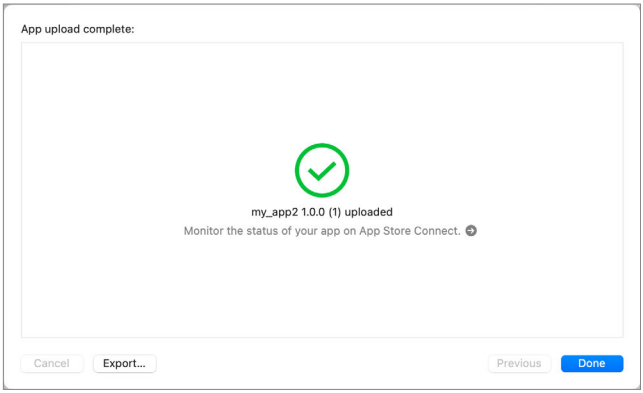
App Store Connectにアプリが登録されていない場合は、アプリ情報を入力するダイアログが表示されます。「Name:」はApp Storeに表示されるアプリ名です。「SKU:」はアプリを一意に識別するためのIDです。デフォルトではBundle IDと同じ値が設定されていて、そのままで問題ありません。日本語にのみ対応するアプリの場合は「Primary Language:」は「Japanese」に設定しましょう。最後に「Bundle Identifier」に誤りがないことを確認して「Next」をクリックします(**図11.14**)。

図11.14 アプリ情報の入力画面



するとアプリの署名とApp Store Connectへのアップロードが開始されます。完了すると**図11.15**の画面が表示されます。

図11.15 **アップロード完了画面**



画面左下の「Export」ボタンをクリックし、任意のディレクトリに出力するとアプリの要約が確認できます。`DistributionSummary.plist`というファイルが出力されますので、テキストエディタで開いてみましょう。

```
./DistributionSummary.plist
<!-- 省略 -->
            <key>certificate</key>
            <dict>
                <key>SHA1</key>
                <string>ハッシュ値</string>
                <key>dateExpires</key>
                <string>有効期限</string>
                <key>type</key>
                <string>Cloud Managed Apple Distribution</string>
            </dict>
```

キー`certificate`の`type`が「Cloud Managed Apple Distribution」になっており、クラウド管理対象証明書で署名されていることがわかります。

Androidのコード署名

Androidでは秘密鍵と公開鍵証明書を格納したキーストアというファイルを使用します。iOSと同様に、Androidも証明書をクラウドで管理するしくみがあり、Google Play Consoleではこの方法を推奨しています。その手順は以下のとおりです。