Build your app

Table of Contents

Building Apps

Getting started to build your app

Once your app design has been setup and content has been added to the Twixl platform, you are ready to build your app and upload it to the stores. This Getting Started article will guide you through the different steps to distribute your app.

Distributing apps on Apple's App Store and Google Play Store is a bit different as both stores have different requirements.

- For the App Store: make sure you have an Apple Developer Account and your signing certificates.
- For the Play Store: make sure you register for Google Play App Signing.

To build apps with an Apple Silicon Mac, first register your device on the Apple Developer Portal.

The Twixl building process is handled via 2 different Twixl components:

- The Twixl platform
- The Twixl Publisher macOS application

The Twixl platform

In the Twixl platform, you will need to create a Build Setting where you will define your app icon, launch image; enter your bundle ID(s), Apple Team ID and other settings (e.g. if you want to use Push Notifications, Google Analytics...).

Check Twixl platform > Menu > Build > Build Settings

The Twixl macOS App

Once you have defined a build setting on the Twixl platform, you can create builds via the Twixl Publisher macOS Application. A folder will be created on your Mac with 2 subfolders (iOS and/or Android, depending on your selection) containing the builds to upload to the Stores.

The macOS App also allows you to preview and share InDesign content on your device or in a simulator.

> ⚠ You can only build Twixl apps on a Mac as creating iOS builds requires the use of Xcode, Apple's Integrated Development Environment. You'll need to sure you have Xcode installed and set up on your computer (but don't worry, you don't need to be a programmer).

Next Steps

- Deploying your iOS app in App Store
- Deploying your Android app in Google Play Store

> ℹ Of course, you can also choose to only distribute your apps in-house, both for iOS and Android.

Twixl platform: Build Settings

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.

1. 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 it. 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.

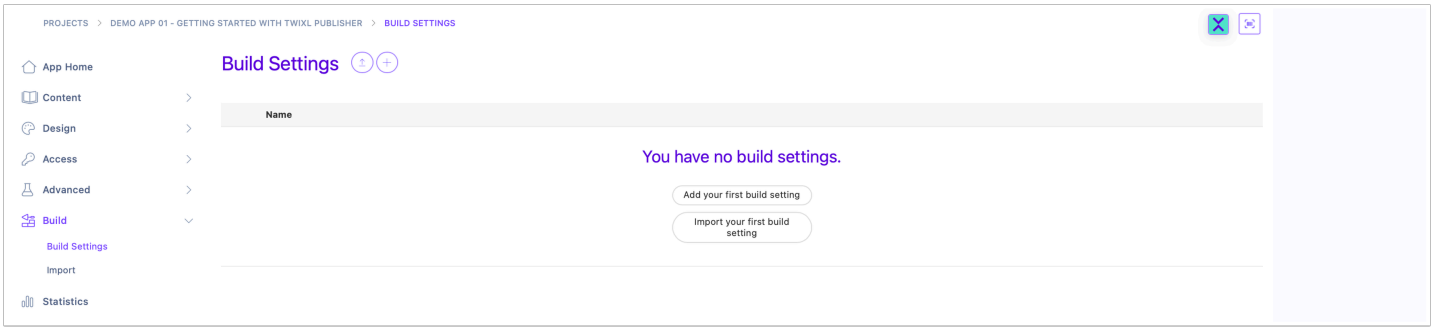
E.g. a build setting can contain the following info: which user interface languages are supported, the artwork (app icon, etc…), code signing information, etc.

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.

Whenever you make changes to any of the Build Settings, this will require that you create a new build of your app.

2. How to create a new build setting

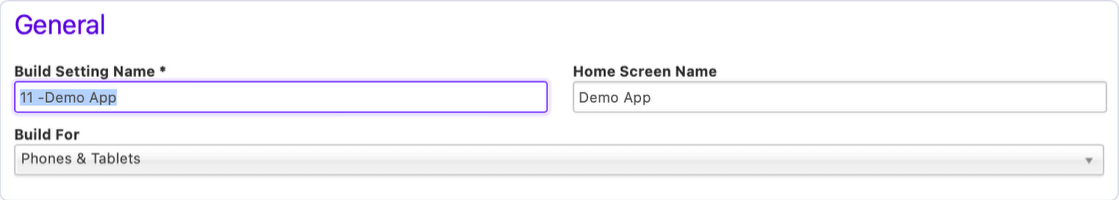
In the left menu bar > Click Build > Select Build Settings



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

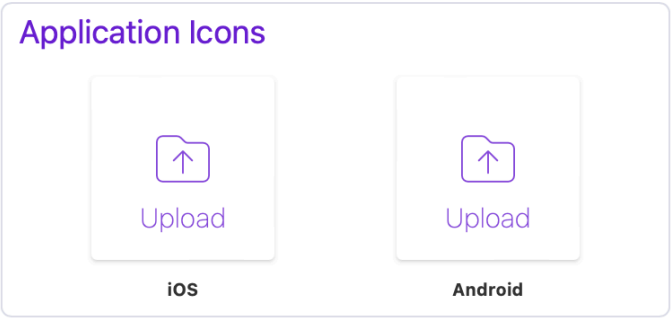
1. Click the '+' icon
2. Or click 'Add your first build setting'

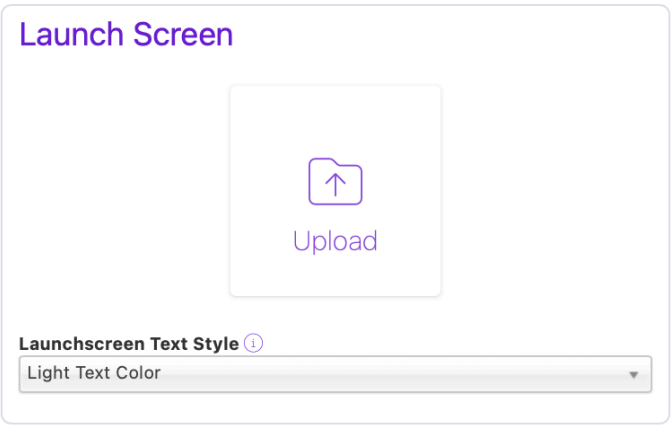
2.1 General



- **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?

2.2 Artwork

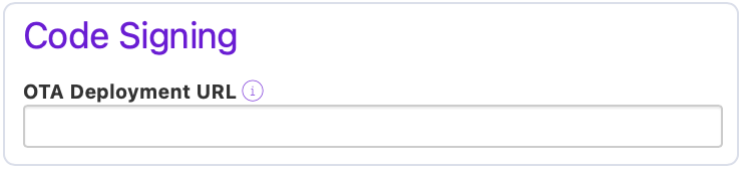




- **App Icon:** You can choose a different icon for iOS and Android
- **Launch Image:** The launch image of your app - see also this article.
- **Launch screen Text Style:** Text will be displayed on top of the launch image. Select the text color that works best with your image.

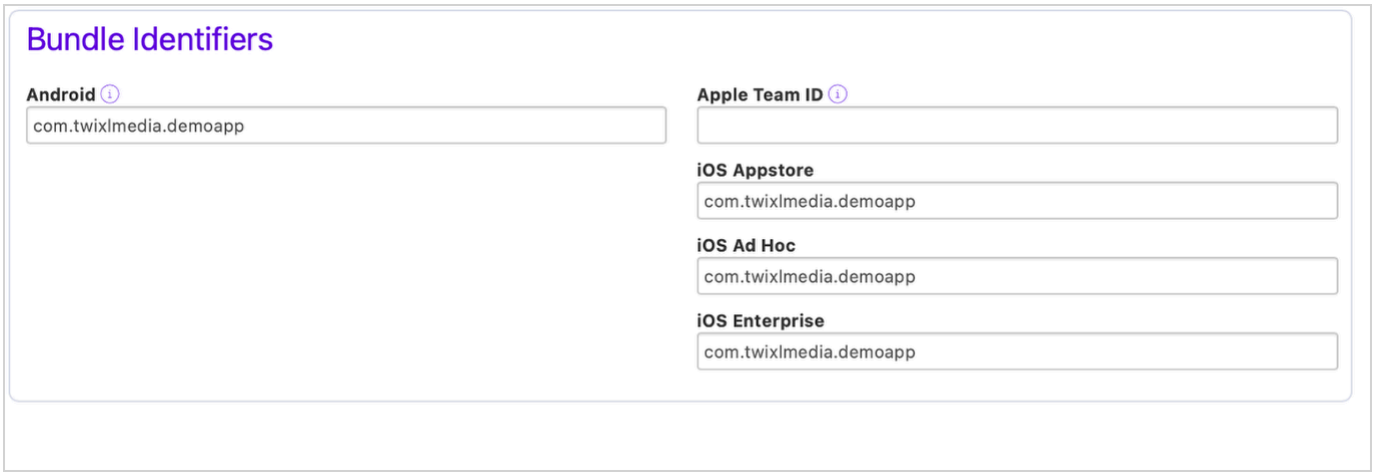
Especially for Android devices, creating an App icon can be confusing as the result might show something different than intended. This is because a lot of manufacturers of mobile Android devices create their own UI on top of the Android UI. If you want to be certain your design fits most devices, check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

2.3 Code Signing

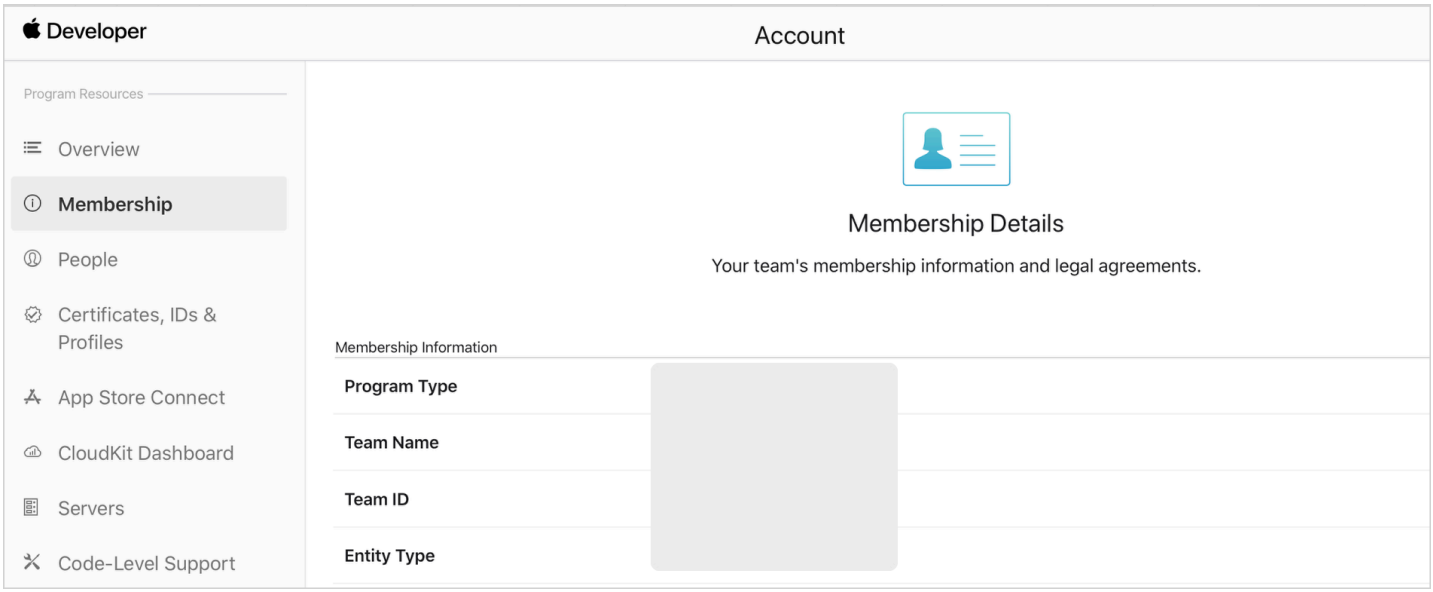


- **OTA Deployment URL:** Only applies to Ad Hoc and Enterprise builds. Should be a `https` URL. See this KB-article for more info.

2.4 Bundle Identifiers

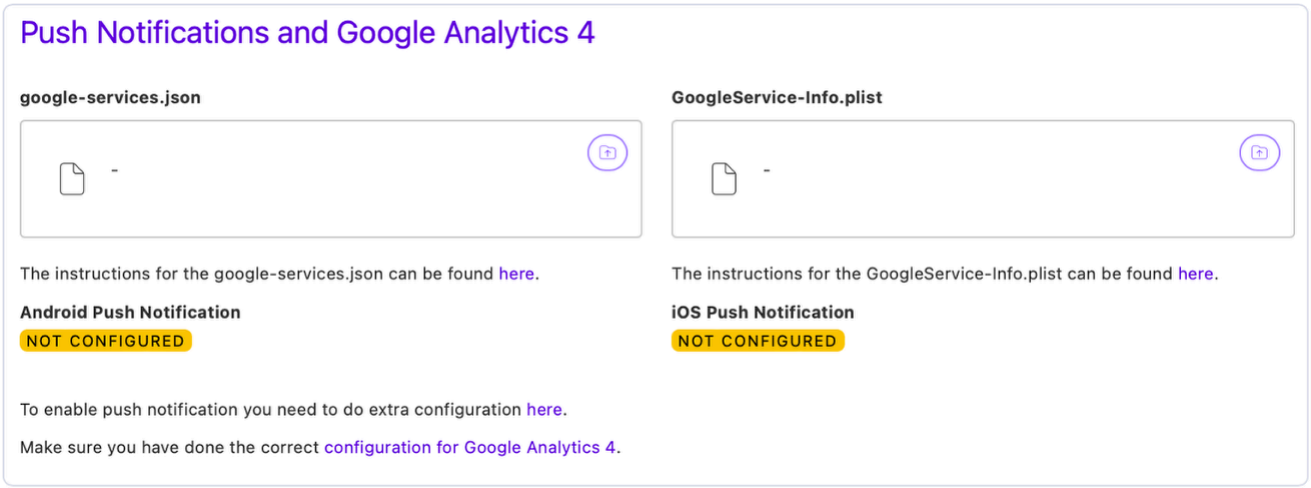


- **Apple Team ID:** Required to code sign and build your app. Navigate to https://developer.apple.com/account and click on *Membership* in the left column, copy the *Team ID,* a 10-character code, and paste it in the Team ID field.



- 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.

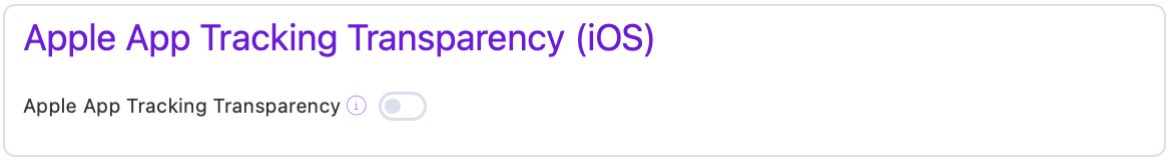
2.5 Push Notifications and Google Analytics



- **google-services.json:** Used for setting up Push Notifications on Android devices. See this article for more info.
- **GoogleService-info.plist:** Contains all of the information required by the Firebase iOS SDK to connect to your Firebase project.
- Both Android and iOS push notification statuses will change as one has completed the configuration.

By enabling Apple App Tracking Transparency via the slider, a list of languages opens up. The languages that are mentioned are also the standard languages that are supported in the UI of the device for all interface messages (error messages, download messages,...). In these fields you can add information about App Tracking Transparency in the language of your users. English always needs to be included as mandatory.

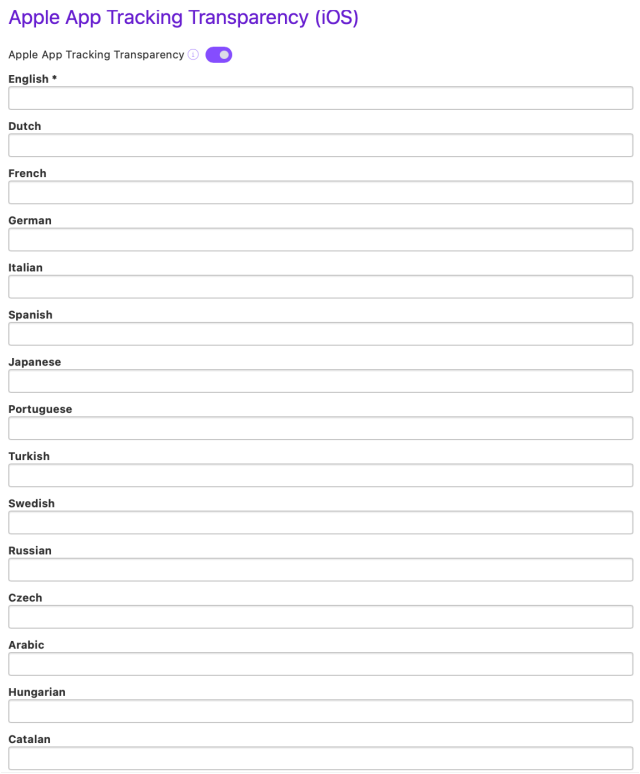
2.6 Apple App Tracking Transparency



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.

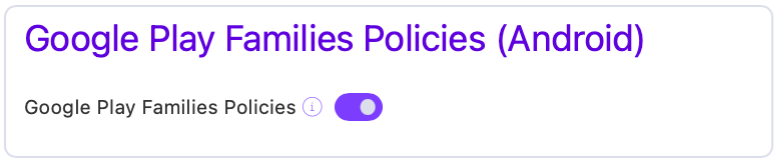
What happens when users select 'Do not track' ?

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.

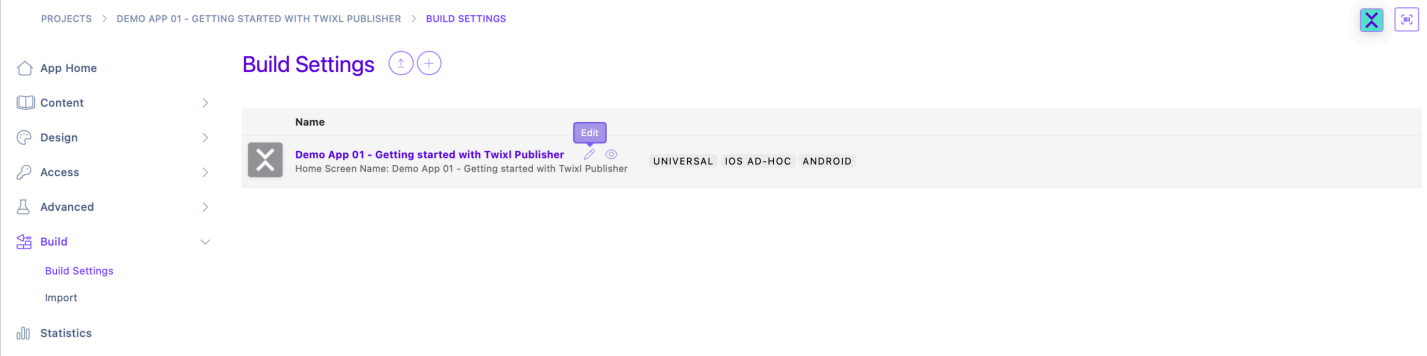


2.7 Google Play Families Policies

Google introduced Google Play Families Policies for apps that target children as their main audience. These types of app will be required to refrain from gathering analytics or even send out push notifications to devices of minors. By enabling Google Play Families Policies in the Build Setting, you adhere to this requirement. This also means that no analytics will be gathered from Android devices and you can only send out push notifications to iOS users.



3. How to edit an existing Build Setting



Hover over the Build Setting title and click on the 'edit' icon.
You can now change the fields you want to update and click 'save' when you're done.

4. 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!

MacOS App: Building your Twixl app

With the Twixl macOS App you can start building the app you have prepared and designed on the Twixl Platform. You can also preview your app or invite someone else to preview it.

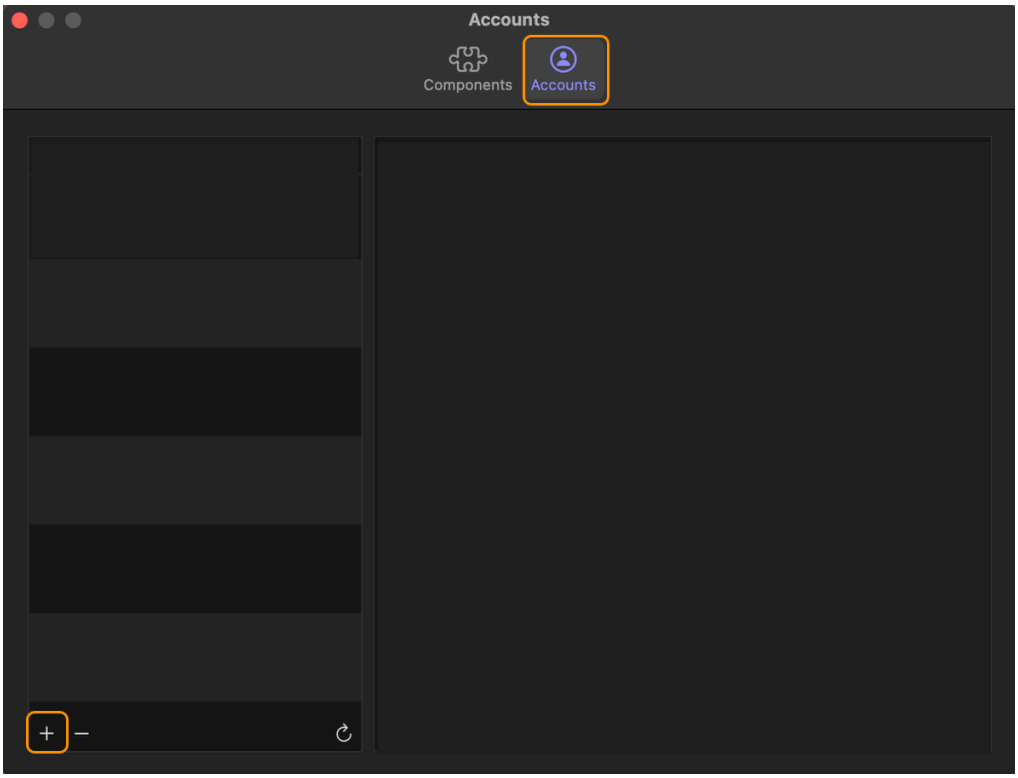
The steps below describe how to use the Twixl macOS App. We start by preparing the Twixl macOS App and then create your first build. You can also use the Twixl macOS App to preview and share InDesign .article and .publication files.

Set up the Twixl macOS App

Start by downloading the Twixl macOS App here.

1. Add your Twixl Account

- Open the Twixl macOS app > Twixl Publisher > Settings
- Select Accounts > Click on the + Icon > Fill in your Twixl platform credentials and *Sign In*



In order to build and preview Twixl apps via the macOS App, you need to have an account on the Twixl platform, and your app needs to be a 'production' app. There are 2 types of Twixl user account roles that will work:

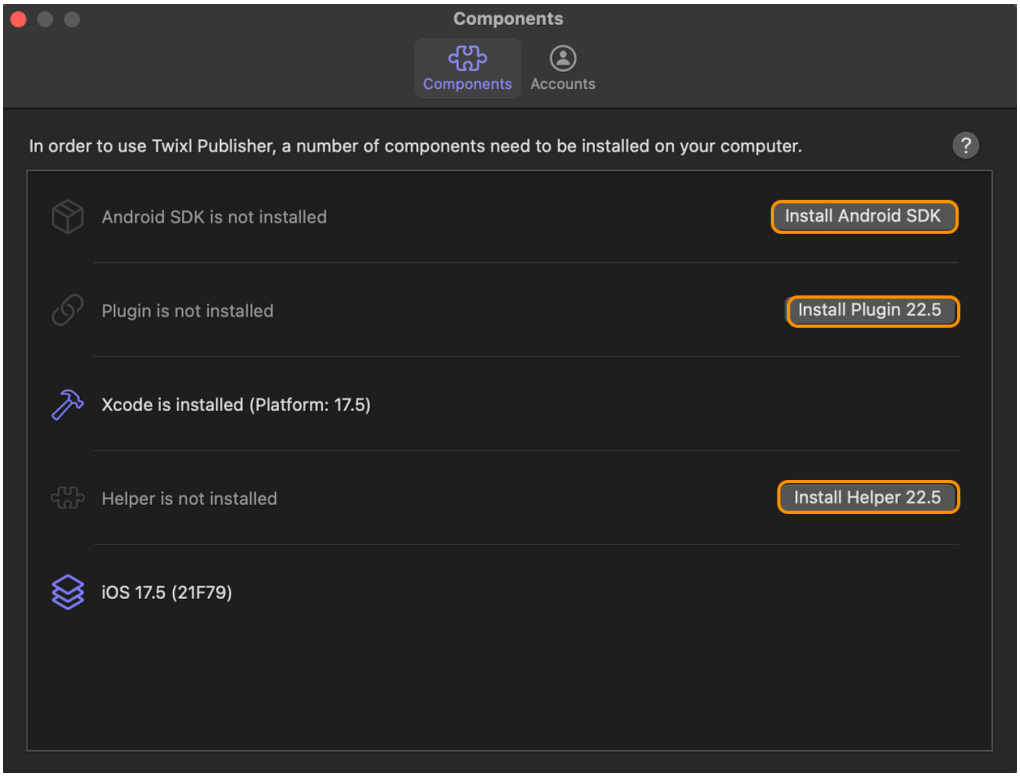
- Administrator
- App Administrator

For more info on how to create accounts on the Twixl platform, see this Help article.

2. Set up Components

You can access the components pop-up via:

- Twixl Publisher > Settings > Components
- Or via the warning at the bottom left of the window that appears if something is missing to build your app.



The macOS app lets you manage extra components that may need to be installed:

- **Twixl Publisher plugin for Adobe InDesign:** only required if you intend to create InDesign-based content with interactive elements
- **Twixl Publisher Helper:** required to export and preview InDesign-based content, and also required to build and preview Twixl Apps
- **Xcode:** required to build apps

- **iOS SDK**: required to build iOS apps
- **Android SDK:** required to build Android apps

INSTALL ALL COMPONENTS:

To avoid problems and unexpected behaviour, we strongly advise you to install all available components.

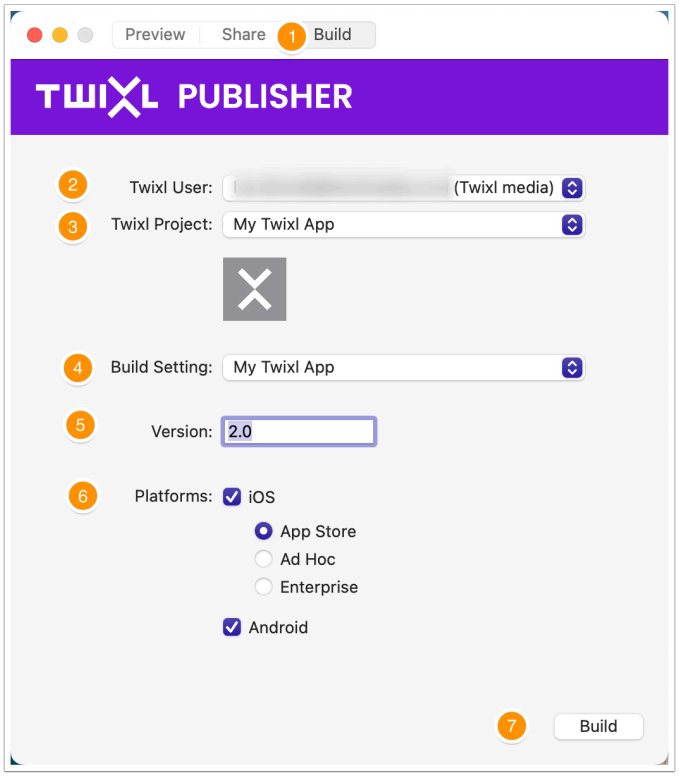
REQUIREMENTS:

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

- 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 (only production apps will appear here)
4. Choose the correct **Build Setting**
5. Enter a **Version Number**
6. Choose whether you want to build an **iOS** and/or **Android** app. More information about the different iOS platforms.
7. Click **Build**



Select where the files will be saved and wait for the process to finish.

For Android apps, the following files will be generated:

- the application itself, in two formats, i.e. as an .aab (Android App Bundle) and as an .apk
- 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)

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`

Next steps

- Deploy your iOS build to App Store
- Deploy your Android build to Play Store

Or

- Deploy your iOS build in-house
- Deploy your Android build in-house

Distributing for iOS

Deploying an iOS app in the App Store

Upload your app to App Store Connect

Submit in App Store Connect

Once you have created your App Store build using the Twixl Publisher app, take the following steps to submit your .ipa file to Apple's App Store Connect portal:

IMPORTANT NOTE:

For more details about the process of adding an app to App Store Connect, making it a free or a paid app to download, creating in-app purchases or subscriptions, you may want to refer to Apple's own App Store Connect Developer Guide. It can be accessed from the home page of the portal.

1. Go to the portal

In App Store Connect, go to the detail window of your app (assuming you went already to the process of creating an App).

2. Prepare for upload

1. Go to the "Versions" part of the application details and click on "View Details" underneath "[App Name] 1.0".
2. In the upper right corner, click the "Ready to upload binary" button.

3. Add artwork

Check the latest Apple requirements for your screenshots here.

4. Upload your app

1. Download & Install the **Transporter** tool from Apple.
2. Login with your App Store Connect credentials.

3. Now follow the instructions to upload your .ipa file.

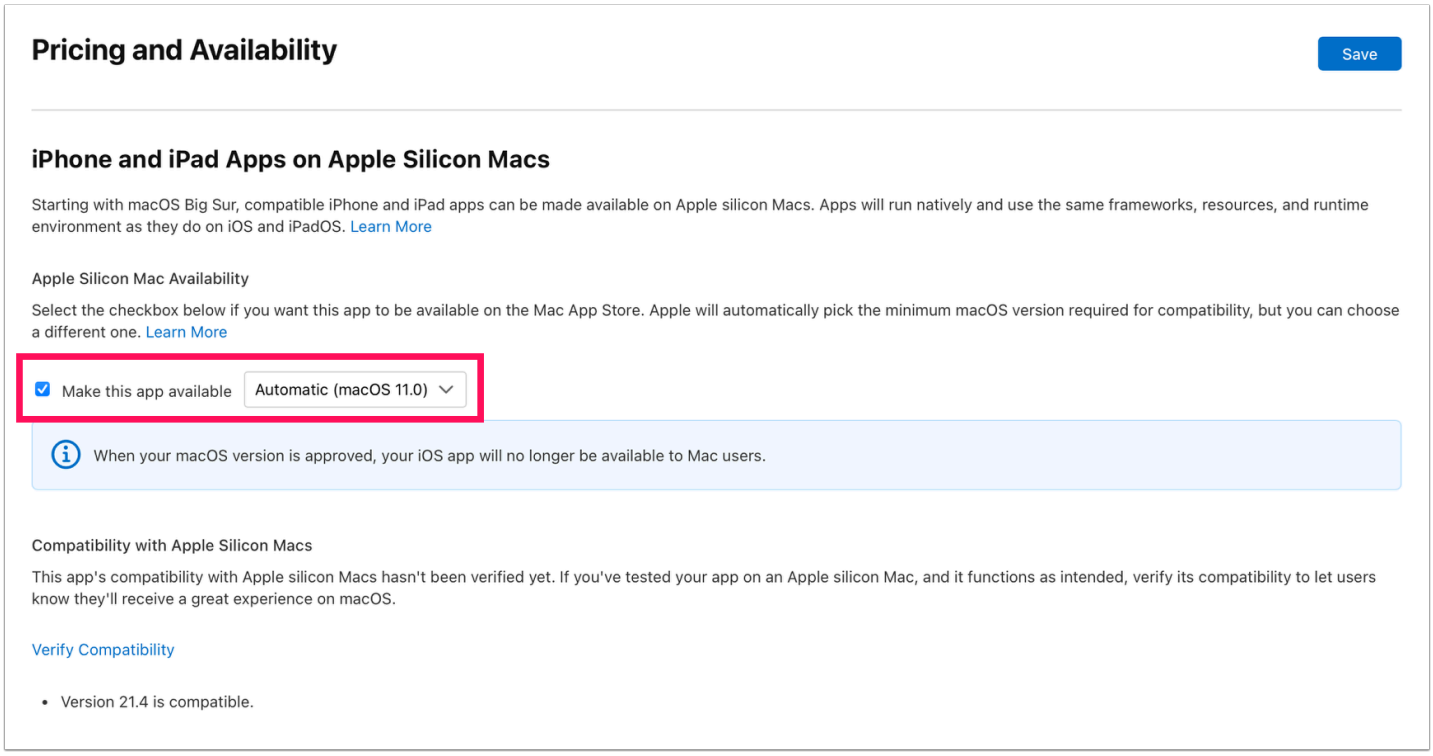
5. Submit for review

To add an App for review we need to select a binary that we uploaded, add it for review, and then submit for review. The procedure is well explained on Apple's Developer site.

Now wait for the review team to approve your app. Note that this may take anywhere between one to several days.

6. Making your iOS app available on Apple Silicon (M1/M2/M3) Macs

If you want to make your Twixl app available in the Mac App Store, you can now do so without any modification whatsoever. Just check the option in the settings for your app on App Store Connect, under '*Pricing and Availability*'.



Getting your iOS app approved

We all know that getting an iOS app approved isn't always an easy process.

- It's our job to make sure that Apple approves Twixl apps on a technical level.
- It's the job of the publisher to make sure that Apple approves the general concept and functionality of the app.

This KB-article provides some tips & tricks for the latter.

This KB-article was also published on our Blog.

Setting the standard

You may wonder why it is sometimes complicated to get your app approved by Apple or why Apple gets involved all in what you are publishing in your app.

But you really value that your app can be distributed via the App Store, because you know it is a respected, secure and reliable platform.

One reason why the platform is so successful is because it has high quality standards. Users appreciate iOS apps because they work, because they are nice, because they add value!

Apple defined guidelines for apps to be published on their platform just because they want to guarantee a top quality products to their users.

High quality standards are part of their identity!

So if we accept there is a need for a third party to distribute our app, we have to accept they impose rules (guidelines) we have to comply with.

If we don’t want to comply or don’t want to publish on the App Store, we can just publish internally.

We use the word ‘rules', Apple uses ‘guidelines’, and it is indeed more about guidelines because the approval process is a human process.

An app you submit will run through some automated tests, but human reviewers will test the app and verify

- the user interface
- the user experience

- all the features and services you offer in your app

This is also the reason the same type of app might be approved in Spain and not in the US. Every team has its own agenda, approach and interpretation of the guidelines, which makes it sometimes frustrating, but the quality standard is what counts.

Basically, it's like you are a car manufacturer. You are building a car and you want that buyers of your car can drive it on public roads. For this you have to comply with legislation by country where you want your customers to drive your car.

So when you design your car you take into account those regulations. You don't just build your car and check afterwards how to comply with regulations.

If you want to create an app you want to distribute via the App Store, the approach needs to be similar.

Start by reading the App Store Review Guidelines and make sure that everything in your app complies with these guidelines. Our advice is also to examine the Human Interface Guidelines, this will help you with the design of the interface of your app, and help you define the concept.

IN THE CASE OF A REJECTED APP:

Keep in mind that once your app gets rejected, it may be more difficult to change Apple's mind when you resubmit. Some small changes will not be sufficient, so depending on the problem Apple refers to, it may require major changes.

Why your app might be rejected

Here's an overview of the most common reasons why your app can be rejected. Keep these in mind: better safe than sorry!

1. This is not an app! – Lack of valuable content

If your app just plays a song or video, displays a brochure or just one story, then it's considered too simple to justify it being an app. Apps shouldn't primarily be marketing materials, advertisements, web clippings, content aggregators, or a collection of links.

Apple says: "Apps that are simply a song or movie should be submitted to the iTunes Store. Apps that are simply a book should be submitted to the iBooks Store."

There is one type of app that can offer 'only' content. This is a news or magazine app. This is accepted by the App Store but any other type of app has to bring an extra value to the user. You can't just make an app of a brochure presenting your product. You could present different products, with extensive technical information, with interactive presentation features. You could also offer extra services in your app.

Always ask yourself why you would like to have this app on your phone/tablet and what it would take to make you use it frequently. If you don't, you'll understand why your app may be rejected.

2. Lacking standard functionality

This is about the same as the previous item. Mobile is an interactive medium. Users expect interactivity. Slide shows, video and audio content. You should add services, things users can do with your app, that make things easier for them, more convenient. Things unique to the medium.

3. Poor user interface – Bad user experience

Before defining the navigation and browse pages of your app, we invite you to first check Apple's Human Interface Guidelines. They'll provide a good baseline on how to design your user interface properly. Also understand that the term user interface is a broad concept. The layout of supporting materials related to content are also important.

Apple and your users place a high value on simple, refined, creative, well thought out interfaces. They take more work but are worth it. Apple sets a high standard. If your user interface is complex or less than very good, it may be rejected.

4. Website or app?

"Your app should include features, content, and a UI that elevate it beyond a repackaged website. If your app is not particularly useful, unique, or *app-like*, it doesn't belong in the App Store."

You are creating a native iOS app, not a web app. A native app is about offering an app-like behavior that supports the functionalities of the device. It is not about a click behavior of a website.

There is a huge difference between a web app and a native app. A web app offers a web experience. While you can create a web experience within a native app, Apple may reject this because there is no added value for having an app if it is 'just' a website.

That doesn't mean you can't have any web content in your app, on the contrary, you can combine both if it adds value, but you shouldn't create an app with only web content or an app that is just a copy of your website.

5. Repeated submission of similar apps

Imagine you create travel guides, and for each travel guide you create a separate app. Well, Apple will get back to you and ask you to combine all those guides in one app. You might think you'll get less exposure but on the contrary. Offering an answer to far more search requests you'll get a better ranking and it will be easier to create brand awareness.

If you have different similar apps or are submitting different similar apps, Apple will always ask you to combine them in one app because it just offers more value to your users and to yourself.

6. Privacy - Support - Permission

It's a clear requirement that all apps submitted to the App Store have a privacy policy in place in order to get approval.

You need to:

- provide a link to your privacy policy statement
- provide an explanation of your data retention policies
- enable users to withdraw consent to data collection
- include a direct link to Support along with contact information

It's very important that your app only requests permissions that it needs to function. This includes permission to send push notifications, to use the device camera, GPS, the user's location and more. If your app requires location services, it is important to explain why your app needs access to it.

7. Improper use of trademarks and logos

"Apps that contain false, fraudulent or misleading representations or use names or icons similar to other apps will be rejected."

Apple requires apps to be unique with their names, icons or interface and discourages imitation of any other apps.

In particular, Apple is very sensitive if you get tempted to use buttons or user interface components in your app that are theirs.

Also, don't mention other supported platforms: if you have an Android app, that's fine, but don't promote it in your iOS app.

8. Crashes, bugs and broken links

Apple doesn't take too kindly to apps that contain explicit bugs or cause full-out system failures. If your testing of the app shows an unstable performance and ongoing crashes, get that fixed first before submitting for review. Otherwise your app is almost guaranteed to be rejected.

Using the Twixl app, you can check all the functionalities of your app without having to build it first. That's the first and most convenient step. After that, we advise you to create an 'ad hoc' build and to test it with some colleagues, and preferably some external people that weren't involved in the development. If everything works fine, then you can submit your app to the App Store.

You can also have a beta testing team and define what needs to be tested. **It is recommended you submit an update of your app at least 4 times a year**, so it is important your testing process is clearly defined, so it can easily be repeated.

9. Placeholder content

One of the most frustrating things for anyone on the receiving end of an app and in this case for Apple's review team is to find placeholder content still in there. The same thing with demo content or beta content.

This is a sign that your mobile app is incomplete and wasn't ready to be submitted yet. It will also give reviewers an extra reason to dig deeper and maybe find more things wrong with it.

If you need to use such content during the development process, make sure it is extremely well recognizable during testing so you can replace it by the real content before submitting your app.

10. Hardware and software compatibility

Per Apple's guidelines, your app must work on all the latest systems; i.e. hardware and software. It is important that apps run on all types of iPads and iPhones, at least if your app is 'universal', so keep that in mind for your tests.

Apple also recommends apps support split-screen usage, which Twixl will offer with TP11.

Twixl will always make sure to support the latest Apple software and hardware, so this is more an issue for Twixl than it is for you, unless you forget to update your app regularly. If your app has not been updated for a long time, you could get into a situation where Apple will come back to you and ask you to update your app, and if you don't, it could even be removed from the store.

11. Linking with outside payment plans

If your app takes payments to unlock functionality or allow the user to download digital content, transactions must go through the official Apple in-app purchasing system.

This goes for both individual in-app purchases and subscriptions.

This rule also applies to web pages linked to from your app.

It makes sense that If you want to distribute your app in Apple's App Store, the least they'll ask is that any transaction in the app will make use of their system. They have a responsibility towards the buyer because they are the marketplace.

As a publisher you can still sell subscriptions in your web shop and work with entitlement to provide access to your content. But within your app you can't link to any external transaction platform.

12. Insufficient or misleading information provided

A major requirement for apps in the App Store is that they have the metadata. That means including screenshots, descriptions, etc.

The main problems that can occur with metadata are:

- incomplete submission form
- incorrect or insufficient description
- not providing a demo login account
- outdated contact information
- missing or incorrect imagery
- inappropriate rating
- unrelated keywords

In relation to the screenshots you need to submit at least 3 different sizes, iPhone 5,5", iPhone 6,5" and iPad 12,9". If you don't have the required devices to create all these screenshots, note that you can always make screenshots in the iOS simulator.

13. Apps that are data hogs will get rejected

Be very careful with the amount of data your users need to download before they can get access to your app. If it takes over 15 seconds, your app may be rejected.

Therefore you should never use our 'full offline mode' on apps you submit to the App Store.

Offline mode is for in-house apps, as you are then in an environment where you are fully in control.

Pay attention

If your app does get rejected, don't get discouraged and do try to understand Apple's motive. It'll often be in your own interest. Try to address the issue Apple is referring to, and resubmit your app, but be thorough in your work.

Everybody makes mistakes and even the most experienced developers have had their app rejected from time to time.

Out of our own experience, we can tell you that while you can try to make your case with Apple, it's not a good idea to start to argue with Apple review. You can ask for more information, but then it is up to you to work on the remarks that Apple provides.

Oh yes, we didn't mention Google and submitting apps to Google Play. Well, the process is about the same, but Google is less strict in controlling their guidelines than Apple is. This is also why the Google Play store is perceived as having slightly lower quality standards than Apple's App Store.

Useful links

General info about the Apple app review process:

- https://developer.apple.com/ios/submit/
- https://developer.apple.com/app-store/review/

Apple App Store Review guidelines:

https://developer.apple.com/app-store/review/guidelines/

iOS Design guidelines:

- https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes/

Apple App testing guide:

- https://developer.apple.com/library/archive/technotes/tn2431/_index.html

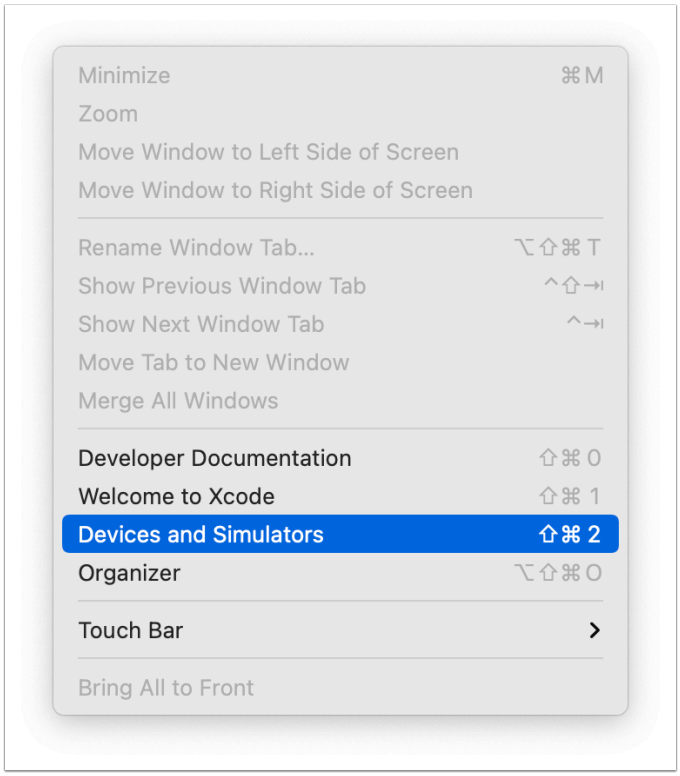
Apple App Store Guidelines and Resources:

- https://developer.apple.com/app-store/resources/

Deploying an iOS app in-house

How to get the .ipa file you created on an iPad or iPhone outside of the App Store.

1. Install from Xcode



Sometimes, mainly for troubleshooting purposes, you may want to install the `.ipa` file via Xcode.

1. Connect your iOS device via USB, then open Xcode
2. From the Window menu, select "Devices and Simulators"
3. Select your device from the list of devices, then use the "+" sign to install the `.ipa`.

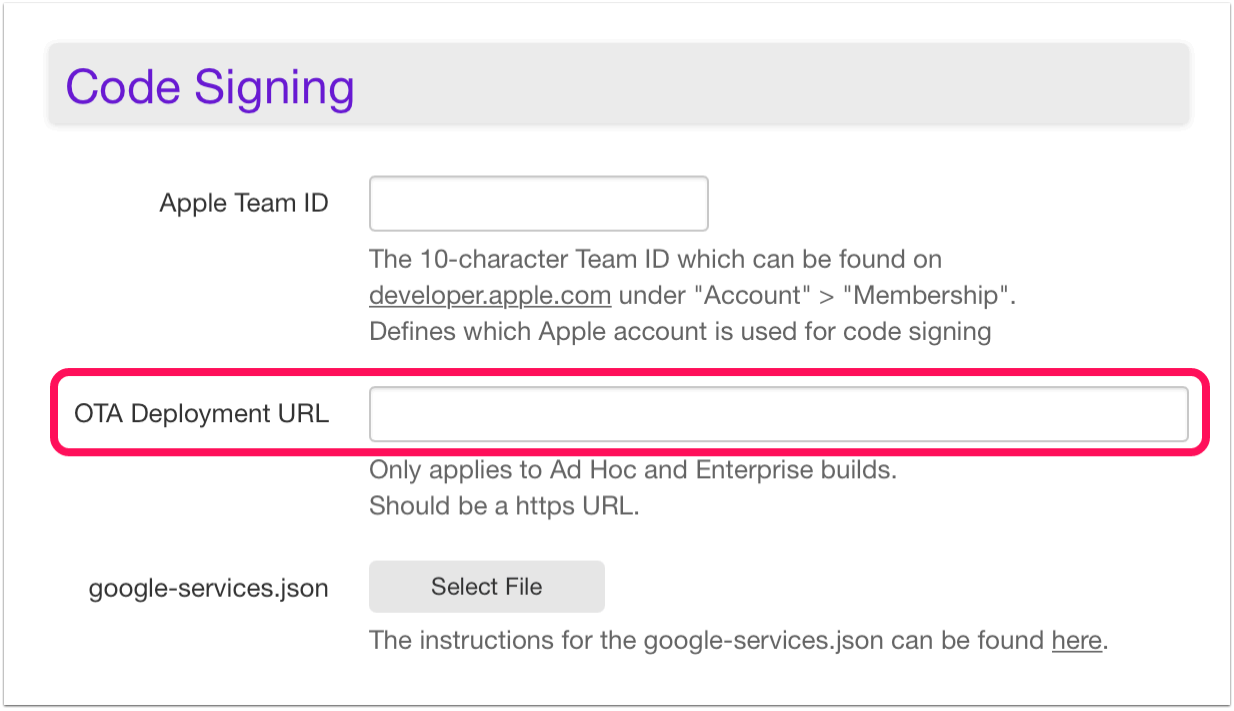
If something is wrong with the build or with your provisioning profile, you will usually get a more detailed error message.

2. Deploy the app over the air (Twixl built-in)

This method allows you to distribute the app more easily to your employees. The `.ipa` file will be installed on a secure web server, and users will be able to install the app directly on the device.

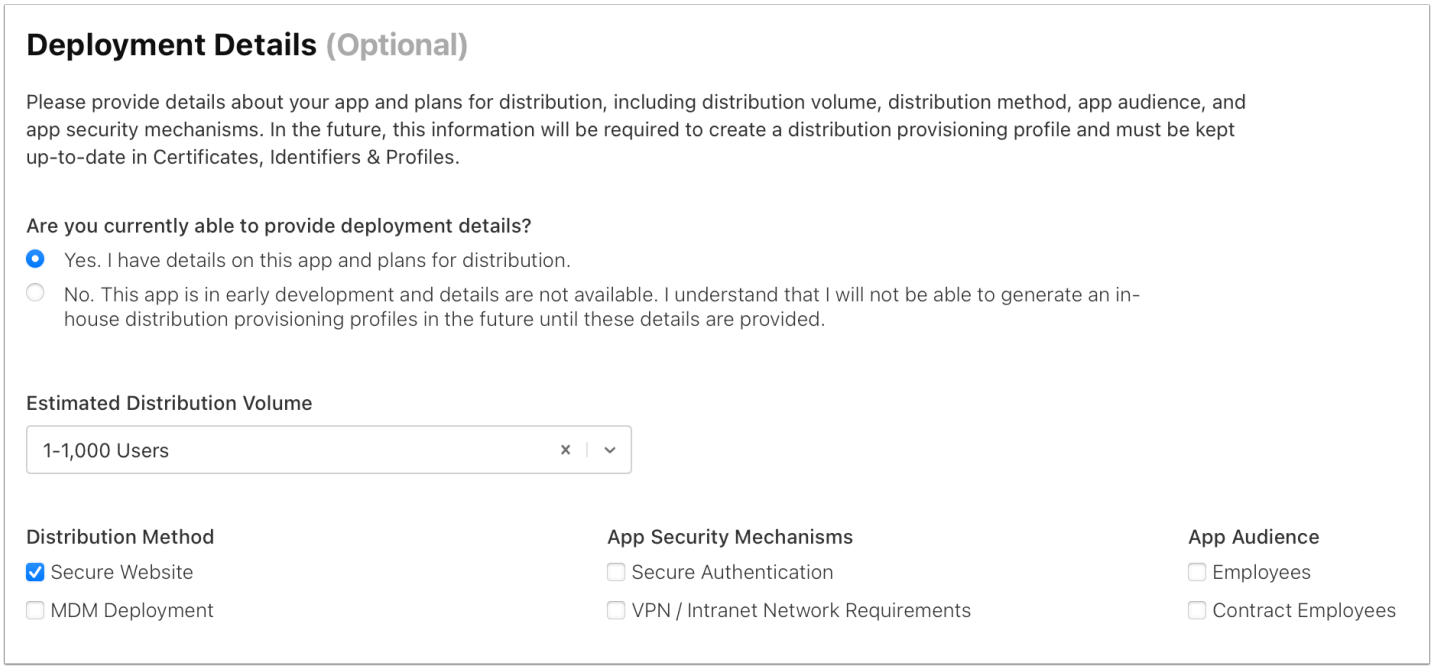
2.1. Deployment URL

Before you create a build of your app, enter the deployment URL in the "Code Signing" section of the build settings. The deployment URL is the location on your web server where you will be putting the app.



Due to a recent change by Apple, Ad Hoc builds now require the mobile device to run in developer mode to be able to install and launch the app.

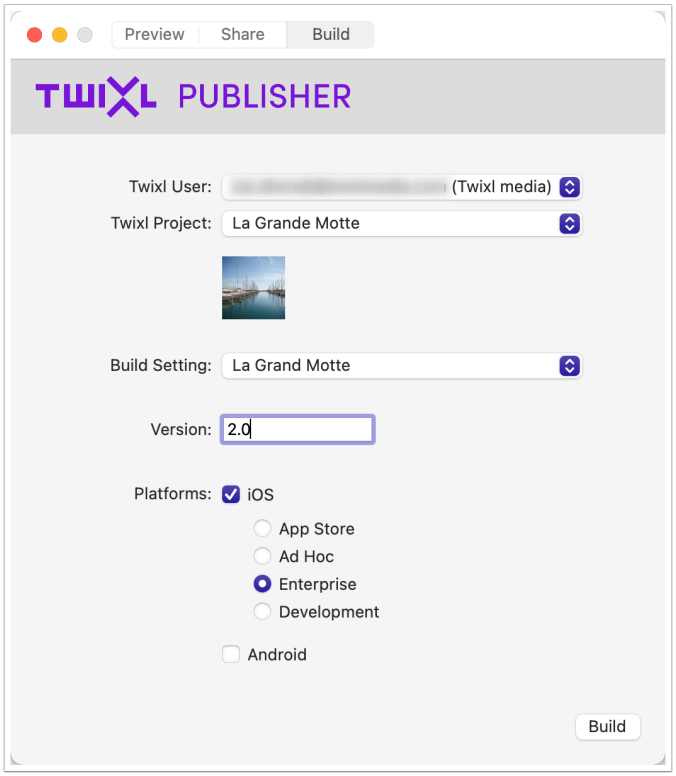
In the App ID Configuration on developer.apple.com, you'll need to check 'Secure Website' in the Deployment Details. Currently this information is optional, but this may become a requirement in the future.



IMPORTANT NOTE:

Apple requires the deployment URL to be a secure (https) URL, so you may need to get an **SSL security certificate** for your web server. The profile for the app also must be explicitly accepted first, before you can use that app on the iPad or iPhone. First install the Enterprise app, then navigate to Settings -> General -> Profiles -> Enterprise app. There you will be able to trust the certificate for the company you just installed the app from.

2.2. Create a build



In the Build Setting for your app on the platform, make sure you have entered the OTA URL first (if you intend to distribute using the Twixl built-in method).

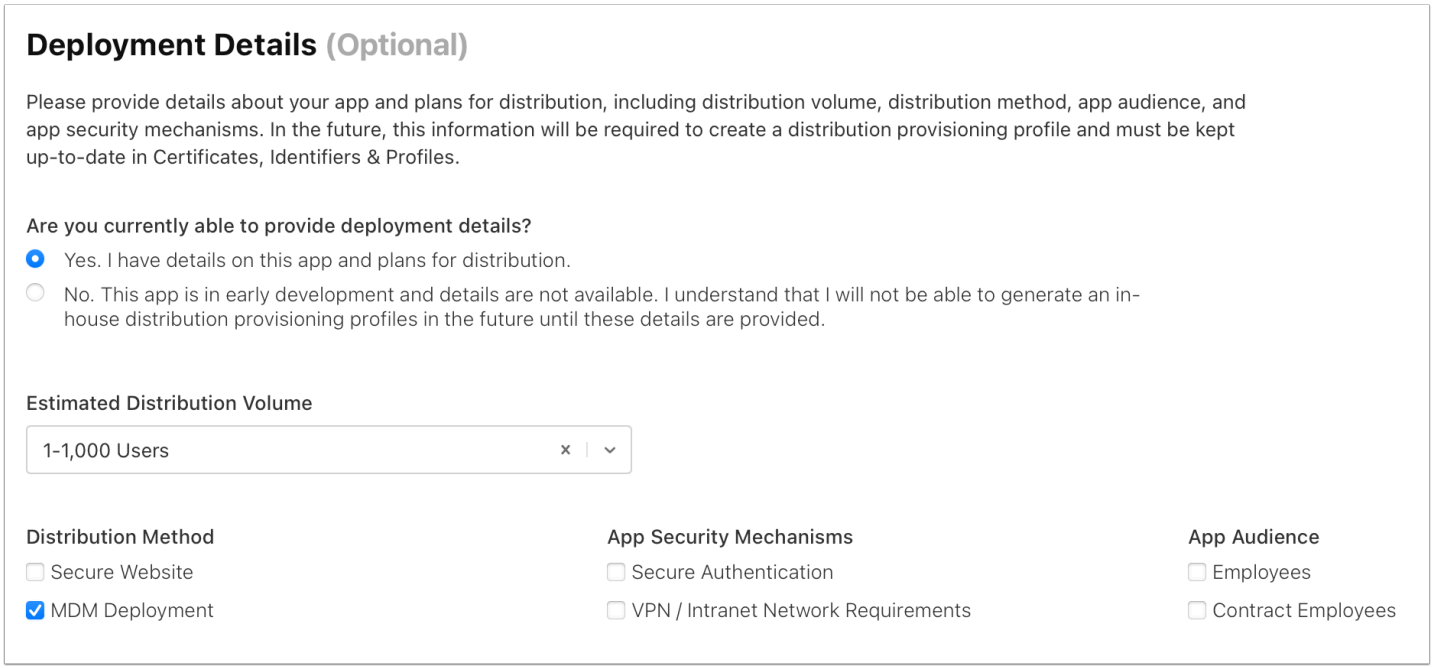
Three files will be generated when you create a build:

- `[appname].ipa` : the application file
- `[appname].plist` : a plist file
- `[appname].html` : the page to navigate to

If you copy the three files to the folder that was specified as the deployment URL, users will be able to navigate to https://[yourserver].com/[appfolder]/[appname].html from their iOS device to automatically install the application.

3. Deploy the app over the air (using an MDM)

This method allows you to distribute your in-house app easily: the `.ipa` file will be installed via a mobile device management system, such as Airwatch, Mobile Iron, JAMF, FileWave, Meraki, etc. The build process is exactly the same as for Twixl's built-in over the air deployment, except that you do not need to enter an OTA URL in the build settings.



In the App ID Configuration on developer.apple.com, you'll need to check 'MDM Deployment' in the Deployment Details. Currently this information is optional, but this may become a requirement in the future.

4. Deploying to Apple Business Manager

If your app is not intended for all audiences, Apple offers two different approaches to publish your app either in-house or privately.

If you are a company with more than 100 employees and you want to deploy your own app internally you can subscribe to the Apple developer Enterprise program. Because you publish internally, there is no app review process in this case. You will deploy the .ipa file (Enterprise build) on an internal server and provide access to the users that need to get the app.

One important guideline from Apple is that you can only publish to employees within your company. There's no strict control of this and in reality we see lots of customers having a "broad interpretation" of this concept.

But then there's the other approach Apple offers to deploy an app privately. Create a Custom app for business.

With Apple Business Manager, you can privately and securely distribute to specific partners, clients, franchisees, … and businesses can also distribute proprietary apps to their internal employees. This is also useful for businesses with less than 100 employees.

The clients, partners, franchisees, … you want to publish to first need to have an Apple Business account.

Then you, as a publisher, will need to identify in App Store Connect to whom you are going to publish by registering the Apple Business IDs of your clients, partners, franchisees, … or if you want to publish internally to your employees their individual Apple IDs.

Businesses that you identify in App Store Connect will see your app and be able to purchase it in the Apps and Books section of Apple Business Manager. You can offer custom apps for free or at any price tier you choose.

You'll need to upload your app to the App Store for review and select the Custom App Distribution option. You'll deploy the .ipa file (a Twixl App Store build).

The organizations that you identified will be able to distribute your app through a Mobile Device Management system. Alternatively, organizations can choose to provide redemption codes to authorized users to download the app on the App Store.

An organization can identify those authorized users with their Apple ID.

Based on this you can also use the Apple Business Manager approach to publish internally to individuals who you will identify with their Apple ID within your own account.

This allows you just to ask the Apple ID of individuals and enable them to download your app privately just by providing them a redemption code.

If you choose not to go for the Custom App Distribution you can still work with restricted access. An app with restricted access will require you to login when you start the app, and after a successful login you can access the content. But understand that Apple might reject such an app if they identify it as only targeting a limited audience in a specific organization. At any rate, when you submit such an app, make sure to provide a test login for the Apple review team, that displays relevant content.

A hybrid approach with parts of the content being available for all users and other parts only to entitled users is also an option. Different users or groups can have access to different parts of the content.

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 an app 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.

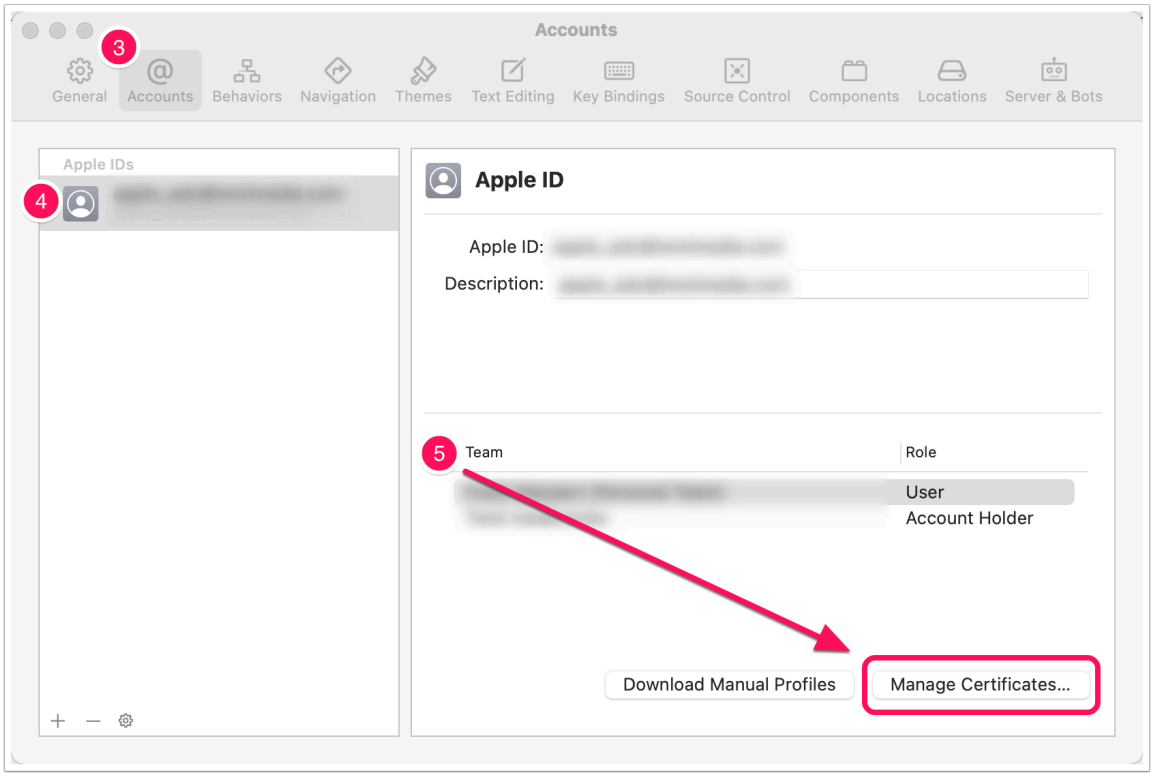
1. Obtain and install the necessary certificates

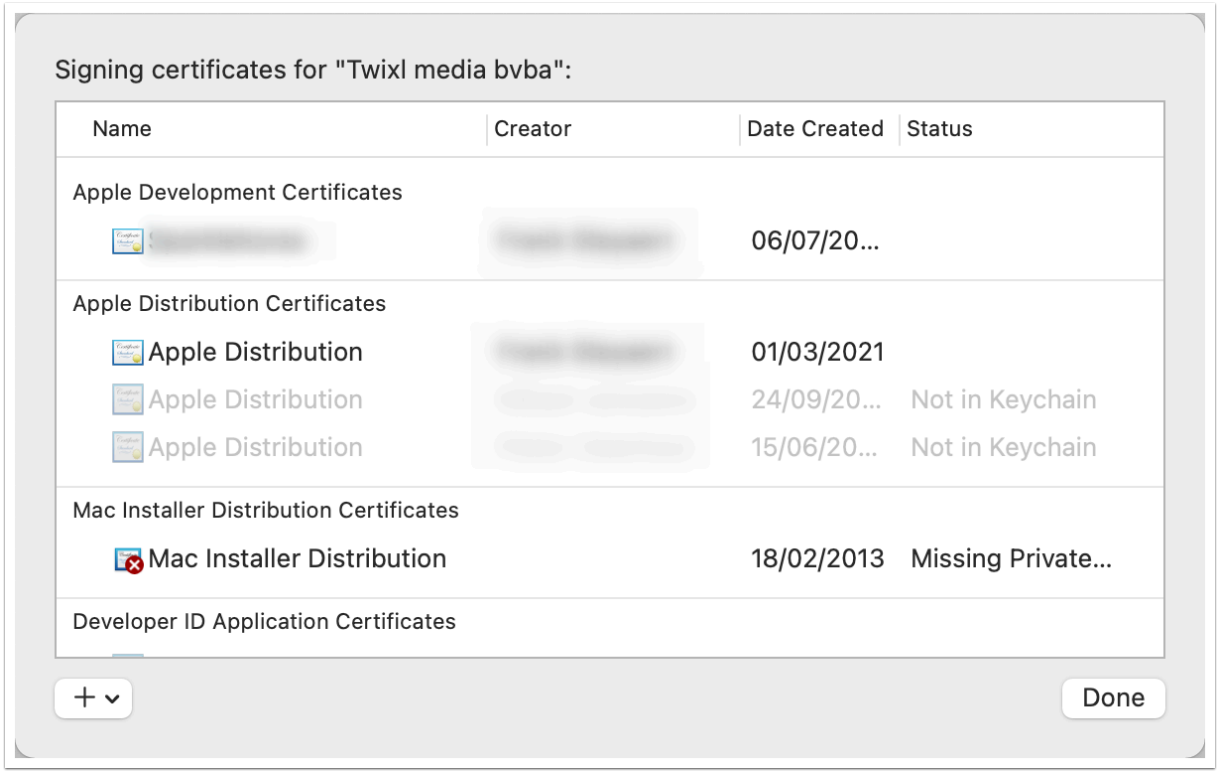
REQUIREMENTS:

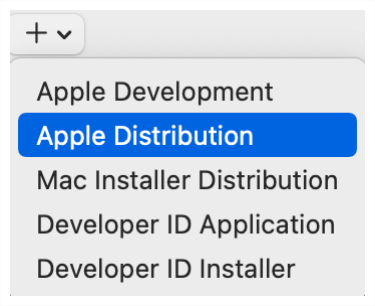
Make sure you are logged in as an Apple Developer in Xcode. For more info on how to do that, see this article.

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 create a distribution certificate.
7. Xcode will do the signing request, get and download the certificate and add it to your Keychain.


Accounts
3
General
Accounts
Behaviors
Navigation
Themes
Text Editing
Key Bindings
Source Control
Components
Locations
Server & Bots
Apple IDs
4
Apple ID
Apple ID:
Description:
5
Team
Role
User
Account Holder
Download Manual Profiles
Manage Certificates...


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
Done



IMPORTANT NOTE:

Twixl Publisher uses a Distribution Certificate 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'). With Twixl Publisher 12 and higher, you need **Apple Distribution** certificates. Therefore we strongly advise you to delete your old certificates and start all over. To do this, you need to:

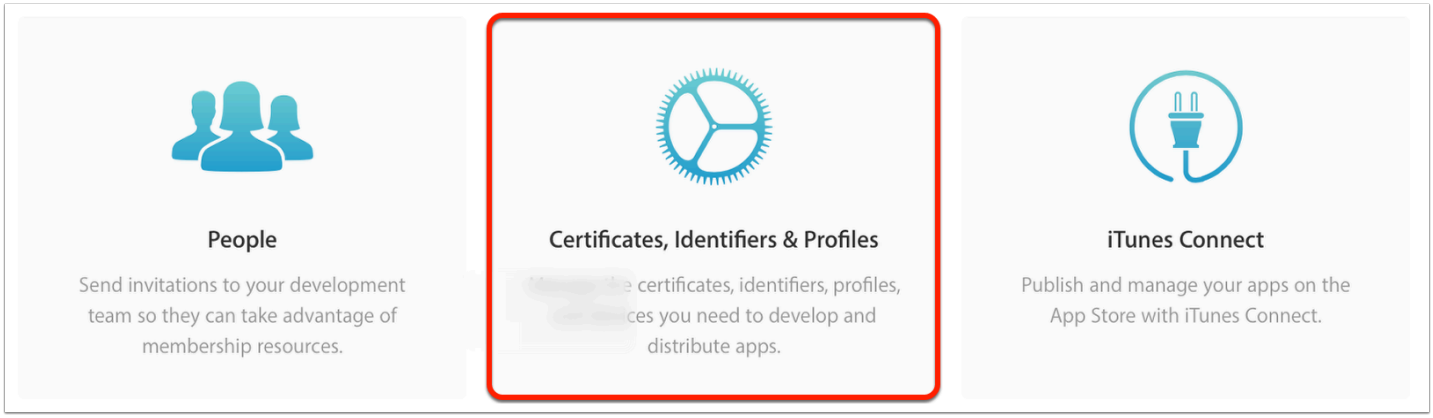
1. Delete all older iOS Developer and Distribution certificates in your Keychain Access app (found in /Applications/Utilities)
2. Revoke all your old iOS Distribution certificates on the Apple Developer Portal

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).

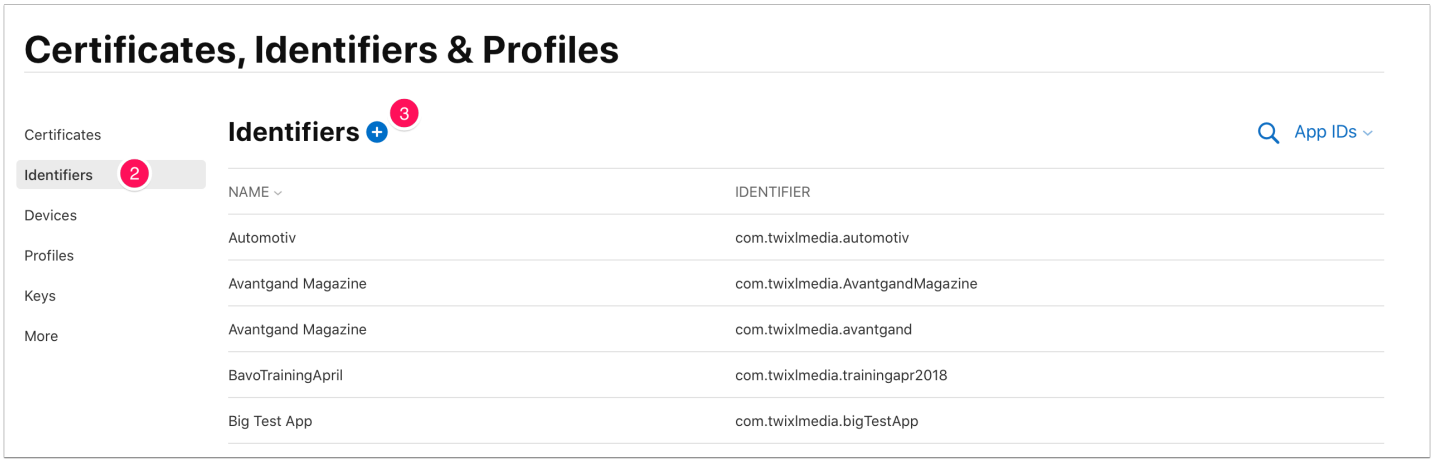
For **Enterprise apps**, though, the consequence will be that apps built using that certificate will no longer start. If you need to extend the expiration date for a provisioning profile of an existing Enterprise app, you can do the following:

- 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



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).



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 (required for your **Ad Hoc** builds only).

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.

5. Building your app

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

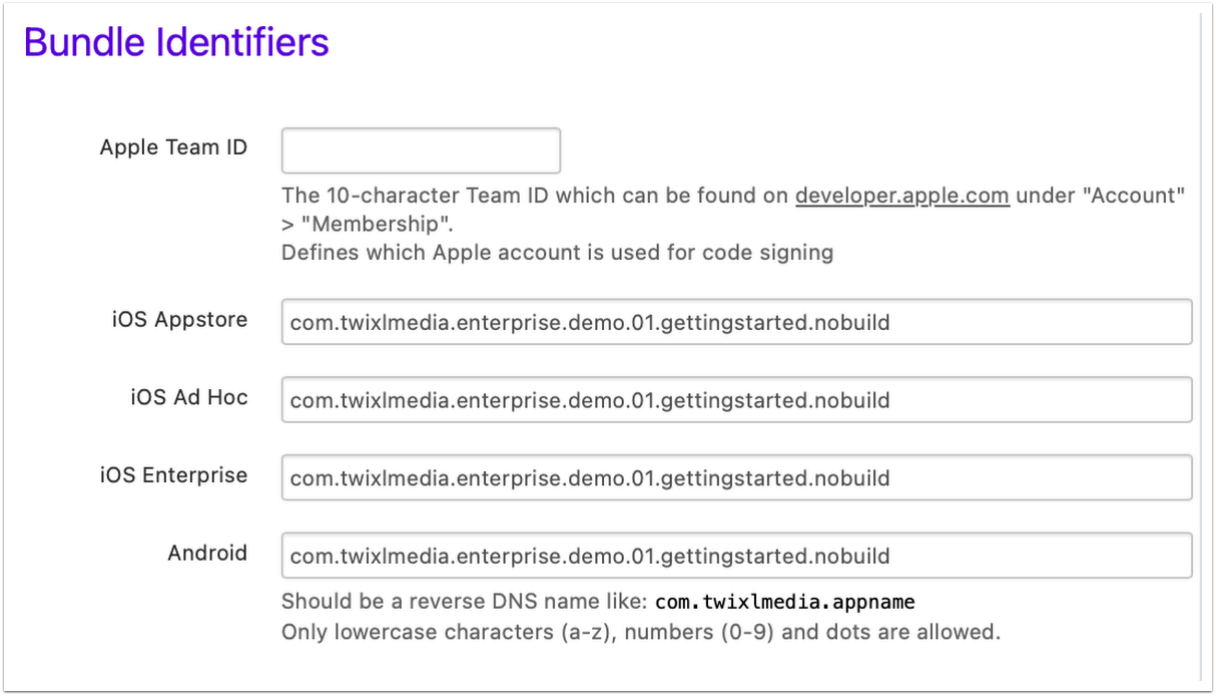
- Build Settings overview
- Building your Twixl app

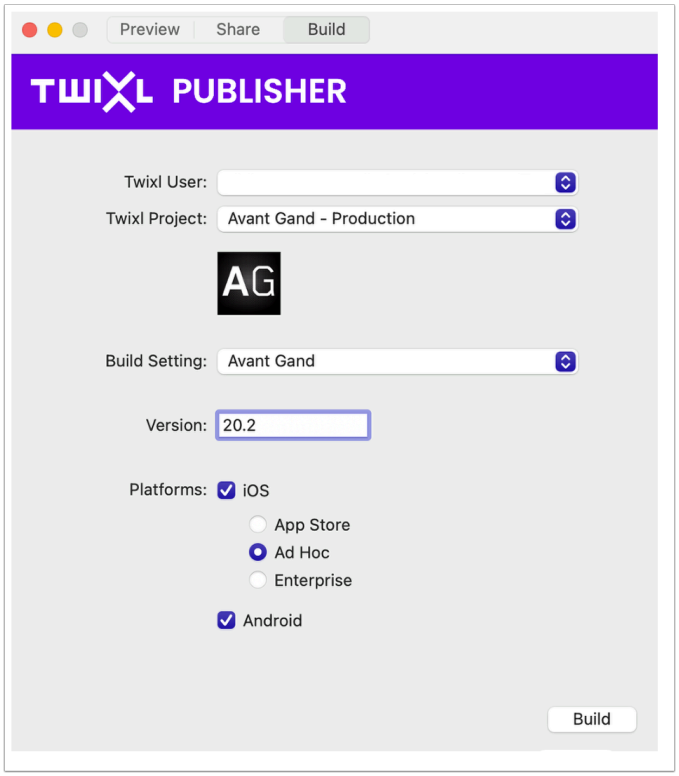
iOS: App build types

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.

In order to build an iOS app with the macOS app, you need to have an app in **Production Mode** on the Twixl platform.





1. 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. Make sure you define the necessary devices (at least one!) that can be used in the Apple Developer portal first, as described here.

2. 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.

3. 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 (or to TestFlight first).

Distributing for Android

Deploying an Android app in Google Play

Uploading to Google Play is very straightforward. When you create an Android build, Twixl Publisher will generate two app files, one with extension `.apk`. and a second one with extension `.aab` (Android App Bundle).

`.apk` : this is the original extension that you can still use for side loading and testing on your Android device, or for in-house distribution.

`.aab` : this is the format that is required for new apps or for updates of existing apps.

1. Prerequisites

- You have a Google Play Developer account.
- Your app is built in Twixl, and you have downloaded the generated .aab file (in the build folder).
- You have access to the private key file (.pepk) and upload certificate if needed (also in the build folder).

2. Steps to deploy your app

2.1 Log in to the Google Play Console

- Go to https://play.google.com/console and sign in.

2.2 Create a New App (if this isn't an update of an existing app)

- Click '**Create app**'.
- Fill in the required information (App name, default language, app type, etc.).
- Accept the developer declarations and click '**Create app**'.

If you update an existing app, you can start with step 2.2.

2.3 Upload the .aab File

- In the left-hand menu > **Test and Release > Production > Create new release.**
- Under the "App bundles and APKs" section > **Upload >** select your .aab file (from the Twixl build folder).

> If you would first want to create an internal testing release, select 'Testing' in the left bar menu in stead of 'Production'. The workflow is exactly the same as for a production release.

2.4 App Signing with Google Play (Handled Automatically)

By default, Google Play uses **Play App Signing**, which means:

- Google manages the App Signing Key used to re-sign your app before distribution so it is always possible to identify yourself as the rightful owner to update an app.
- You sign your .aab file using an Android keystore, which is already embedded in your Twixl-generated build.

> No manual action is needed here unless this is your first release and Google asks you to provide the **Upload Key certificate** or a **private key export (.pepk file)**.
>
> If prompted:
>
> - Always select: '**Let Google manage and protect your app signing key.**'
> - Upload the .pepk file (from your Twixl build folder).

> Do not click 'Change app signing key' as it may revoke access for your current upload key.

2.5 Complete Release Details

- Add release notes.
- Click 'Next' through the steps (Device compatibility, App content, etc.).
- Make sure all policy sections (Data safety, Ads, Target audience, etc.) are completed under **Policy**.

2.6 Add artwork

Also upload your artwork, showcasing what to expect and attracting the user to download the app. Check the latest Google Play requirements and guide for artwork here.

2.7 Review and Publish

- Review your release.
- Click '**Start rollout to production**' > '**Roll out**'.

Google will process your submission and review it. If it's your first release, this may take a few days.

3. General remarks

Android Keystore

When you publish an app in the Google Play store, Google expects that app to be signed with a certificate so you can identify yourself as being the rightful owner of an app.

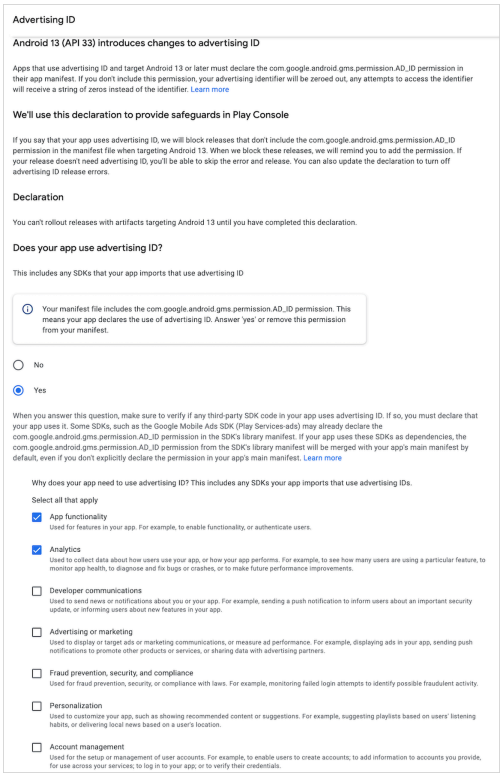
Twixl Publisher takes care of creating this "**keystore**" for you in the background. Every update of your app will have to be signed using the same keystore information (the app's "fingerprint"), otherwise it will not be accepted on Google Play. If you are migrating your app from another platform, check out this paragraph specifically.
The keystore information is automatically saved as part of the build setting for your app on the platform.

If you move from another system and need to add an existing keystore to the Twixl platform, check this help article.

Advertising ID

Google is constantly changing the security approach during the upload process. Currently they ask for apps if they use advertising ID.

Here we advise to answer 'Yes' and indicate 'App functionality' and 'Analytics' in the list of options.



Cleartext traffic warning

Google Play might return a warning about cleartext traffic in the Pre-launch Report. Although we recommend to always use https, Google still allows http-requests but warns you about this only at this point. While we are aware this might change in the future, it is important you prepare to only use secure https links. For the time being, however, you can continue deploying your app in Google Play.

Deploying an Android app in-house or for testing

How to get the .apk file you created on an Android tablet or phone outside of Google Play

1. Sideloading

This method allows you to install the .apk using a USB cable, with the help from extra software.

WARNING

Depending on which method and which application you use, it might be necessary to activate *USB Debugging* and *Allow Unknown Sources* in the *Settings* of your Android device. Please refer to the manual of your mobile device for instructions on how to activate those options. Due to the vast multitude of available Android devices, it's impossible to provide those instructions here.

1.1. Tools for Windows

- Android File Transfer for Windows
- File Explorer (default application on Windows-machines)

1.2. Tools for Mac

- Android File Transfer for Mac

WARNING

Please refer to the manuals of the according software for instructions on how to setup and connect your Android device.

2. Deploy the app over the air (using an MDM)

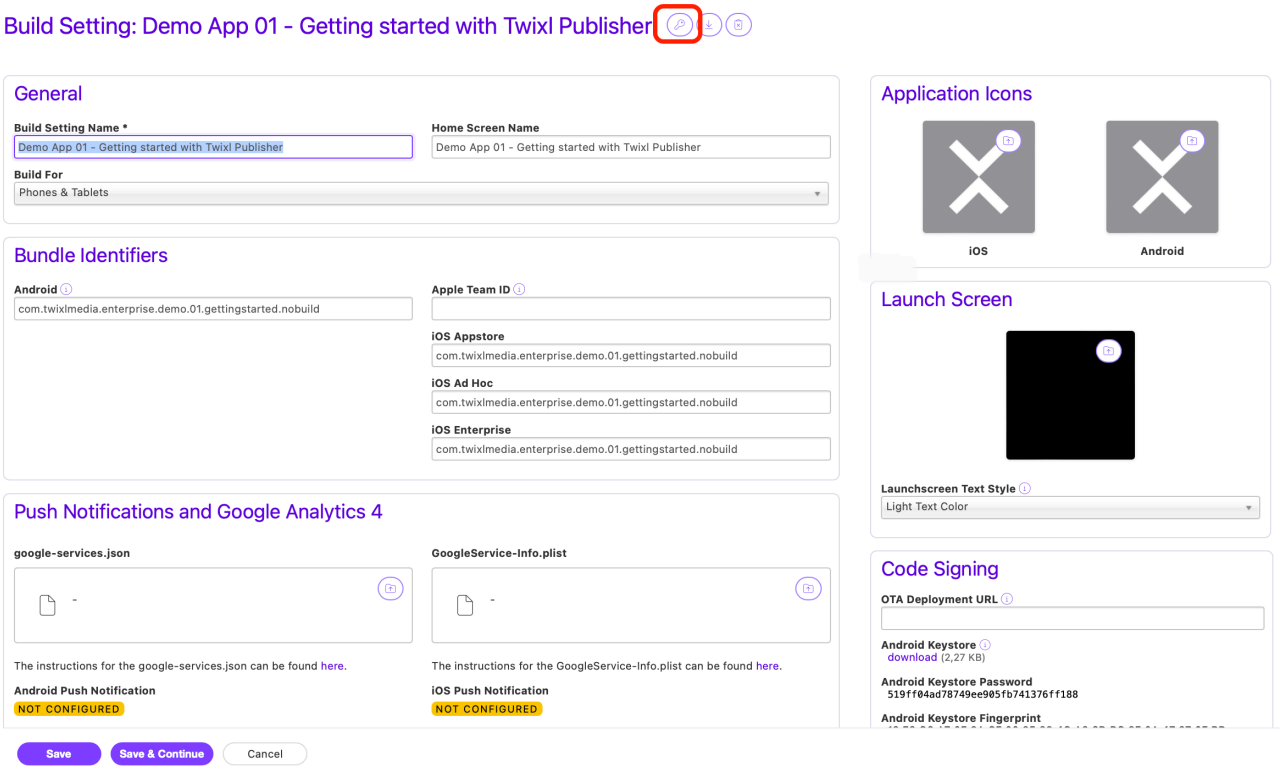
This method allows you to distribute the app easily without the need for sideloading. The .apk file will be installed via a mobile device management system, such as Airwatch, Mobile Iron, JAMF, FileWave, Meraki, etc.

Android: Migrating from another platform

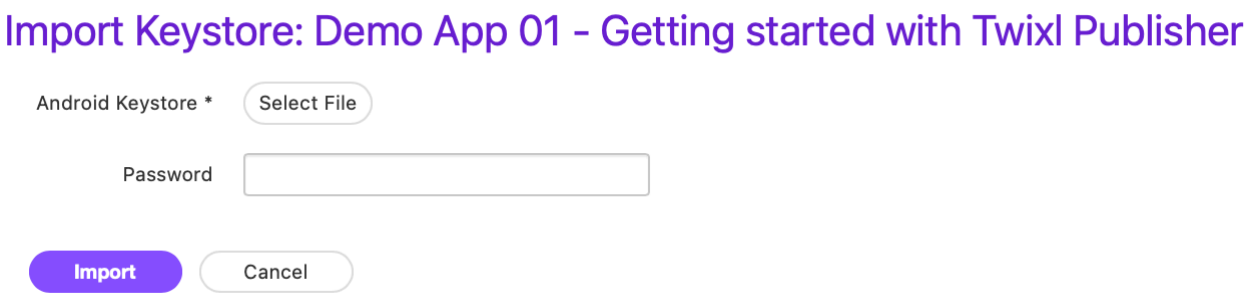
Migrating your Google Play app from another platform, such as Adobe DPS, etc.

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 keystore' option in the top action buttons on top of the Build Setting 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 .

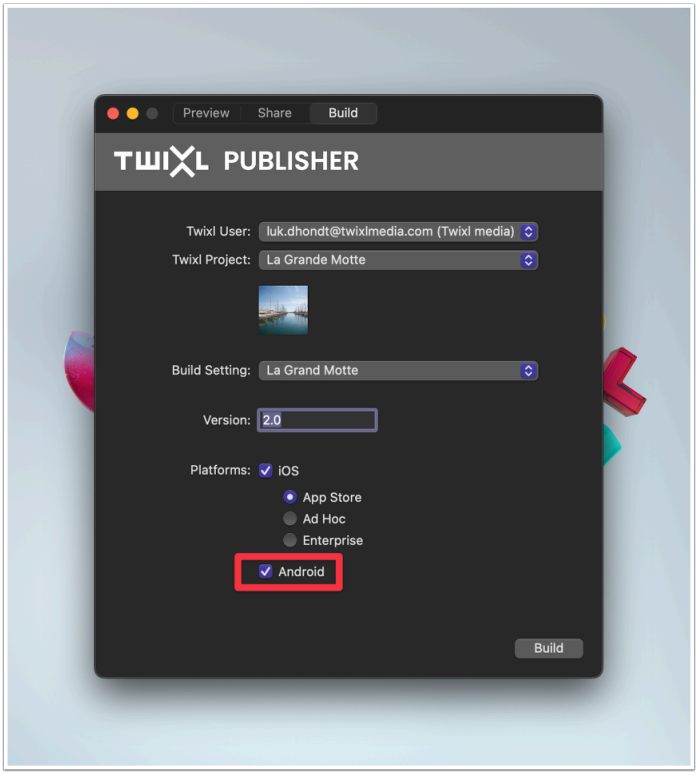

Import Keystore: Demo App 01 - Getting started with Twixl Publisher
Android Keystore *
Select File
Password
Import
Cancel

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.

In order to build an Android app with the macOS app, you need to have an app in Production Mode on the Twixl platform.



Deploying an Android test build

Removed this article from the portal as it is obsolete (May 5, 2021)

1. Preview on a device

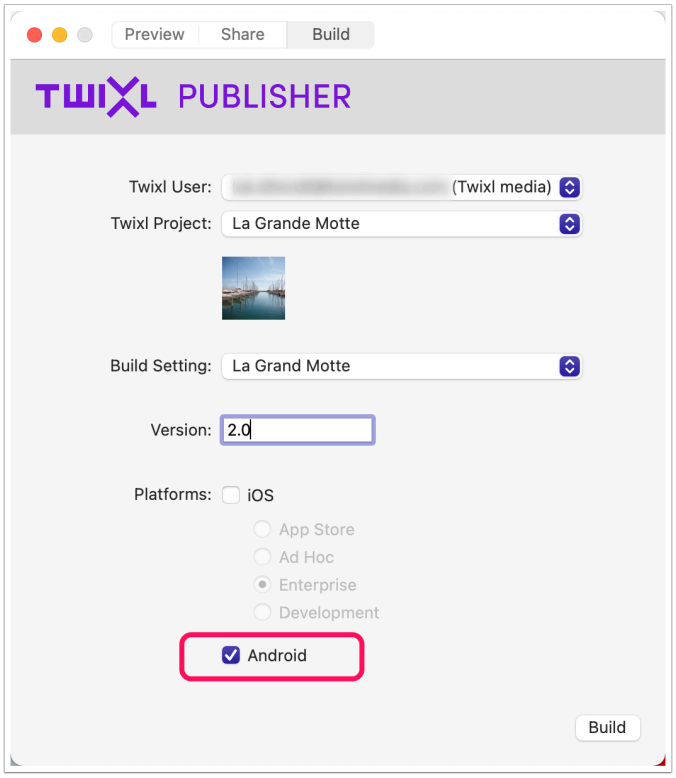
Preview on a device is the quickest way to get content on your tablet or phone, and allows you to preview in the Twixl app.

More details are available in this article.

2. Single-file Android test build via the SD card

This method will allow you to install the whole app and its contents as a single .apk file.

Start by creating the test build of your app. The result will be a folder containing a file with the extension .apk (e.g. main.10000.com.twixlmedia.mymagazine.apk). The package name will be based on the Android Application ID you defined in the Build Settings.



1. Copy the installation file (.apk) to the SD Card (any location will do as this is only an installation file). If you are on a Windows computer, there is no additional installation needed to copy files to the SD Card of the tablet.

2. After you copied that file to the SD Card, use a tool like ES File Explorer or File Manager on your tablet to navigate to the contents on that SD Card. Now open the installation .apk file to launch the publication.

TIP:

If you're on macOS, you can use 2 tools to copy files to your Android device:

- Android File Explorer
- Android tool for Mac

IMPORTANT NOTE:

Although this method may be a quicker way to test your app, it's always good practice to test a Google Play build according to the method below before submitting your final app to the Google Play store.

Other sections

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

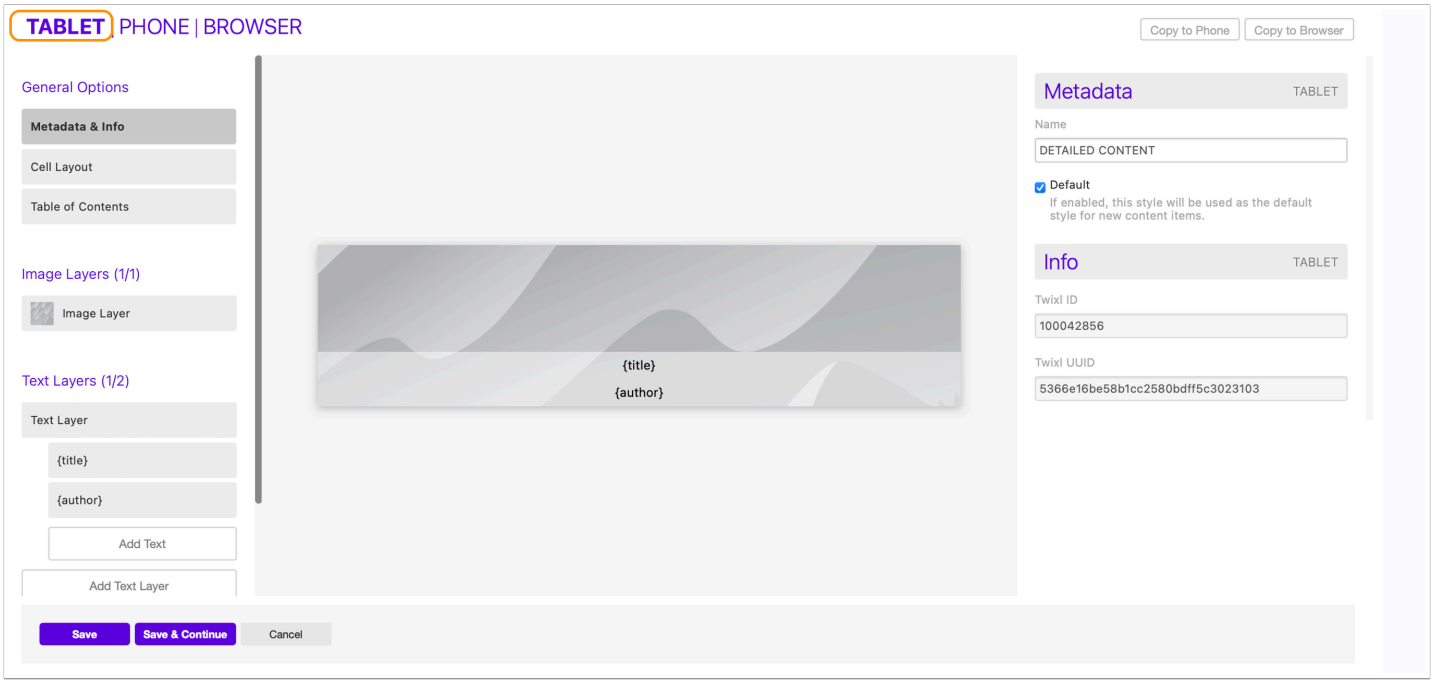
TIP CONCERNING TWIXL ARTICLES and PDF CONTENT ITEMS:

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.



Related Content

- Working with Content Items
- Working with Collections
- Application Options
- Styling: Item Styles

App privacy details on the App Store

In early 2021, Apple has new requirements in terms of the privacy information that apps published on the App Store use. Below is the information you need to be aware of when publishing a Twixl app on the App Store.

What does this mean for Twixl apps?

This means you will need to clarify the usage of Data Types in your Twixl app. By default, Twixl apps use the following Data Types:

- Coarse Location
- Device ID
- Product Interaction
- Other Usage Data
- Crash Data

ABOUT OTHER DATA TYPES:

Depending on the content in your app, it might be possible that you'll need to add even more used Data Types.

Example:

If an HTML-article contains technology to determine the exact location of the mobile device on a map, you will need to add Precise GPS locations as a used Data Type.

How to manage app privacy?

For more info on how to add and remove data types, see this Apple KB-article (> Adding and removing data types).

Data Types Edit

- 5 data types collected from this app: Coarse Location, Device ID, Product Interaction, Other Usage Data, Crash Data

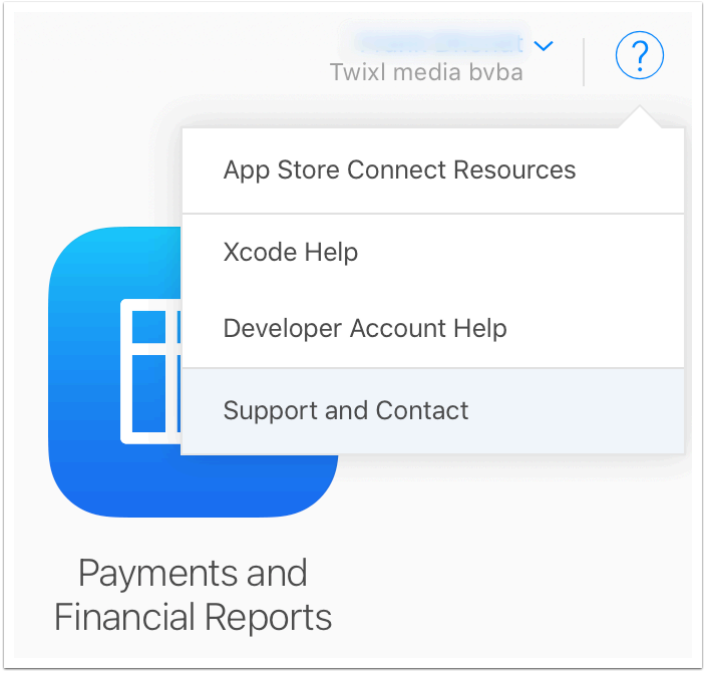
Asking Apple for an "expedited" review

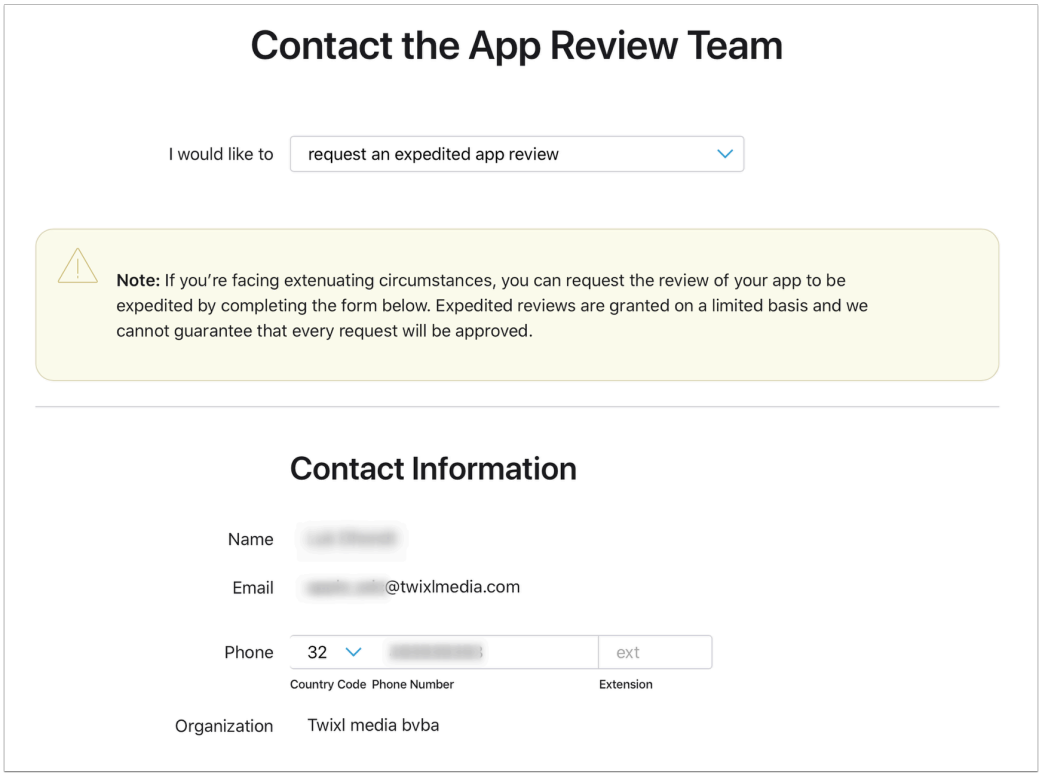
If you ever run into a problem with an app you published in Apple's App Store, and want to update it as fast as possible, you can ask the Apple review team for an "expedited review" explaining the problem.

How to ask Apple for an "expedited" review?

In AppStore Connect, select “Support and Contact”. From the list of topics that follows, select "App Review", then "Request Expedited Review", and fill out the form.

While your request does not give you any guarantee, in many cases the request is taken into account and your update may indeed be approved faster.




Contact the App Review Team
I would like to
request an expedited app review
Note: If you're facing extenuating circumstances, you can request the review of your app to be expedited by completing the form below. Expedited reviews are granted on a limited basis and we cannot guarantee that every request will be approved.
Contact Information
Name
Email
@twixlmedia.com
Phone
32
ext
Country Code
Phone Number
Extension
Organization
Twixl media bvba

Google Play App Signing

1. About "Play App Signing" by Google

With Play App Signing, Google manages and protects your app's signing key for you and uses it to sign your APKs/AABs for distribution. It's a great and secure way to store your app signing key, that helps to protect you if your key ever gets lost or is compromised.

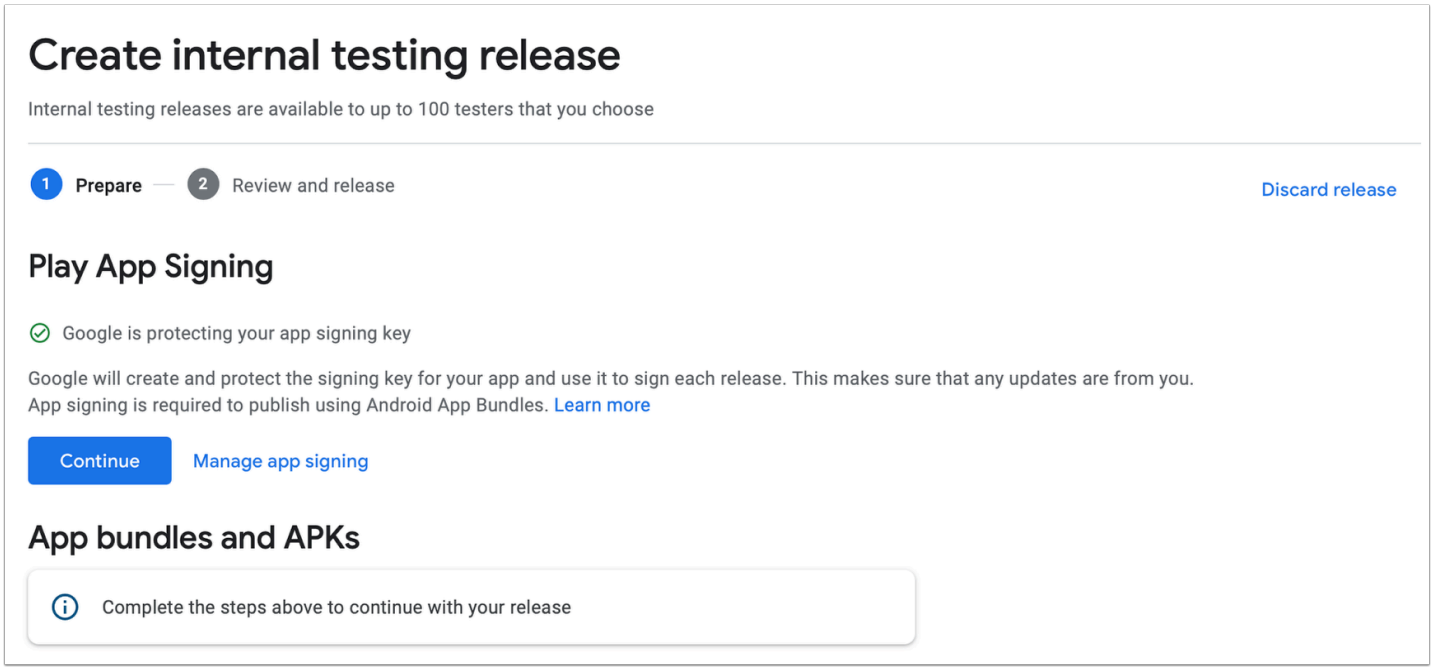
When you use Play App Signing, your keys are stored on the same infrastructure that Google uses to store its own keys (keys are protected by Google's Key Management Service).

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. If you're not enrolled in Google Play App Signing and you lose your keystore, you'll need to publish a new app with a new package name.

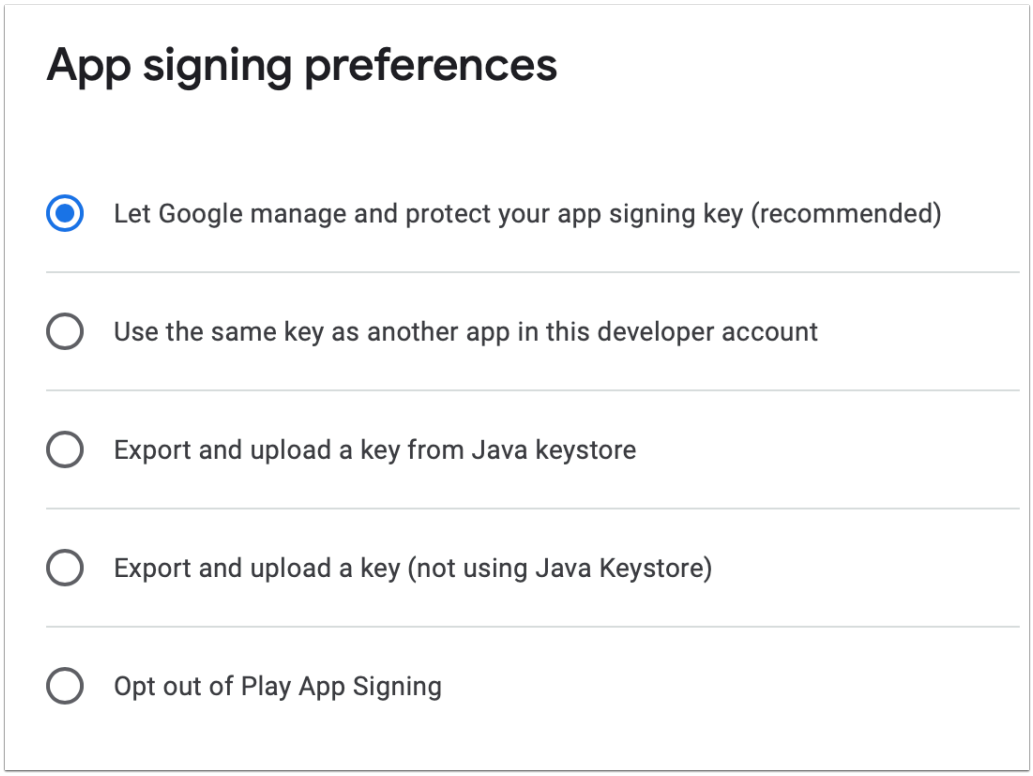
2. How to opt in to app signing for a new app

1. Open Play Console.
2. Select your app.
3. In the left menu > Test and release > Setup > App Signing.

Google Play App Signing will be activated by default for a new app.



If you select *Manage app signing*, you'll see a number of signing preferences. It is recommended to use the default option: *Let Google manage and protect your app signing key.*



3. How to opt in to app signing for an existing app

Make sure you have your original App Signing Key ready! You can find this file called 'app signing private key.pepk' in the Android build folder of your app that was created with the Twixl Publisher macOS app.

1. Open Play Console > Select your app
2. Left menu > Test and release > Setup > App Signing
3. (If you haven't already, review the Terms of Service > select Accept.)
4. Select 'Let Google Play manage your app signing key' > 'Use existing app signing key from Java Keystore'
5. Upload your App Signing Key from the Android build folder.

About the new Google Play Console

Google introduced a new Google Play Console, still in beta though. Our advice:

Don't use it (yet)!

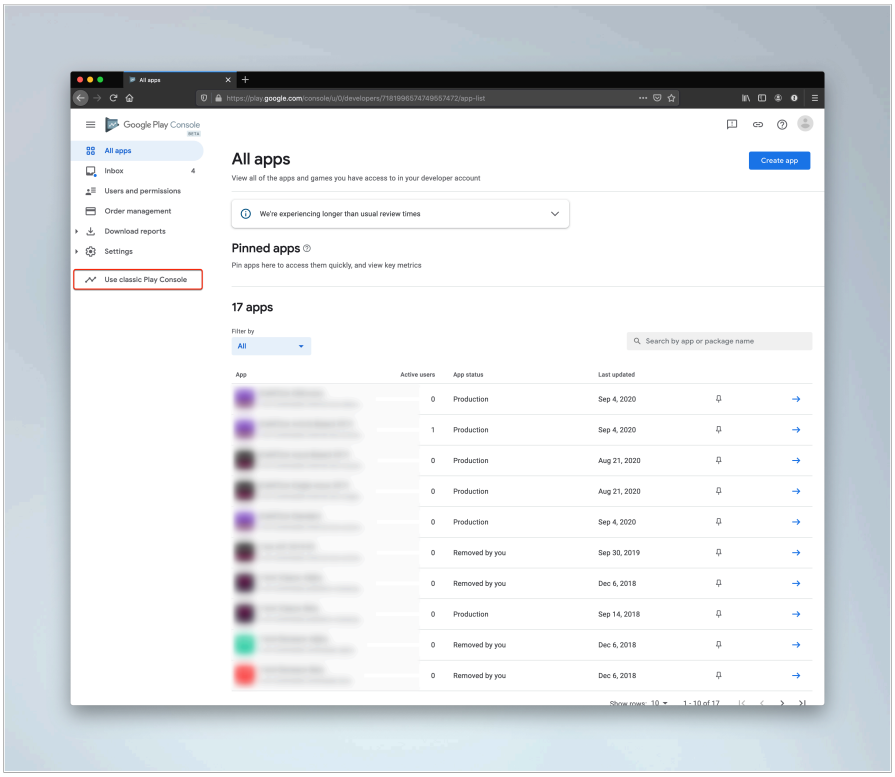
The new Google Play Console is supposed to introduce a new style and an improved experience, but there are just too many problems in this beta:

- Only Google Chrome is supported. Other browsers render the new interface **terribly slow** (yes, really slow).
- Not all interface elements are already in place. For example: Essential *Store Listing* options are not visible and as a result the upload of **new** `.apk` files will fail.
- ...

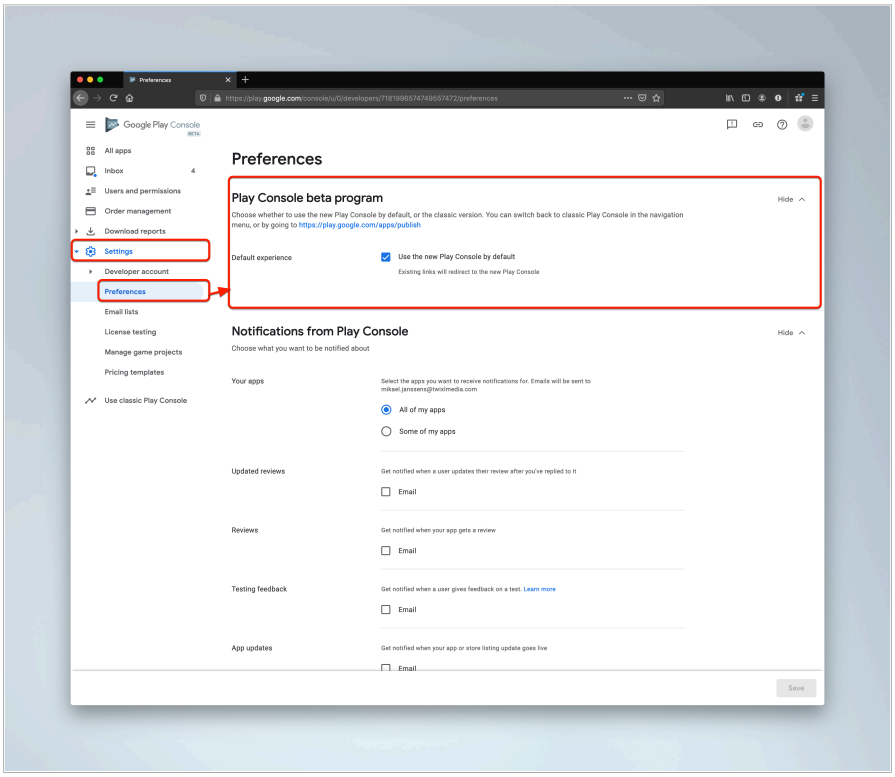
Switch back to the Classic Google Play Console (at least for now)

Because of those problems, we strongly suggest you to switch back to the Classic Google Play Console for now:

1. Make sure your user had Admin permissions
2. Select *Use Classic Play Console* in the left navigation bar



- You can also use the old interface, by navigating directly to https://play.google.com/apps/publish
- And you can even set a preference to use the Classic Console by default:

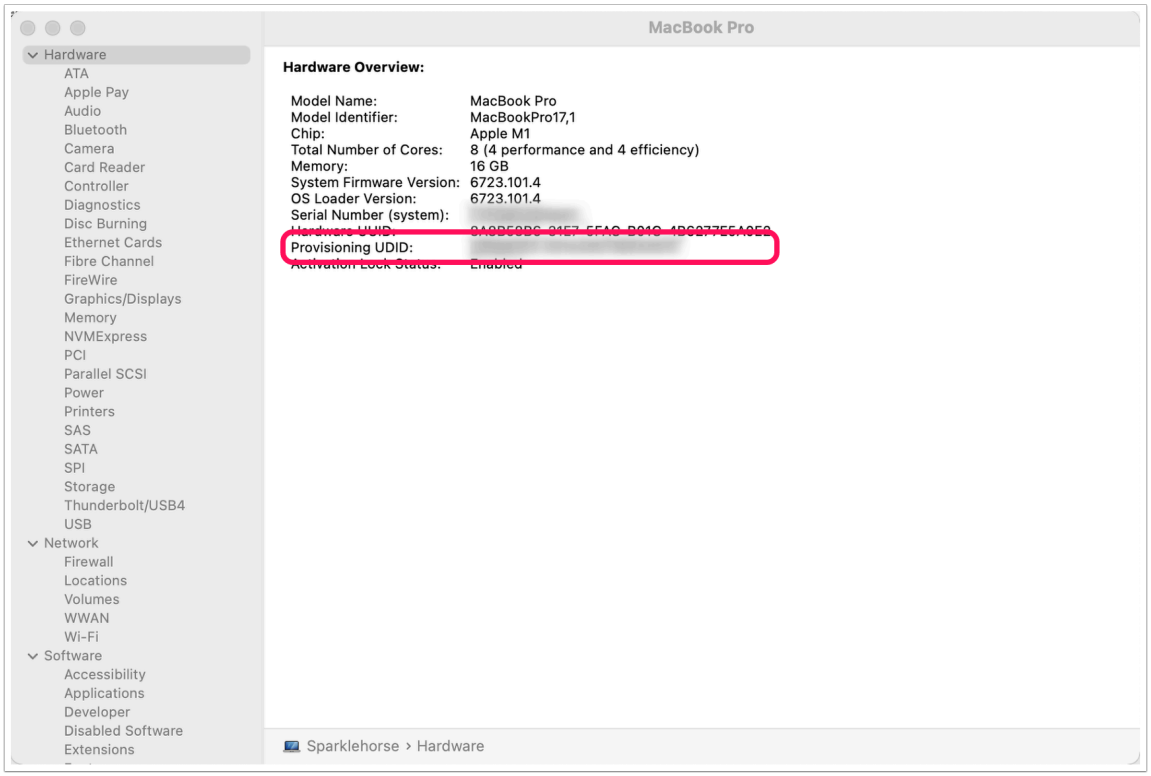


Registering an Apple Silicon Mac

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

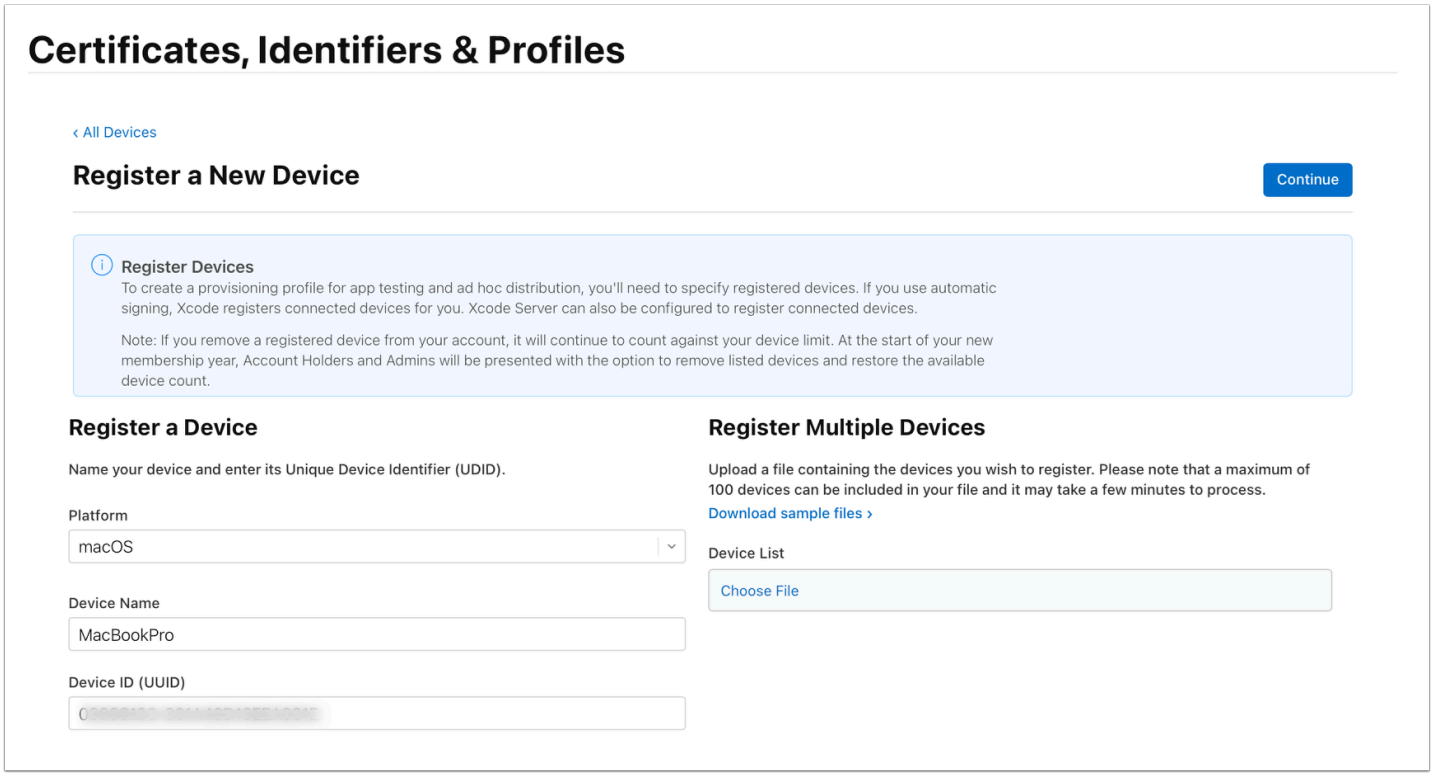
1. Find the Mac's device UDID

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.



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



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

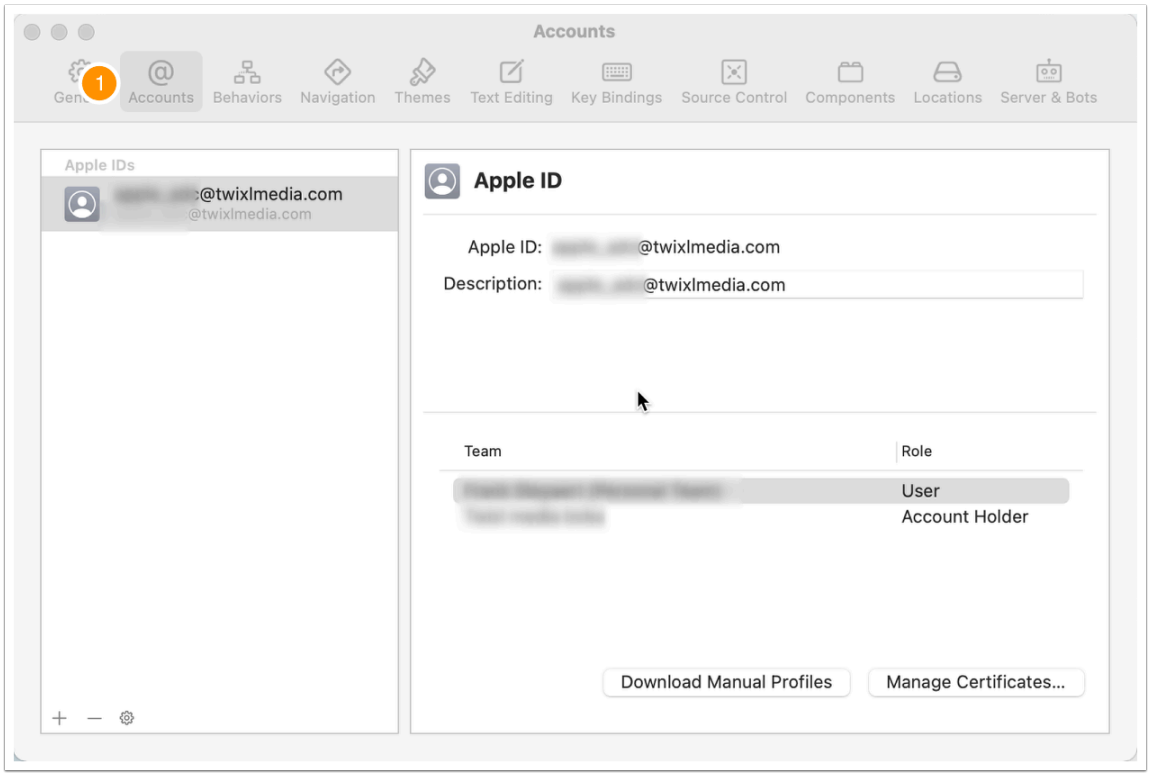
Xcode set up

If you want to preview or build a Twixl iOS app, you also need to have Xcode installed. Follow the steps below to set up Xcode.

Install the **latest version of Xcode** from the Mac App Store.

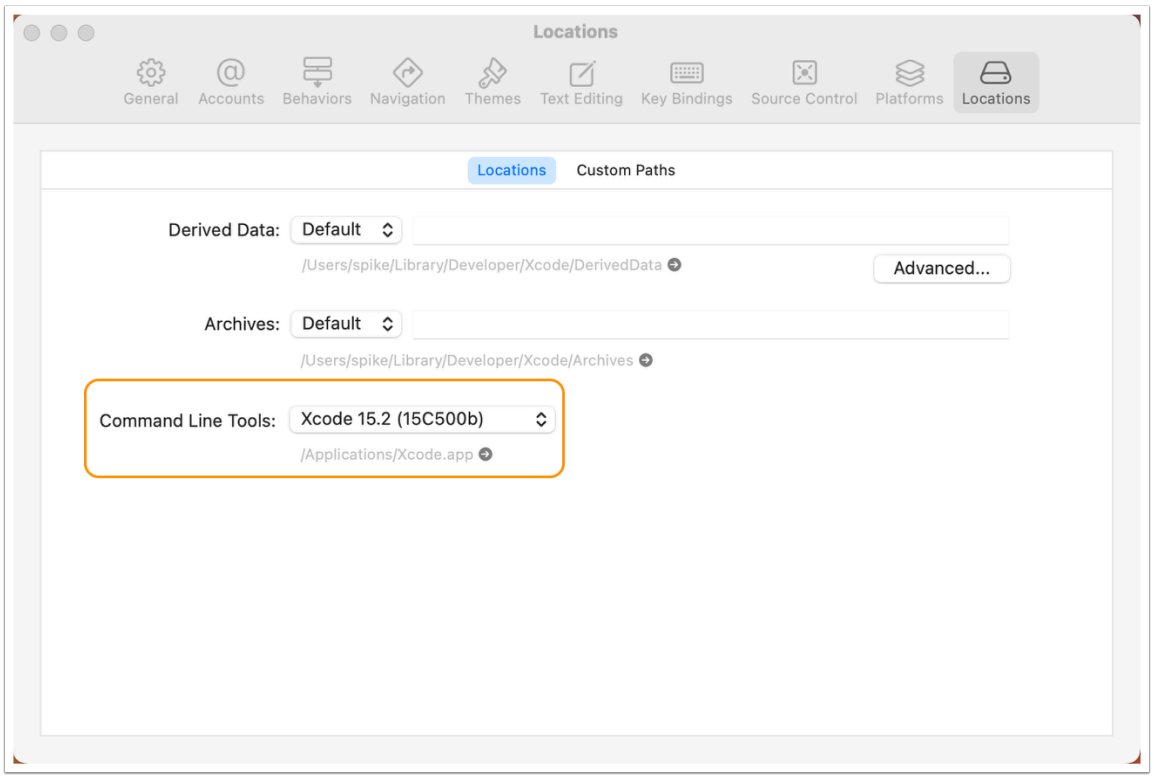
Next steps:

- Launch Xcode > Xcode Settings > Accounts > Add your **Apple Developer account ID**

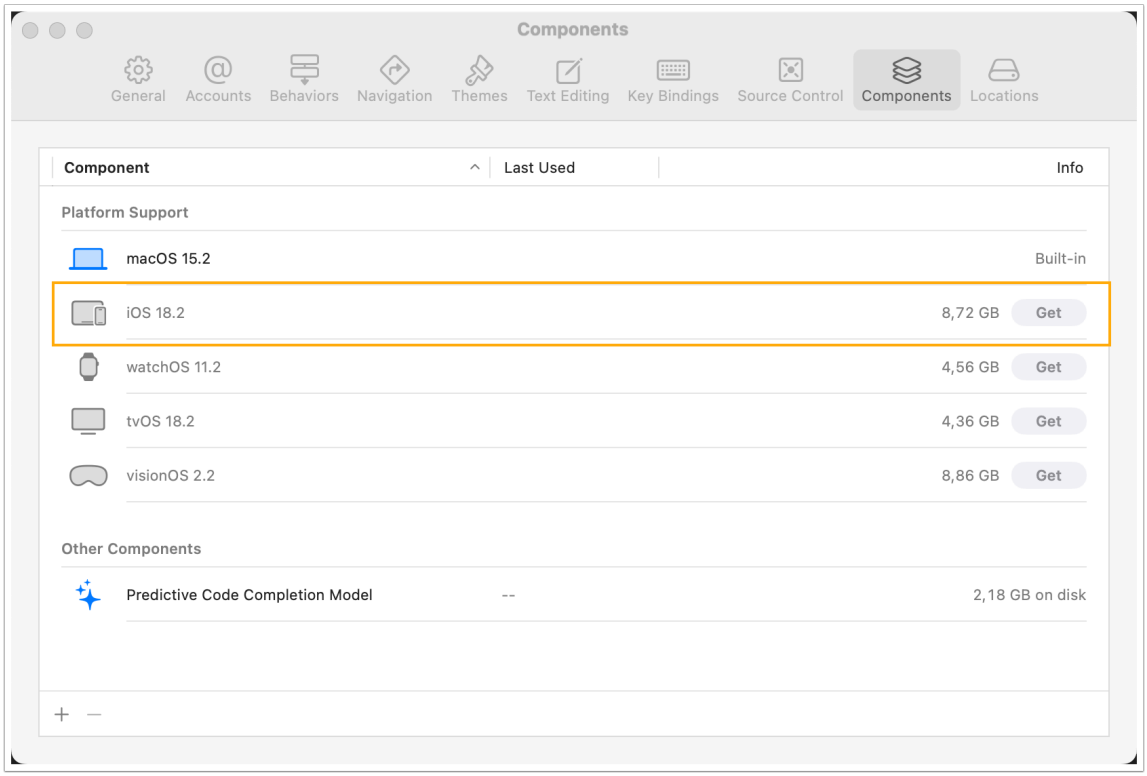


The account requires a role of type *Account Holder*, *Admin* or *App Manager* to be able to build an app.

- In Xcode > Xcode Settings > Locations > Make sure a version of the **Command Line Tools (CLI)** is selected



In Xcode > Xcode Settings > Components > Click 'Get' to download the latest **iOS SDK**.



1. What's next ?

Before you are ready to finally build your app, there are two important steps to go through:

- Configure a build setting for your app on the platform.
- Create a signing certificate in Xcode.

Then you are ready to **build your app**.

2. Related articles

These articles may be relevant for further reading:

- iOS app build types
- Google Play app signing

Creating your Launch Image

If you want to build your custom Twixl App, you can add a Launch Image.

1. What?

The Launch Image is what it says: it's the first thing your users see when they launch your app.

2. Requirements?

You configure the Launch Image of your app in the build setting of your app. It needs to be:

- square
- `2208px` by `2208px`
- `.png` format

3. How to design?

We use one Launch Image for both Phones and Tablets.

This means that the image gets cropped and this can impact your design. So it's better to take this into account when creating your launch image.

4. What's next?

Once you finished your Launch Image, you can go ahead and build your app.

The following articles will help with the next steps:

- Build Settings overview
- Building your Twixl app