Build your app

TWIXL

Table of Contents

Building Apps

Building your Twixl app

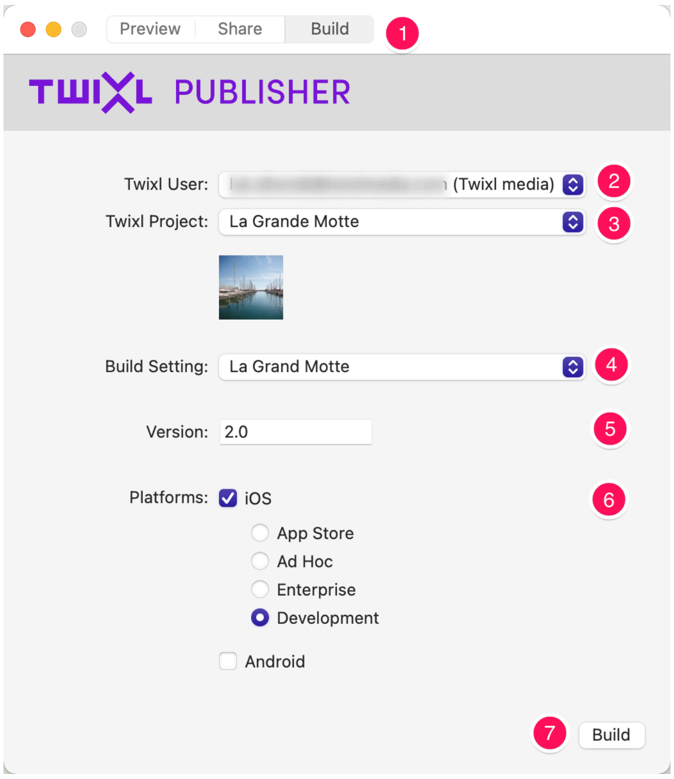
⚠ **REQUIREMENTS:**

In order to build a Twixl app, first double-check whether you have gone through the following steps first:

- Activate a Twixl user in the macOS App.
- Configure a build setting on the platform. You may need to create an identifier (App ID) on https://developer.apple.com for a new app, or otherwise copy the App ID for an existing app.
- Configure Xcode.
- Create a signing certificate.
- **IMPORTANT for users of an 'Apple Silicon' M1 Mac:** make sure to first register your device on the Apple Developer Portal, see here.

Building your Twixl App

1. Select the tab *Build*
2. Select the appropriate **Twixl User**
3. Select the **Twixl Project** you want to build an app for
4. Choose the correct **Build Setting**
5. Enter a **Version Number**
6. Choose whether you want to build an **iOS** and/or **Android** app.
7. Click *Build*
8. Wait for the process to finish
9. Done!



Twixl User: [redacted] (Twixl media)
Twixl Project: La Grande Motte
Build Setting: La Grand Motte
Version: 2.0
Platforms: [✓ iOS]
- App Store
- Ad Hoc
- Enterprise
- [✓ Development]

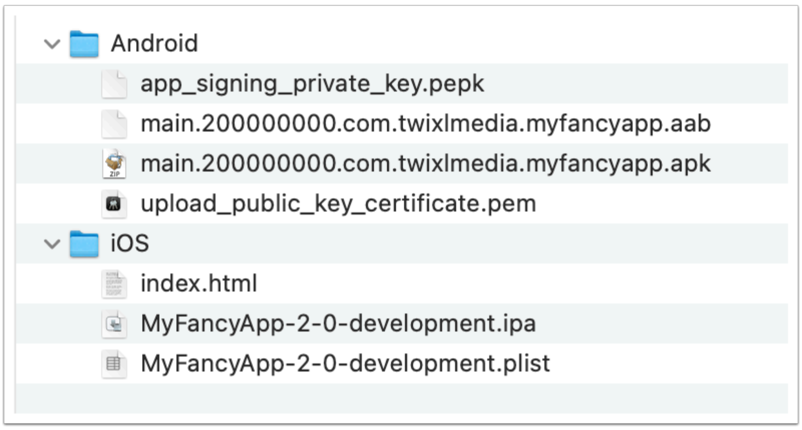
Android

For Android apps, the following files will be generated:

- the application itself, in two formats, i.e. as an .aab (Android App Bundle)
- the app signing upload key (with extension .pepk)
- the upload public key (with extension .pem)

For iOS apps, the following file(s) will be created:

- the application itself (with extension .ipa)
- an index.html and a .plist file (only if you have selected over the air distribution)



```
Android
    app_signing_private_key.pepk
    main.200000000.com.twixlmedia.myfancyapp.aab
    main.200000000.com.twixlmedia.myfancyapp.apk
    upload_public_key_certificate.pem
iOS
    index.html
    MyFancyApp-2-0-development.ipa
    MyFancyApp-2-0-development.plist
```

ABOUT THE VERSION NUMBER:

Every time you want to submit an update for your app in the stores, the version number of your app should get a bump.

- **Good example:**
  - Current version: `1.5`
  - Update: `1.6`
- **Bad example:**
  - Current version: `1.5`
  - Update: `1.4.9`

Build Settings overview

The Build Settings for your apps are managed and stored on the Twixl platform. This article explains how to create a Build Setting and what the available settings exactly mean.

What is a Build Setting?

Before you can actually build your app in Twixl Publisher, Twixl needs some information in order to be able to build an app. This information has to be entered in the **Build Settings** on the Twixl platform.

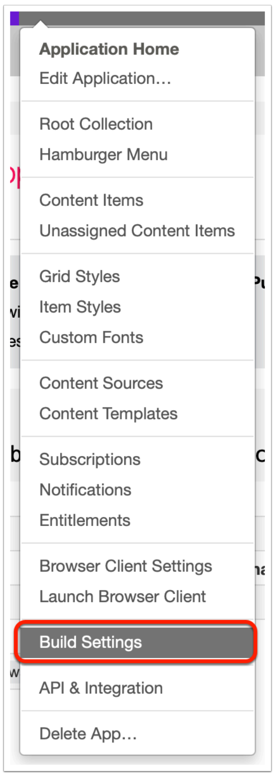
A **Build Setting** consists of a number of configuration options that determine the type of app that you will be creating, and which features will be enabled.

E.g. a build setting can contain the following info: which user interface languages are supported, the artwork (app icon, etc…), code signing information, etc.

Once a build setting has been saved, creating a build of the app only requires a few mouse clicks. A build setting can also be exported so it can be imported in another Twixl account.

How to create a new build setting

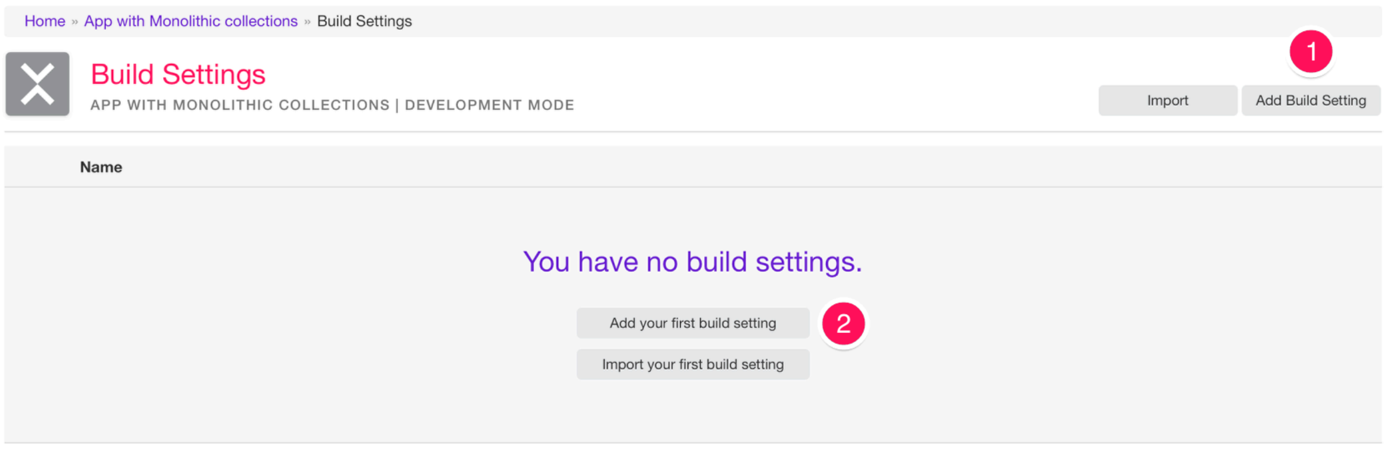
1. Go to the Twixl platform
2. Login
3. Select the app you need to create a build setting for
4. From the App Menu, select *Build Settings*



Start from scratch

If you want to create a new build setting, select one of the following:

1. *Add Build Setting*
2. *Add your first build setting*

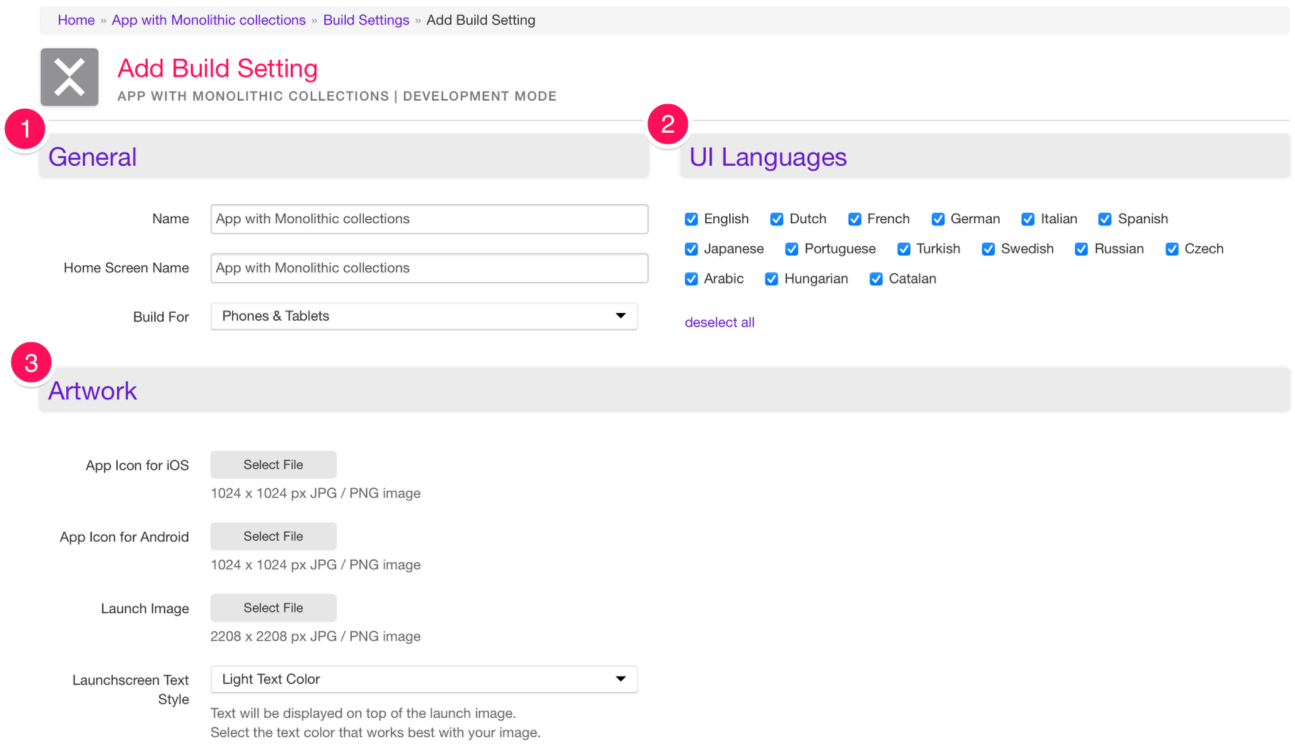

Home » App with Monolithic collections » Build Settings

Build Settings

APP WITH MONOLITHIC COLLECTIONS | DEVELOPMENT MODE

Name

You have no build settings.


Home » App with Monolithic collections » Build Settings » Add Build Setting

Add Build Setting

APP WITH MONOLITHIC COLLECTIONS | DEVELOPMENT MODE

1 General

Name	App with Monolithic collections
Home Screen Name	App with Monolithic collections
Build For	Phones & Tablets

2 UI Languages

✓ English ✓ Dutch ✓ French ✓ German ✓ Italian ✓ Spanish
✓ Japanese ✓ Portuguese ✓ Turkish ✓ Swedish ✓ Russian ✓ Czech
✓ Arabic ✓ Hungarian ✓ Catalan

deselect all

3 Artwork

App Icon for iOS	Select File 1024 x 1024 px JPG / PNG image
App Icon for Android	Select File 1024 x 1024 px JPG / PNG image
Launch Image	Select File 2208 x 2208 px JPG / PNG image
Launchscreen Text Style	Light Text Color Text will be displayed on top of the launch image. Select the text color that works best with your image.

1. General

- **Name:** The name of your build setting.
- **Home Screen Name:** The name of your app, visible on the home screen of your mobile device
- **Build for:** Which types of devices are supported? Phone, tablets or both?

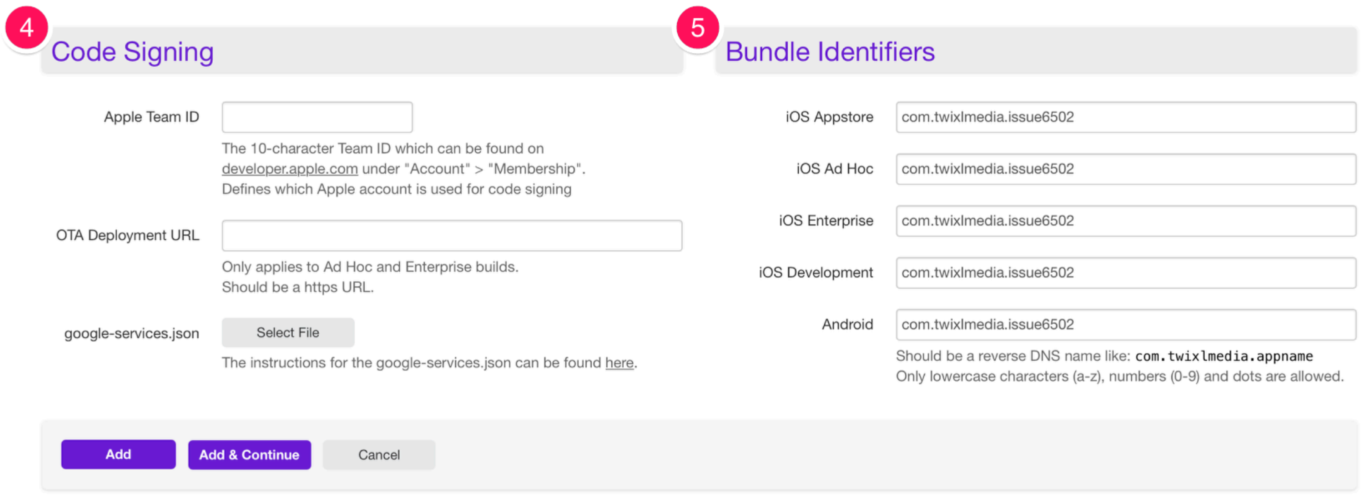
2. UI Languages

Here you can select the user interface languages your app will support. This is related to Twixl-specific messages that may be shown to the user/reader. Either you can check only the languages that are relevant for your market, but there's no harm in just checking all the languages.

3. Artwork

- **App Icon:** You can choose a different icon for iOS and Android
- **Launch Image:** The launch image of your app - see also this article.
- **Launch screen Text Style:** Text will be displayed on top of the launch image. Select the text color that works best with your image.

Especially for Android devices, creating an App icon can be confusing as the result might show something different than intended. This is because a lot of manufacturers of mobile Android devices create their own UI on top of the Android UI. If you want to be certain your design fits most devices, check your icon with Android Studio! Read all about it on this Android helppage.


4 Code Signing

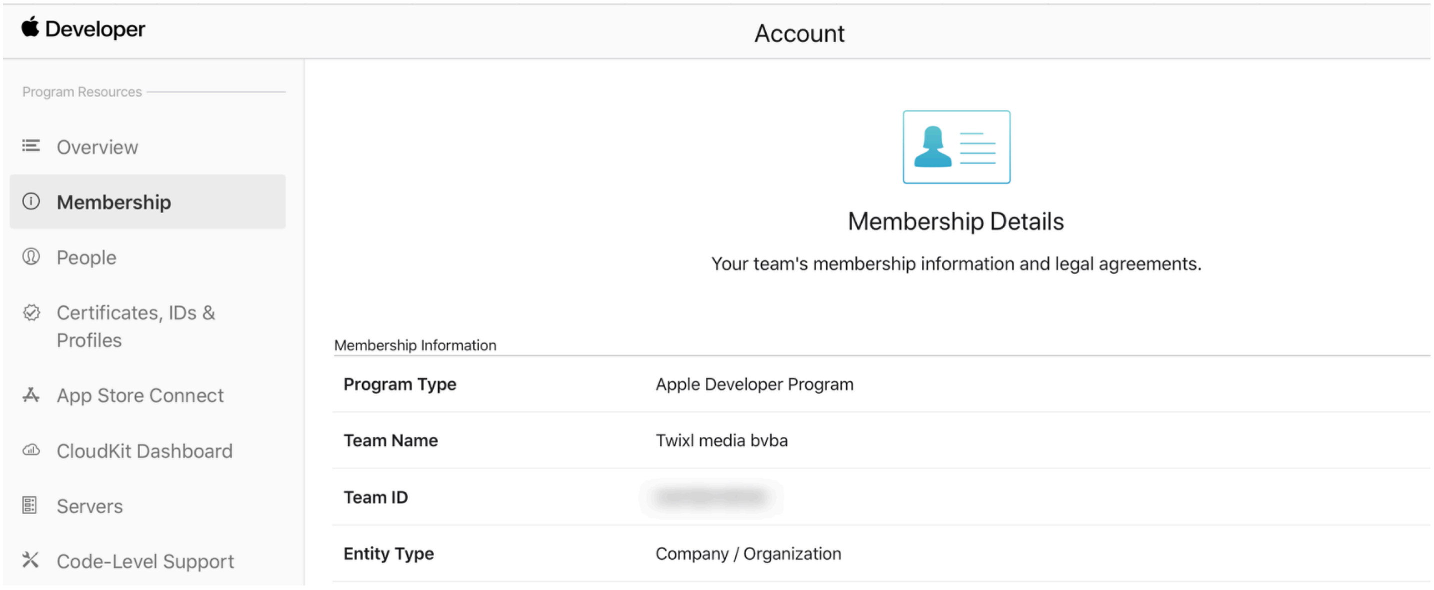
Apple Team ID	
	The 10-character Team ID which can be found on developer.apple.com under "Account" > "Membership". Defines which Apple account is used for code signing
OTA Deployment URL	
	Only applies to Ad Hoc and Enterprise builds. Should be a https URL.
google-services.json	Select File
	The instructions for the google-services.json can be found here.

5 Bundle Identifiers

iOS Appstore	com.twixlmedia.issue6502
iOS Ad Hoc	com.twixlmedia.issue6502
iOS Enterprise	com.twixlmedia.issue6502
iOS Development	com.twixlmedia.issue6502
Android	com.twixlmedia.issue6502

Should be a reverse DNS name like: `com.twixlmedia.appname`
Only lowercase characters (a-z), numbers (0-9) and dots are allowed.

4. Code Signing



Membership Details

Your team's membership information and legal agreements.

Membership Information

Program Type	Apple Developer Program
Team Name	Twixl media bvba
Team ID	[redacted]
Entity Type	Company / Organization

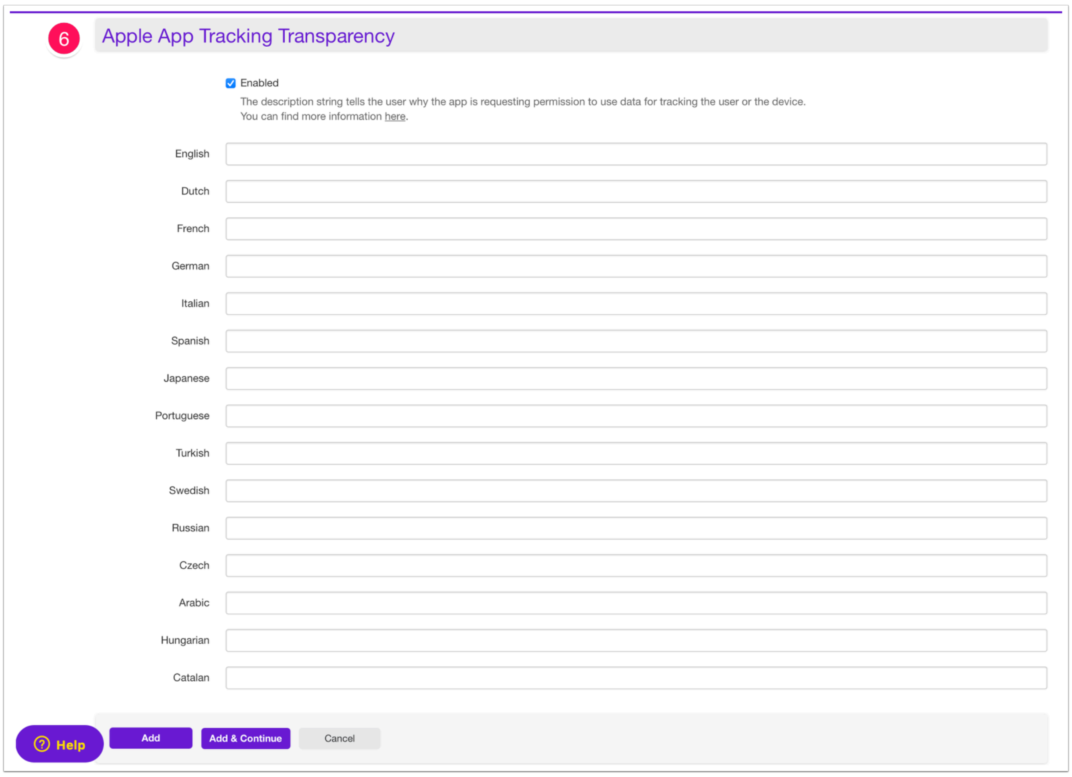
- **Apple Team ID:** Required to code sign and build your app. Navigate to https://developer.apple.com/account and click on *Membership* in the left column, copy the *Team ID,* a 10-character code, and paste it in the Team ID field.

- **OTA Deployment URL:** Only applies to Ad Hoc and Enterprise builds. Should be a `https` URL. See this KB-article for more info.
- **google-services.json:** Used for setting up Push Notifications on Android devices. See this KB-article for more info.

5. Bundle Identifiers

- A bundle ID or bundle identifier uniquely identifies an app. This means that no two applications can have the same bundle identifier. To avoid conflicts, Apple encourages developers to use reverse domain name notation for choosing an application's bundle identifier, like `com.twixlmedia.appname` . If the value for an App Type is empty, that App Type can't be built. We recommend to use the same identifier for all app types.

6. Apple App Tracking Transparency


6 Apple App Tracking Transparency

☑ Enabled

The description string tells the user why the app is requesting permission to use data for tracking the user or the device.
You can find more information here.

English	
Dutch	
French	
German	
Italian	
Spanish	
Japanese	
Portuguese	
Turkish	
Swedish	
Russian	
Czech	
Arabic	
Hungarian	
Catalan	

Apple introduced **App Tracking Transparency** in iOS 14.5. If you want to ask permission to your users to track them, you can enable this option in the build settings for your app, and you can enter the correct usage description of why you are asking permission in all relevant languages.

⚠ **What happens when users select 'Do not track' ?**

Please note that Twixl doesn't change anything to the way cookies are handled in online web content. In order to allow you to handle cookies properly when a user asks not to be tracked, you can read the `trackingAuthorizationStatus` custom variable from within your web page to determine whether or not the user has allowed to be tracked. This variable can be read by using the `tp-get-custom-vars://` URL scheme from within JavaScript.

How to import an existing 'legacy' build setting

If you have previously built Twixl apps using a release ***before Twixl Publisher 12***, it's good to know that you can easily upload your existing build settings to the Twixl platform:

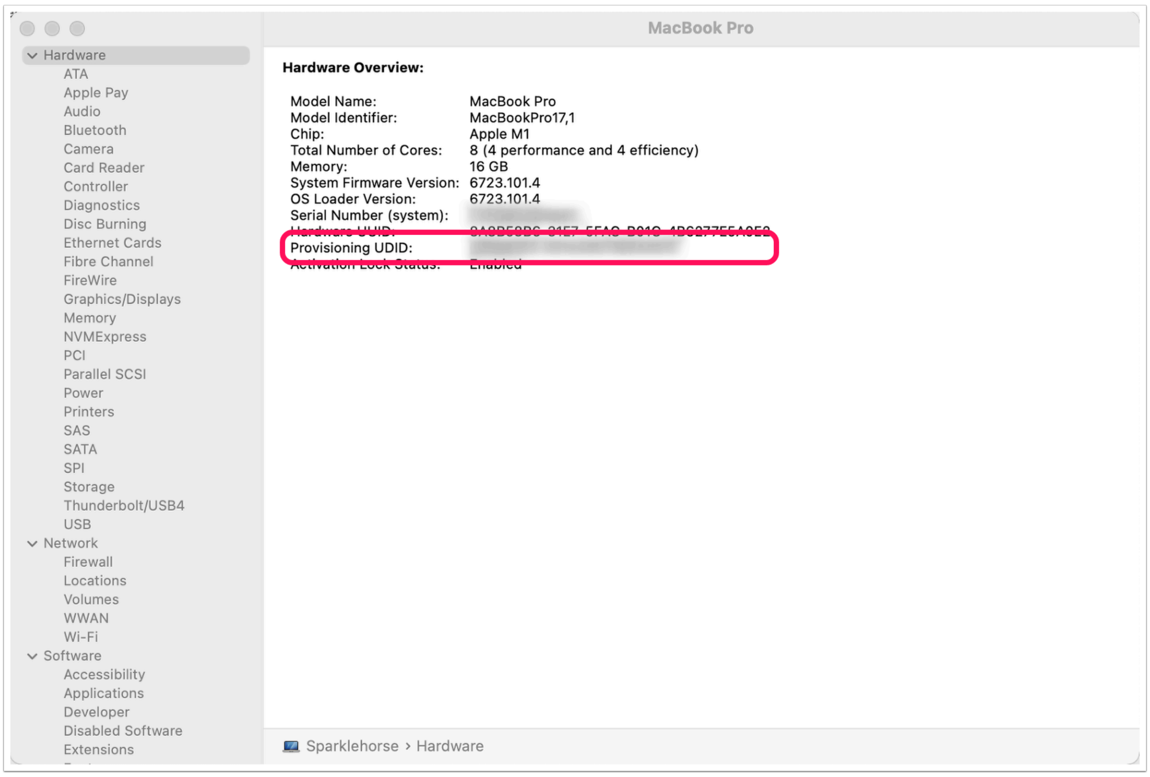
1. Open the Twixl macOS App.
2. Select the *Tools* menu.
3. Select *Show Legacy Build Settings…*
4. Finder will open the folder where your legacy build settings are located.
5. Select the relevant build setting and compress it (`.zip`).
6. Go to the Builds Settings of your app on the Twixl platform.
7. Select 'import' to upload your build setting.
8. Upload it to the Twixl platform.
9. All done!

Registering an Apple Silicon M1 Mac

In order to use a Mac that uses the 'Apple Silicon' M1 processor for building apps, you need to make sure the device is registered on the Apple developer portal.

1. Find the Mac's device UDID

1. Open the "System Information" app (you can find this app in /Applications/Utilities or you can also access it by selecting 'About this Mac' from the Apple menu, then click 'System Report'.)
2. Under hardware, look for "Provisioning UDID" and copy it.

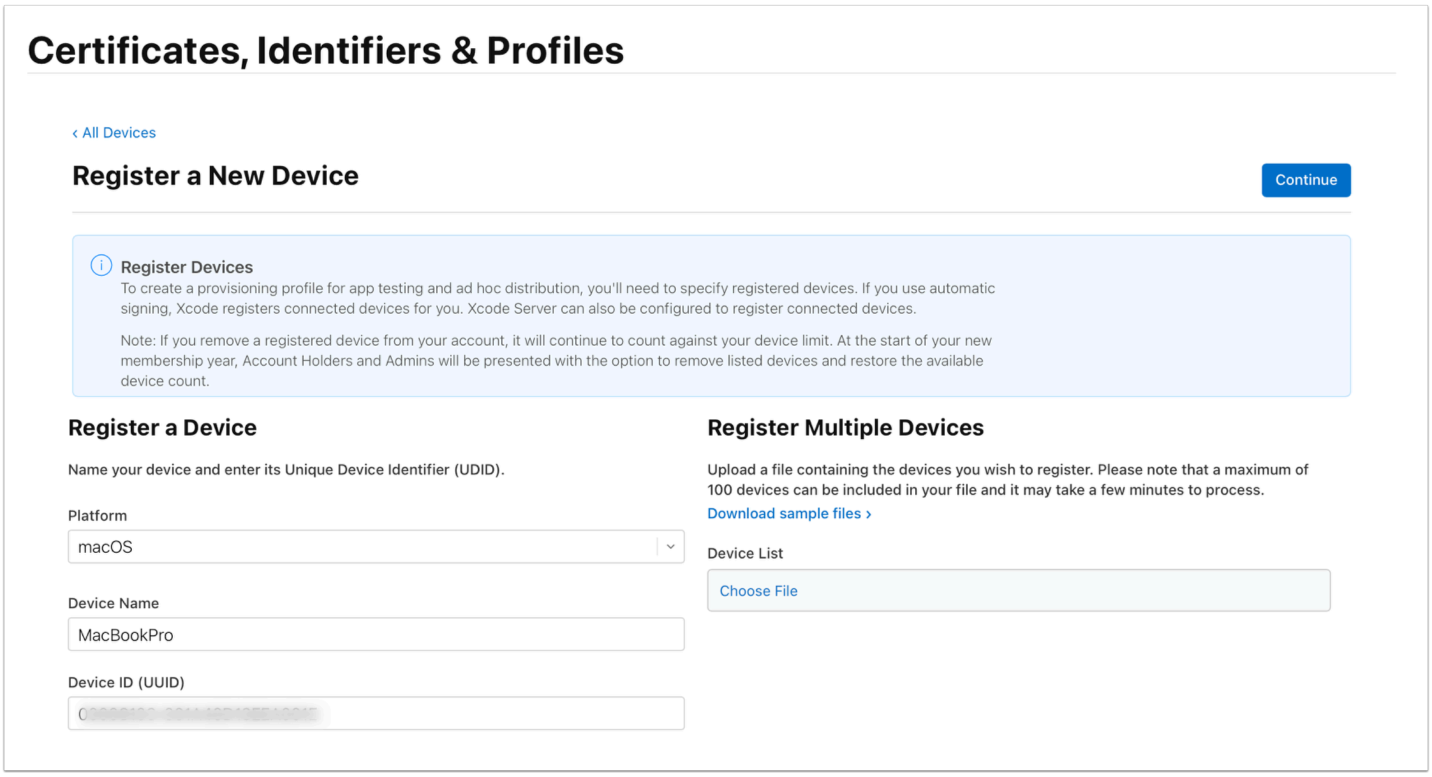


Hardware Overview:

```
Model Name:                 MacBook Pro
Model Identifier:           MacBookPro17,1
Chip:                       Apple M1
Total Number of Cores:      8 (4 performance and 4 efficiency)
Memory:                     16 GB
System Firmware Version:    6723.101.4
OS Loader Version:          6723.101.4
Serial Number (system):     [redacted]
Hardware UUID:              8A8B58B6-31E7-5FAC-B01C-4B6277E5A0E2
Provisioning UDID:          [redacted]
Activation Lock Status:     Enabled
```

2. Register your Mac on the Apple Developer Portal

1. Navigate to the ['Devices' list on the Developer Portal]
2. Click the '+' sign to add a device
3. Choose macOS as the platform
4. Add a name for your device and enter the UDID you just copied



Certificates, Identifiers & Profiles

‹ All Devices

Register a New Device

Continue

Register Devices

To create a provisioning profile for app testing and ad hoc distribution, you'll need to specify registered devices. If you use automatic signing, Xcode registers connected devices for you. Xcode Server can also be configured to register connected devices.

Note: If you remove a registered device from your account, it will continue to count against your device limit. At the start of your new membership year, Account Holders and Admins will be presented with the option to remove listed devices and restore the available device count.

Register a Device

Name your device and enter its Unique Device Identifier (UDID).

Platform: macOS

Device Name: MacBookPro

Device ID (UUID): [redacted]

Register Multiple Devices

Upload a file containing the devices you wish to register. Please note that a maximum of 100 devices can be included in your file and it may take a few minutes to process.

Download sample files ›

Device List: Choose File

This only needs to be done once. You are now ready to start using your M1 Mac to build a Twixl app.

Platform Specific

iOS: Manage signing certificates for your apps

Apple built a security model for iOS that requires a combination of a security certificate and a private key in order to 'codesign' apps and be able to either install them on an iPad or iPhone outside of the App Store, or to submit these apps to the App Store.

Below are the steps required to obtain and install that combination. More detailed documentation about certificates, identifiers and provisioning profiles is available from Apple's developer portal.

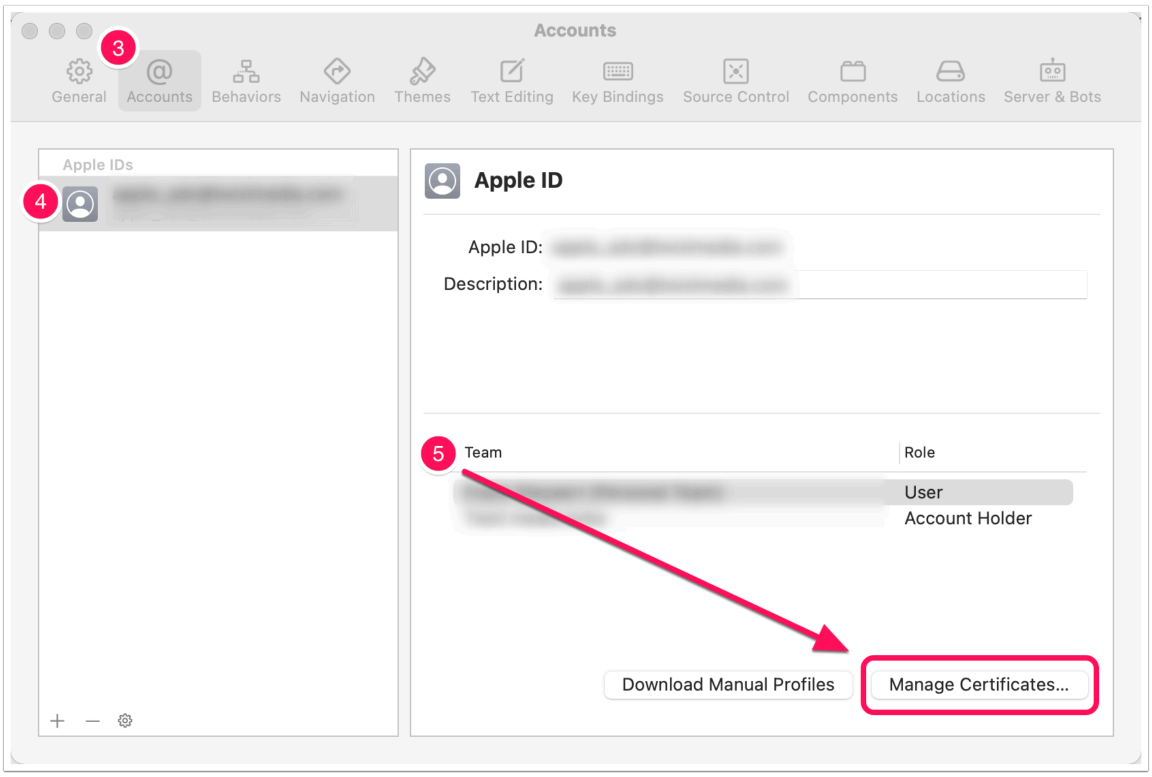
1. Obtain and install the necessary certificates

REQUIREMENTS:

Make sure you are logged in as an Apple Developer in Xcode. For more info on how to do that, see this article.

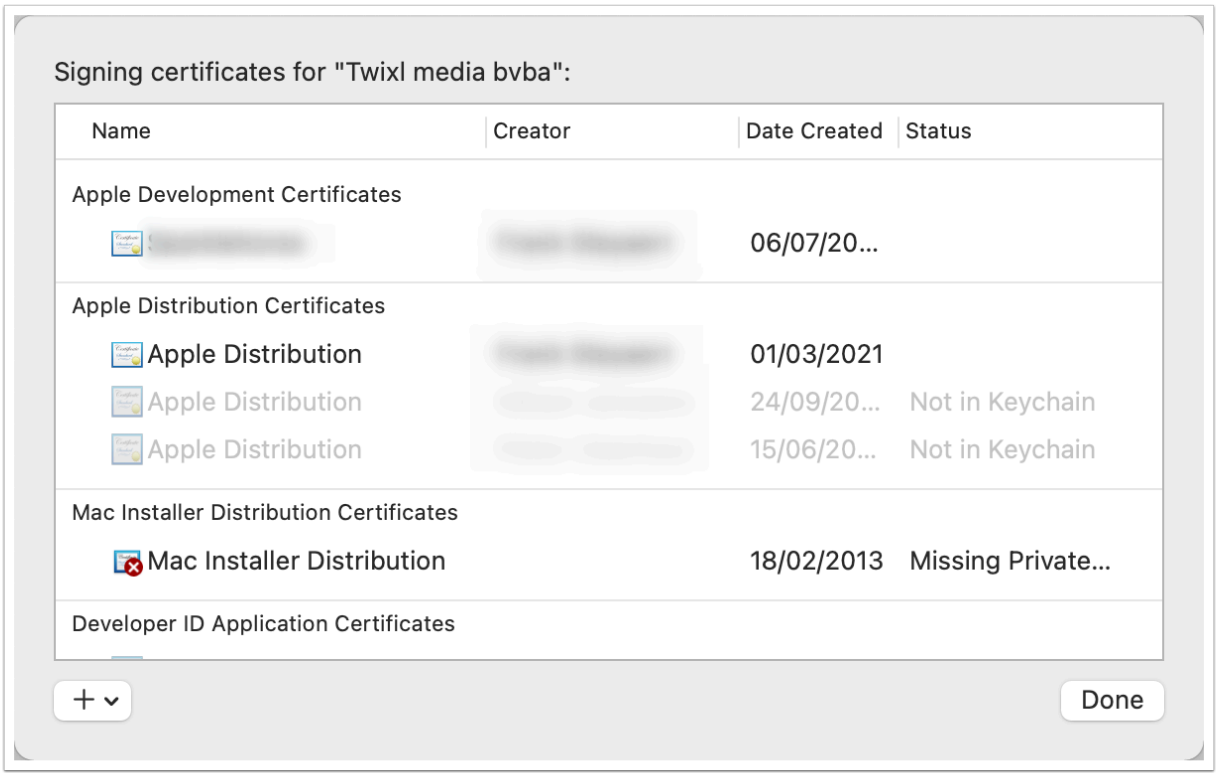
After you have logged in to Xcode, it's time to get the necessary certificates.

1. Launch *Xcode*
2. Go to the *Preferences...*
3. Select the *Account* tab
4. Select your **account**
5. Select the appropriate *Team* and click on *Manage Certificates...* (Note that Distribution Certificates belong to the team but only the Account Holder or Admin role can create Distribution Certificates. If you're enrolled as an individual, you are the Account Holder).
6. Click on the + icon and select the type of certificate you need, depending on the Build Type. For more info about Build Types, see this article.
7. Xcode will do the signing request, get and download the certificate and add it to your Keychain! We told you we made it easier!

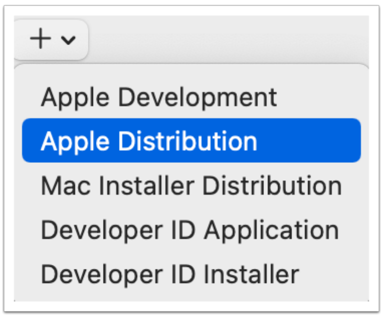


Apple ID:
Description:

Team	Role
	User
	Account Holder


Signing certificates for "Twixl media bvba":

Name	Creator	Date Created	Status
Apple Development Certificates			
		06/07/20...	
Apple Distribution Certificates			
Apple Distribution		01/03/2021	
Apple Distribution		24/09/20...	Not in Keychain
Apple Distribution		15/06/20...	Not in Keychain
Mac Installer Distribution Certificates			
Mac Installer Distribution		18/02/2013	Missing Private...
Developer ID Application Certificates			



IMPORTANT NOTE:

Twixl Publisher uses Development certificate and Development provisioning profiles for testing purposes, and a Distribution certificate and Distribution provisioning profile for Ad Hoc and App Store distribution.

2. For existing customers only: delete your old certificates

To avoid running into any issues while building your app, if you created (Twixl) apps before, you may have existing **iOS Developer or Distribution** certificates (you'll find these in the login keychain, under the section 'My Certificates'). Starting with Twixl Publisher 12, you need **Apple Developer or Distribution** certificates. Therefore we strongly advise you to delete your old certificates and start all over. To do this, you need to:

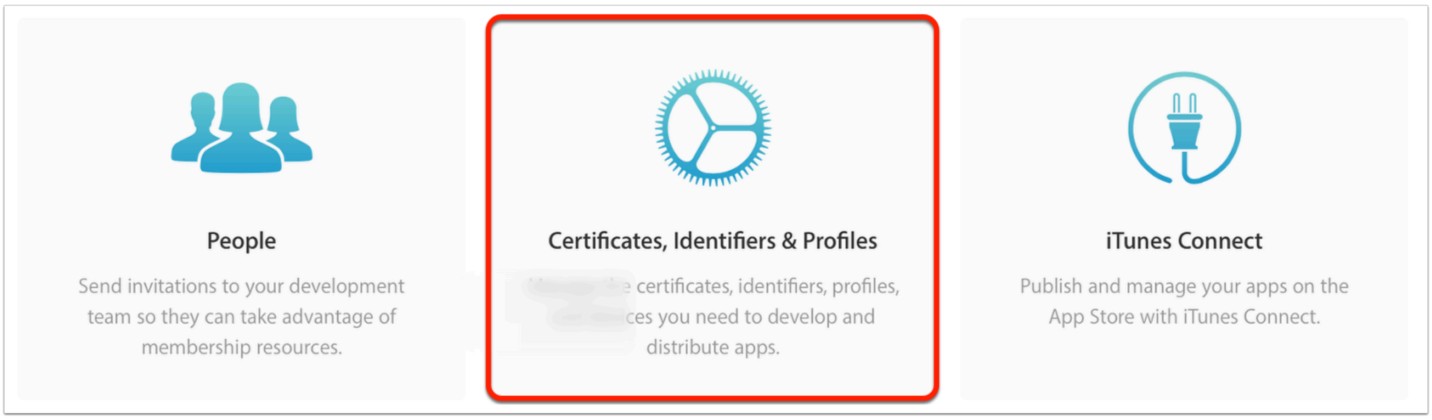
1. Delete all older iOS Developer and Distribution certificates in your Keychain Access app (found in /Applications/Utilities)
2. Revoke all your old iOS Developer and Distribution certificates on the Apple Developer Portal

Deleting and revoking your old certificates for **App Store apps** will not have any influence on your existing iOS apps. It will only influence the build process for new apps (and updates of existing apps).

For **Enterprise apps**, though, the consequence will be that apps built using that certificate will no longer start. If you need to extend the expiration date for a provisioning profile of an existing Enterprise app, you can do the following:

- Delete the provisioning profile in: ~/Library/MobileDevice/Provisioning Profiles/
- Create a new build, and a new profile will be created that will expire after one year

3. Create an App ID



People

Send invitations to your development team so they can take advantage of membership resources.

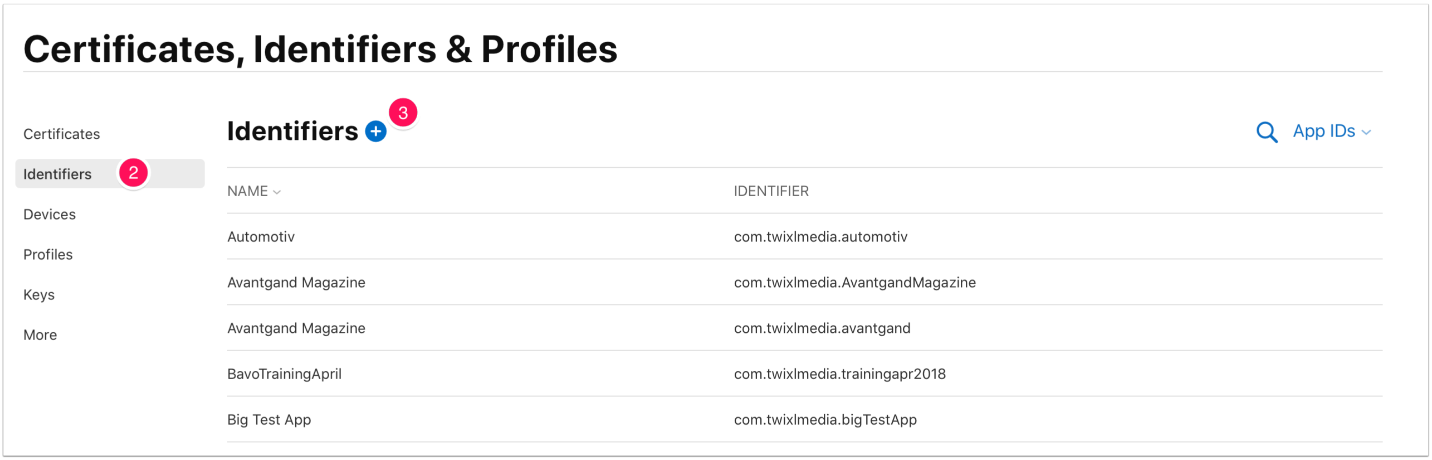
Certificates, Identifiers & Profiles

[illegible] certificates, identifiers, profiles, [illegible]ces you need to develop and distribute apps.

iTunes Connect

Publish and manage your apps on the App Store with iTunes Connect.

1. Login to Apple's Developer Member Center, then select **Certificates, Identifiers & Profiles**.
2. In the left column, click on **Identifiers**.
3. Next, click on + to add an Identifier
4. Choose **App IDs**
5. Choose **App**
6. Enter a **Description**, a **Bundle Identifier** (type **Explicit**). As an option you can activate **Push Notifications** (see this Chapter for more info).



Certificates, Identifiers & Profiles

NAME	IDENTIFIER
Automotiv	com.twixlmedia.automotiv
Avantgand Magazine	com.twixlmedia.AvantgandMagazine
Avantgand Magazine	com.twixlmedia.avantgand
BavoTrainingApril	com.twixlmedia.trainingapr2018
Big Test App	com.twixlmedia.bigTestApp

ABOUT AN APP ID

- An App ID consists of a unique 10 character "Bundle Seed ID" prefix generated by Apple and a "Bundle Identifier".
- The recommended practice is to use a reverse-domain name style string for the "Bundle Identifier" portion of the App ID, e.g. `com.yourdomain.yourapp`.

4. Assign the devices

Now you need to assign the devices you want to use for your Ad Hoc builds.

Locate the Unique device ID in Xcode (click on "Serial Number" and the Identifier will be displayed - copy the UDID via cmd-C). Now add the UDID(s) to the 'Devices' section of the iOS Provisioning Portal.

IMPORTANT NOTE:

Adding these devices is only required for creating **Development** or **Ad Hoc** builds for internal use.

5. Building your app

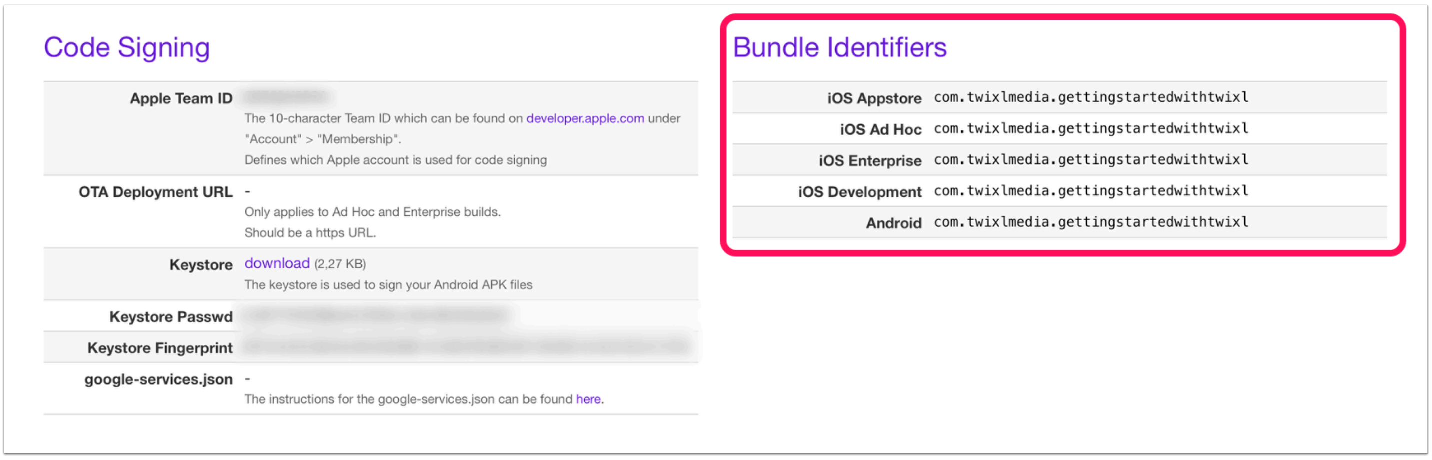
You are now ready to prepare the build of your Twixl App:

- Build Settings overview
- Building your Twixl app

iOS: App build types

Apple provides different types of app builds, depending on your use case.

In order to create any type of build, the **build settings** for your app on the platform must contain the relevant Bundle Identifier. For more details about build settings, please check this article.



Code Signing

Apple Team ID	[redacted] The 10-character Team ID which can be found on developer.apple.com under "Account" > "Membership". Defines which Apple account is used for code signing
OTA Deployment URL	- Only applies to Ad Hoc and Enterprise builds. Should be a https URL.
Keystore	download (2,27 KB) The keystore is used to sign your Android APK files
Keystore Passwd	[redacted]
Keystore Fingerprint	[redacted]
google-services.json	- The instructions for the google-services.json can be found here.

Bundle Identifiers

iOS Appstore	com.twixlmedia.gettingstartedwithtwixl
iOS Ad Hoc	com.twixlmedia.gettingstartedwithtwixl
iOS Enterprise	com.twixlmedia.gettingstartedwithtwixl
iOS Development	com.twixlmedia.gettingstartedwithtwixl
Android	com.twixlmedia.gettingstartedwithtwixl

1. Development build

This type of app always expects an app in **Development Mode** on the platform.

2. Ad Hoc build

iOS Developers enrolled in the standard iOS Developer program have the opportunity to distribute their application outside of the App Store on up to 100 devices. Use the Ad Hoc Bundle ID in the build setting of your app to create a build of your app to distribute internally. This type of app always requires an app in **Production Mode** on the platform.

3. Enterprise build

iOS Developers enrolled in the iOS Developer Enterprise program have the opportunity to distribute their application outside of the App Store and without the 100 devices limit. Use the **Enterprise** Bundle ID in the build setting to distribute a build of your app internally. This type of app always requires an app in **Production Mode** on the platform.

4. App Store build

Use the "App Store" Bundle ID in the build setting to create a build that you will be submitting to the App Store. This type of app always requires an app in **Production Mode** on the platform.

iPad: Twixl Apps and Split View

Split View is part of the multitasking experience available for iPads. This article explains the basics.

What is Split View?

For more info on how Split View works, we advise you tho read the following article (from Apple Support):

Use Multitasking on your iPad

How do Twixl Apps work with Split View?

Adding support for Split View has had one important consequence: orientation changes are handled fully automatically on iOS, so they can not be set manually to 'portrait' or 'landscape' only.

Requirements

Split View requires you to have a:

- recent iPad with a recent iOS version
- an app created with a recent version of Twixl Publisher

What is supported?

Twixl apps fully support all Content Items in Detail Mode. That means readers can use Split View while they are consuming the following types of Content Items:

- HTML Articles
- Twixl (InDesign-based) Articles
- PDF Content Items
- Movies
- Vimeo Movies
- YouTube Movies

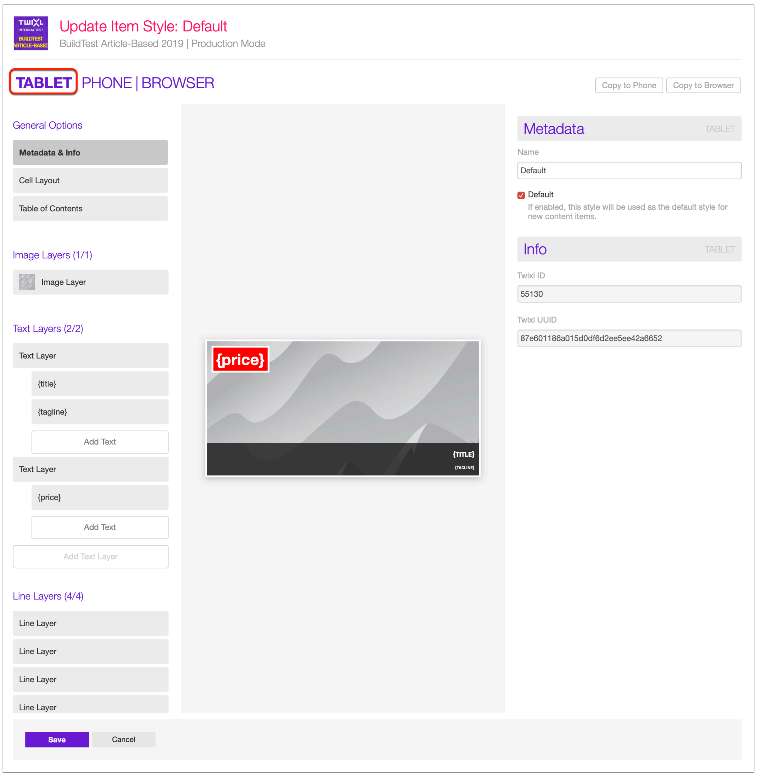
TIP CONCERNING TWIXL ARTICLES and PDF CONTENT ITEMS:

Twixl InDesign-based articles (and PDF Content Items) are all about pixel-perfect content. They are not responsive and they have a fixed layout. As a result, the following rules apply:

- Twixl Articles are being scaled. We strongly advise you to make use of Alternate Layouts.
- Letterboxing will occur.
- Depending on the size of the activated Split View, the Portrait or Landscape renditions will be rendered.

What about the 'Browse Grids' ?

Currently the Browse Grid for Collections in Browse Mode will only show the Tablet settings of your Item Styles. Be aware that this can result in strange behaviour when the activated Split View uses 1/4 of your iPad screen, in other words: when the available screen estate resembles a smartphone aspect ratio.


Update Item Style: Default
BuildTest Article-Based 2019 | Production Mode

Metadata (TABLET)
Name: Default
✓ Default — If enabled, this style will be used as the default style for new content items.

Info (TABLET)
Twixl ID: 55130
Twixl UUID: 87e601186a015d0df6d2ee5ee42a6652

Related Content

- Working with Content Items
- Working with Collections

- Application Options
- Styling: Item Styles

Android: About the Keystore

IMPORTANT INFO ABOUT THE KEYSTORE

As of **Twixl Publisher 15.4**, Twixl Android apps will now be **Android App Bundles** (with extension .aab instead of previously .apk). This requires that you use the new Google Play App Signing method in order to publish these .aab files.

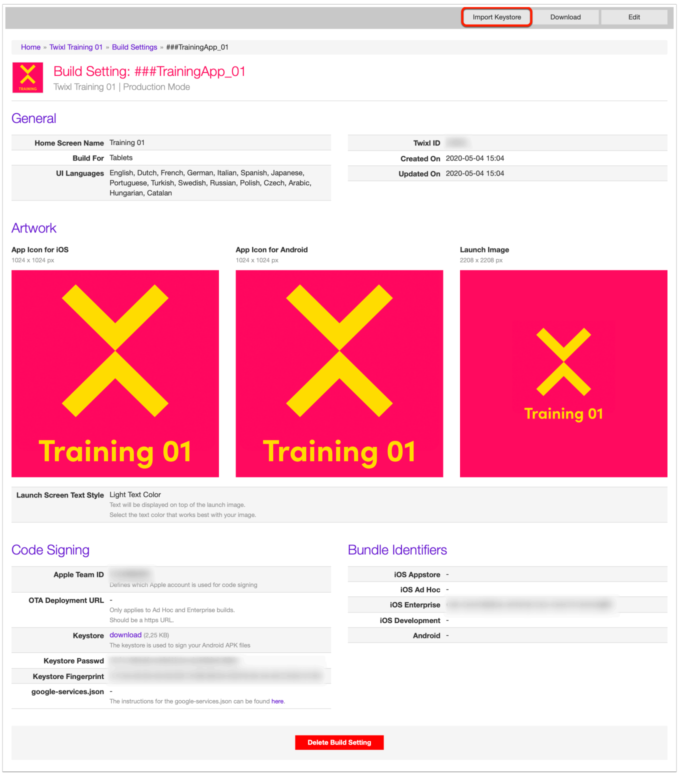
When you publish an app in the Google Play store, Google expects that app to be signed with a certificate. Before Twixl Publisher 15.4, when you created an Android app, we took care of creating this "**keystore**" for you in the background. Every update of your app had to be signed using the same keystore information (the app's "fingerprint" so to speak), otherwise it would not be accepted on Google Play. In this situation, if you ever lost your keystore, you needed to publish a new app with a new package name, and could no longer update your existing app.

Google Play App Signing solves that problem (requires **Twixl Publisher 15.4** or higher). If you lose your keystore or think it may be compromised, Google Play App Signing makes it possible to request a reset to your upload key, so you can keep updating your app.

Migrating your Google Play app from another platform, such as Adobe DPS, etc.

If you are moving to Twixl Publisher from a platform like Adobe DPS, you will have created an Android keystore using the Keytool via terminal, according to the instructions outlined in Adobe's documentation.

During that process you generated a keystore with an extension `.p12`. In order to update your existing Adobe DPS app to an app created with Twixl Publisher, you can use the 'Import' option in the Code Signing tab of the build settings for that app. When importing the `.p12` keystore, you will also need to provide the password you used to create it. For more info on how to manage your Build Settings on the Twixl platform, see: Build Settings overview



Home » Twixl Training 01 » Build Settings » ###TrainingApp_01

Build Setting: ###TrainingApp_01

Twixl Training 01 | Production Mode

General

Home Screen Name	Training 01
Build For	Tablets
UI Languages	English, Dutch, French, German, Italian, Spanish, Japanese, Portuguese, Turkish, Swedish, Russian, Polish, Czech, Arabic, Hungarian, Catalan

Twixl ID	[redacted]
Created On	2020-05-04 15:04
Updated On	2020-05-04 15:04

Artwork

App Icon for iOS 1024 x 1024 px

App Icon for Android 1024 x 1024 px

Launch Image 2208 x 2208 px

Launch Screen Text Style	Light Text Color Text will be displayed on top of the launch image. Select the text color that works best with your image.

Code Signing

Apple Team ID	[redacted] Defines which Apple account is used for code signing
OTA Deployment URL	- Only applies to Ad Hoc and Enterprise builds. Should be a https URL.
Keystore	download (2,25 KB) The keystore is used to sign your Android APK files
Keystore Passwd	[redacted]
Keystore Fingerprint	[redacted]
google-services.json	- The instructions for the google-services.json can be found here.

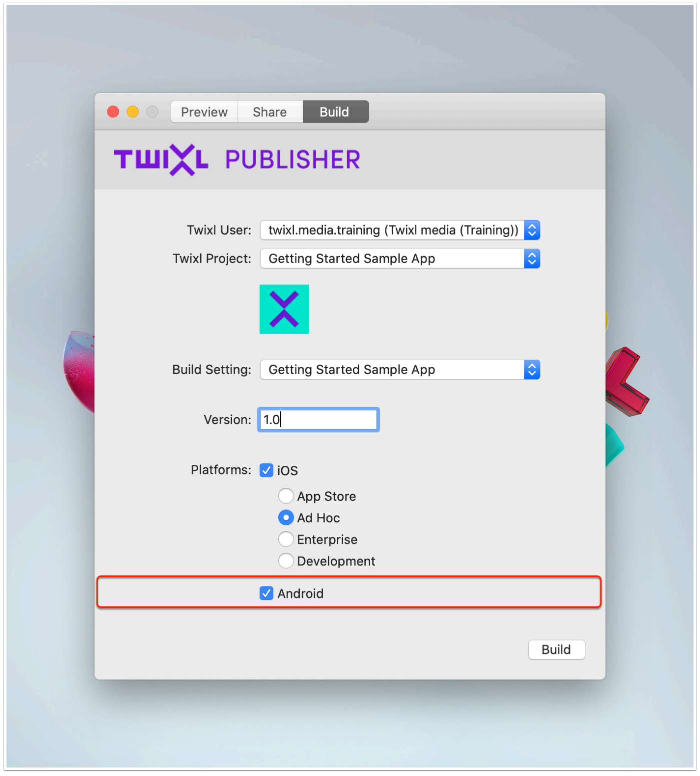
Bundle Identifiers

iOS Appstore	-
iOS Ad Hoc	-
iOS Enterprise	[redacted]
iOS Development	-
Android	-

Android: App build types

1 type: Android build

There is only one build type for Android. That build type can be used for testing, deployment to Google Play and for In-House deployment.



Twixl User: twixl.media.training (Twixl media (Training))
Twixl Project: Getting Started Sample App
Build Setting: Getting Started Sample App
Version: 1.0
Platforms: ✓ iOS
( ) App Store
[✓ Ad Hoc]
( ) Enterprise
( ) Development
✓ Android