

- ❶開発者がキーストアを作成する
- ❷開発者が❶のキーストアを使い、アプリに署名をする
- ❸署名済みの❷のアプリを Google Play Console にアップロードする
- ❹Google Play Console がアプリの署名を検証する
- ❺Google Play Console が生成・管理するキーストアでアプリを再署名する
- ❻再署名済みのアプリを Google Play Store に配布する

❶で作成したキーストアの秘密鍵をアップロード鍵、❺で Google Play Console が生成・管理するキーストアの秘密鍵をアプリ署名鍵と呼びます。

Android はアプリをアップデート(新しいバージョンのアプリを上書きインストール)する際に、同じ証明書で署名されている必要があります。証明書を紛失してしまうと、アプリのアップデートができなくなってしまう。そのため、ユーザーの手もとで検証される証明書を Google Play Console がクラウド上で管理することで、リスクを軽減しているのです。

手順が非常に多いように見えますが、実際には Android Studio の GUI を使って簡単に行うことができます。

apk ファイルと aab ファイル

Android のプロジェクトは、apk ファイルと aab ファイルの2種類のファイルを生成することができます。インストール可能なアプリのファイル形式は apk ファイルです。aab は Android App Bundle の略で、そのままではインストールできません。aab はコンパイル済みのコードとリソースをすべて内包したファイルで、インストールする端末向けに aab から最適化した apk ファイルを生成することができます。

Google Play で公開するアプリは、aab の形式でのアップロードしかサポートされていません。Google Play Console が aab ファイルを apk ファイルに変換、その際にアプリ署名鍵で署名を行い、ユーザーの端末にインストールされるのです。

アプリに署名する

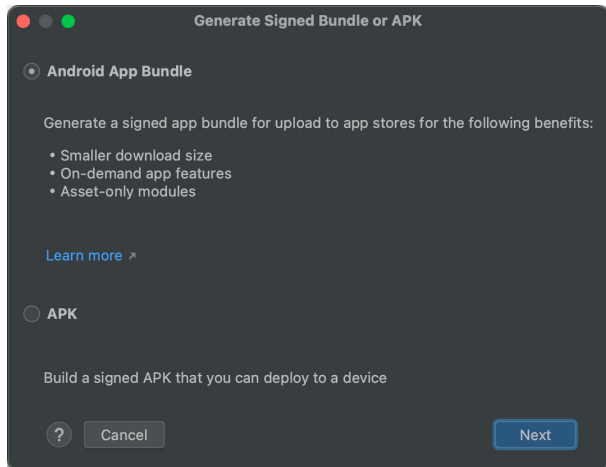
それでは、アップロード鍵の生成とアプリへの署名手順を解説します。Android Studio を使うと簡単に行えます。

プロジェクトルート直下の **android** ディレクトリを Android Studio で開きま

す。アプリケーションメニューの「Build」→「Generate Signed Bundle/APK」を選択します。

ダイアログで「Android App Bundle」を選択し、「Next」をクリックします(図 11.16)。

図 11.16 ファイル形式の選択ダイアログ



次の画面で「Create new...」を選択し、アップロード鍵を生成します(図 11.17)。

図 11.17 署名設定のダイアログ

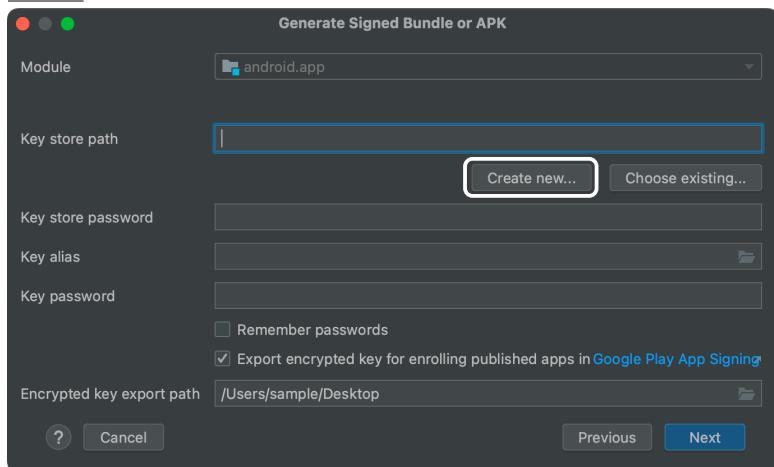
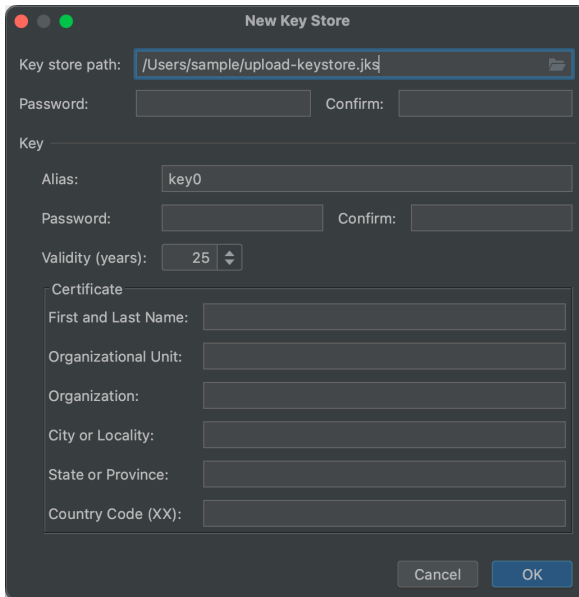


図11.18のダイアログの上部から、「Key store path:」にはキーストアの保存先を指定します。ファイルの拡張子は「.jks」を指定します。「Password:」と「Confirm:」にはキーストアのパスワードを入力します。「Alias:」には鍵を識別する任意の名前を入力します。その下の「Password:」と「Confirm:」には先ほどのキーストアのパスワードと同じものを入力します。「Validity (years):」は鍵の有効期限を指定します。25年以上が推奨されています。「Certificate」には証明書の所有者情報を入力します。ストアで公開されることはありませんが、証明書の情報としてアプリに組み込まれます。

「OK」をクリックすると、キーストアが生成されます。

図11.18 キーストア作成ダイアログ

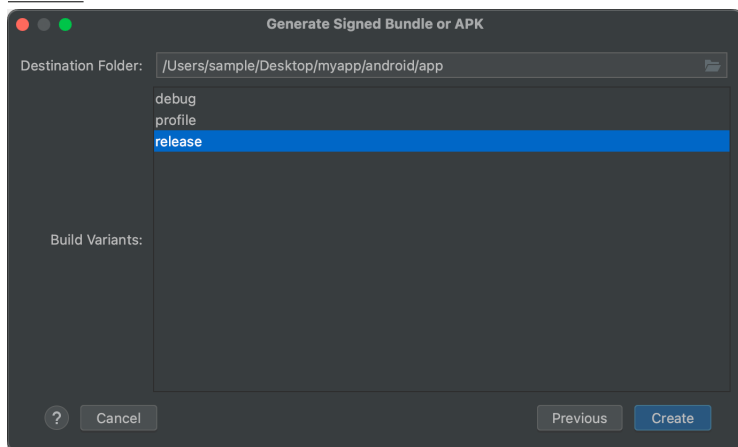
A screenshot of the 'New Key Store' dialog box. It has a title bar with standard window controls. The 'Key store path:' field contains '/Users/sample/upload-keystore.jks'. Below it are 'Password:' and 'Confirm:' fields. The 'Key' section has an 'Alias:' field with 'key0', and another 'Password:' and 'Confirm:' pair. The 'Validity (years):' field is set to 25. The 'Certificate' section contains several text fields for 'First and Last Name', 'Organizational Unit', 'Organization', 'City or Locality', 'State or Province', and 'Country Code (XX)'. At the bottom are 'Cancel' and 'OK' buttons.

もとのダイアログ(図11.17)に戻ったら「Next」をクリックします。

次の画面では「release」を選択し、「Create」をクリックします(図11.19)。

ビルドが完了すると、`android/app/release`ディレクトリにaabファイルが生成されます。

図 11.19 ビルドバリエانتの選択ダイアログ

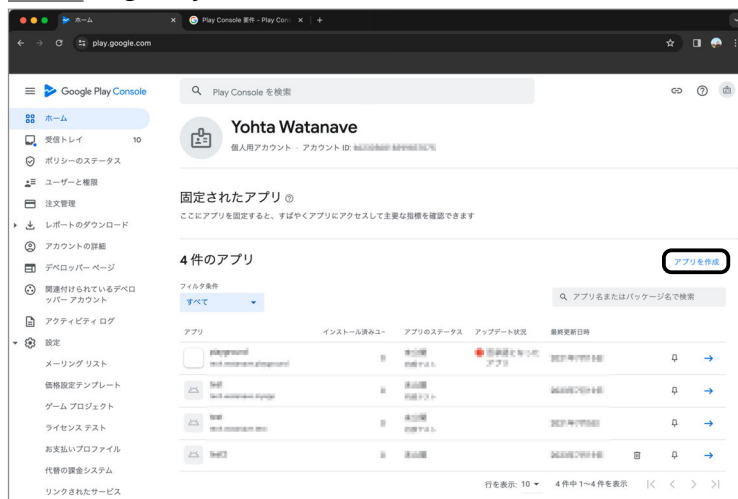


aabファイルをアップロードする

実際にaabファイルをGoogle Play Consoleにアップロードし、処理される様子を見てみましょう。aabファイルをアップロードするために、Google Play Consoleでアプリの情報登録などの操作が必要になりますが、その操作については駆け足で説明します。本項はアプリの署名にフォーカスした内容のためご容赦ください。

Google Play Consoleにログインし、右上の「アプリを作成」をクリックします(図11.20)。

図 11.20 Google Play Consoleのトップ画面

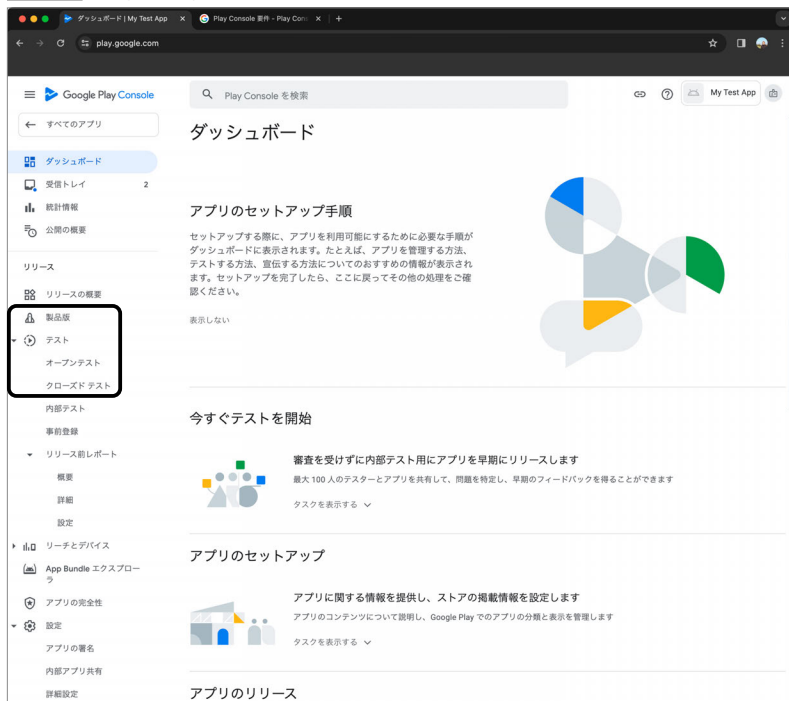


アプリの必要情報を入力し、「アプリを作成」をクリックします(図 11.21)。

図 11.21 Google Play Console のアプリ作成画面

左側のメニューに「製品版」のほか、「テスト」の配下に「オープンテスト」や「クローズドテスト」などがあります(図 11.22)。

図11.22 Google Play Consoleのダッシュボード画面



「製品版」はGoogle Playで公開中のバージョンを管理する画面です。アプリを「製品版」として公開する前にテスターに配布するなどの用途で「テスト」以下の項目を使います。今回はアップロードの動作を見ることを目的に「内部テスト」を選択します。「内部テスト」をクリックし、右上の「新しいリリースを作成」をクリックします(図11.23)。

図11.23 Google Play Consoleの内部テスト画面

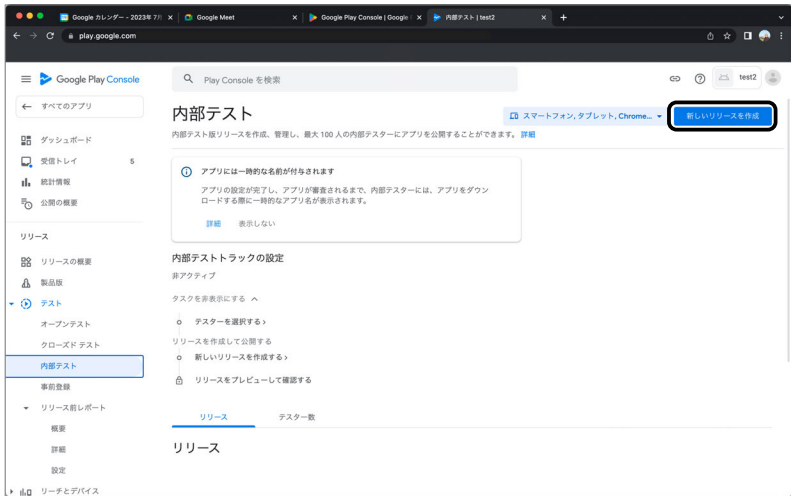


図11.24の画面で「署名鍵を選択」をクリックすると、署名鍵の選択ダイアログが表示されます(図11.25)。「Google生成の鍵を使用」を選択します。

図11.24 Google Play Consoleのリリース作成画面



図 11.25 署名鍵の選択ダイアログ

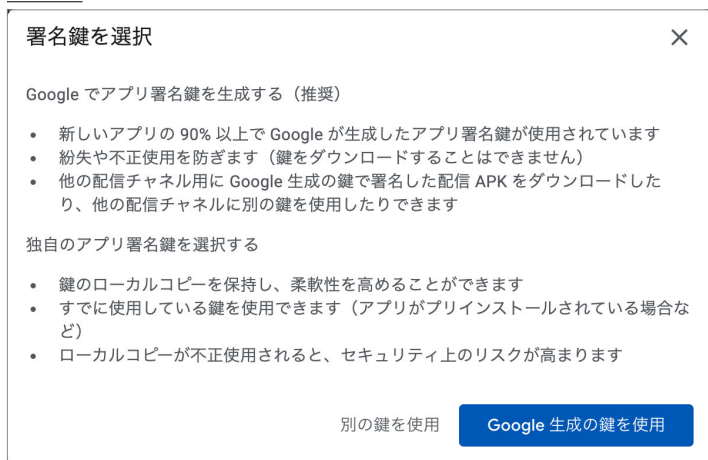


図 11.24 の画面に戻ったら「アップロード」をクリックすると、ファイル選択ダイアログが表示されますので、先ほど生成した aab ファイルを選択します。続いて「リリースの詳細」に必要な事項を入力し、「次へ」をクリックします。

右下の「保存して公開」をクリックすると、内部テストのリリースが作成されます。

aab ファイルのアップロードが完了しました。これで先ほどアプリをビルドする際に作成したキーストアがアップロード鍵として自動的に登録されます。次回以降のアップロードでは、同じキーストアを使ってビルドしなければアップロード時にエラーとなります。

11.4

まとめ

Flutter アプリを開発するうえで欠かせない iOS、Android ネイティブの知識を解説しました。Swift や Kotlin に触れずとも Flutter アプリを開発することは可能ですが、本章で紹介したアプリ ID や最低サポート OS バージョンの設定などは、欠かすことのできない知識です。

また、アプリの署名に関しては、時にはネイティブアプリエンジニアも苦戦することのある難しいものと筆者は感じています。本章の内容が、みなさんの Flutter アプリをリリースする一助になれば幸いです。