Distribute your app

Table of Contents

iOS Specific 3
- Deploying an iOS app in the App Store 4
- Deploying an iOS app in-house 6
- In-app purchases for iOS apps 12
- Subscriptions for iOS & Android 15
- Getting your iOS app approved 20
- App privacy details on the App Store 27
- Asking Apple for an "expedited" review 29

Android Specific 31
- Deploying an Android test build 32
- Deploying an Android app in Google Play 34
- Deploying an Android app in-house or for testing 38
- In-app purchases for Android apps 40
- Subscriptions for iOS & Android 45
- Google Play App Signing 50
- About the new Google Play Console 53

iOS Specific

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. 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.

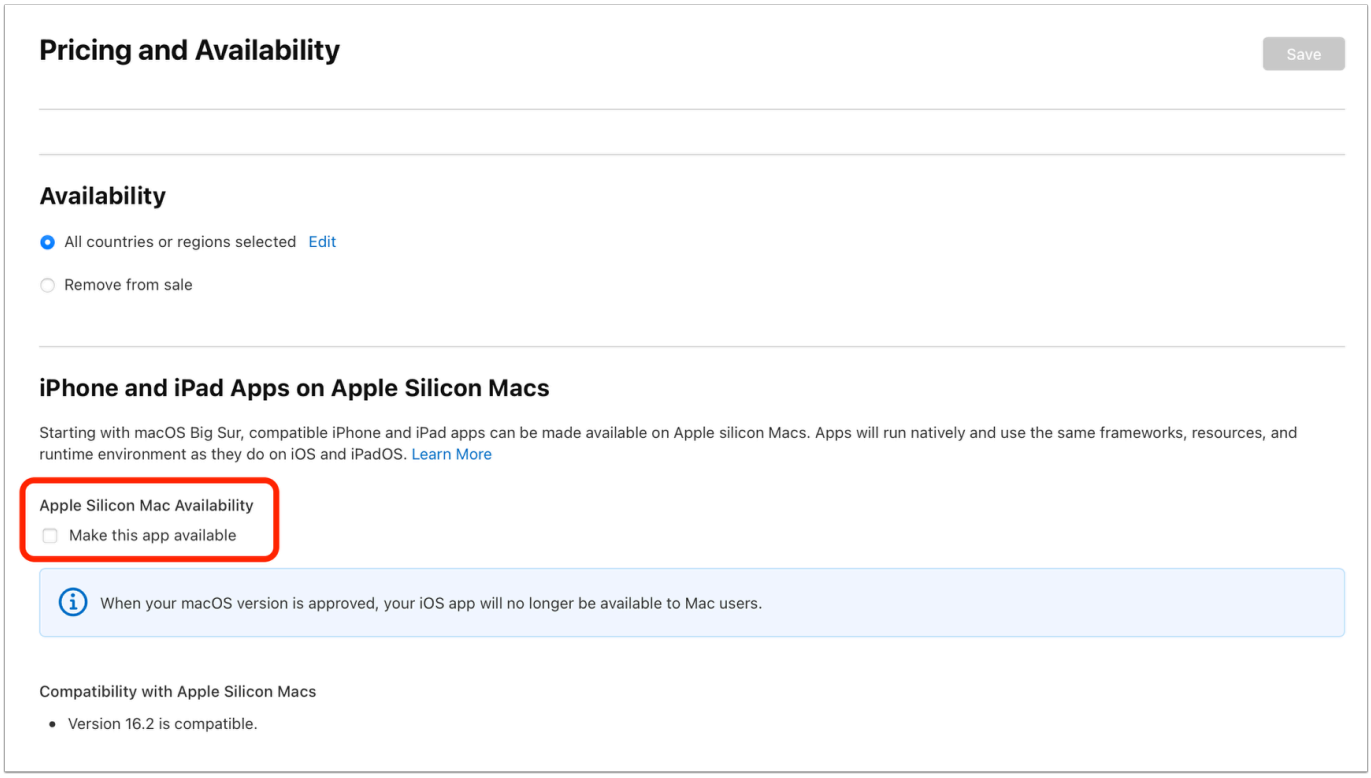
4. Submit for review

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.

5. Making your iOS app available on Apple Silicon (M1) Macs

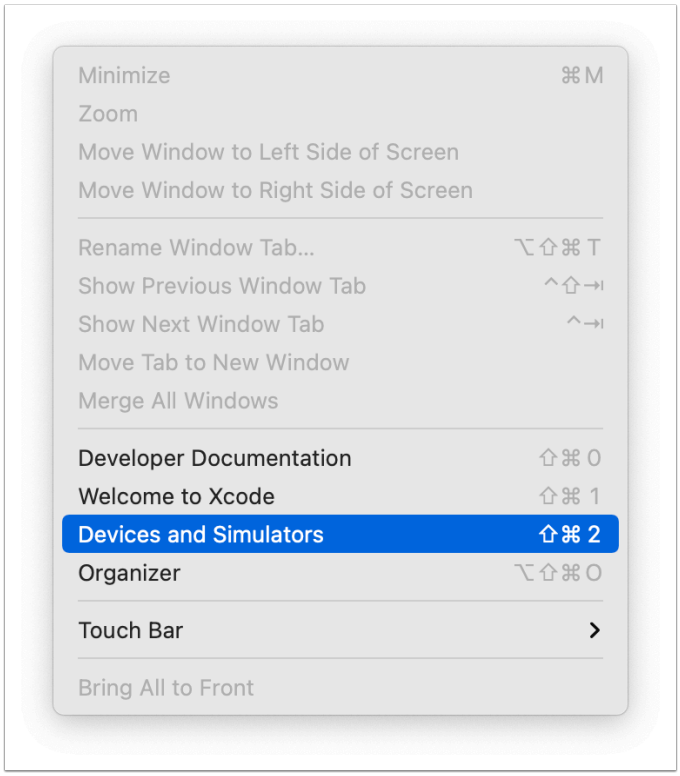
If you want to make your Twixl app available in the Mac App Store, you can now do so without any modification whatsoever. Just check the option in the settings for your app on App Store Connect, under '*Pricing and Availability*'.



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`.

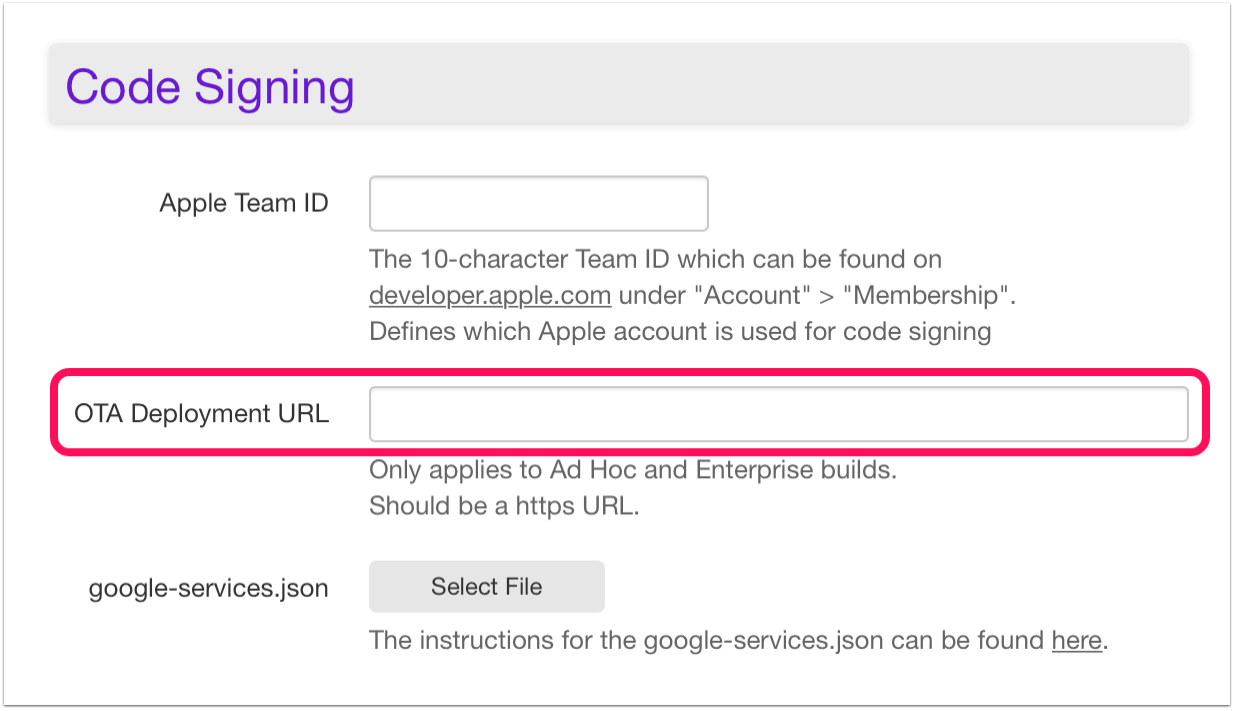
If something is wrong with the build or with your provisioning profile, you will usually get a more detailed error message.

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

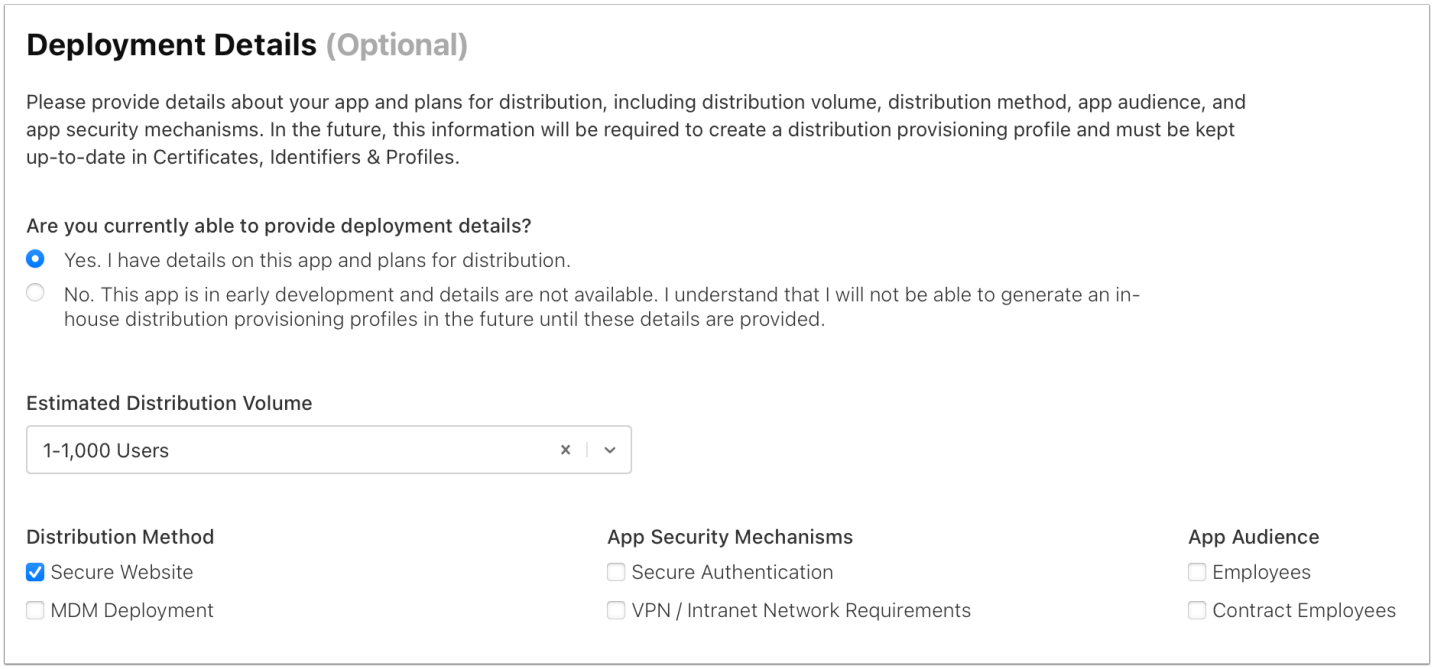
This method allows you to distribute the app more easily without the need for iTunes syncing. The `.ipa` file will be installed on a secure web server, and users will be able to install the app directly on the device.

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.



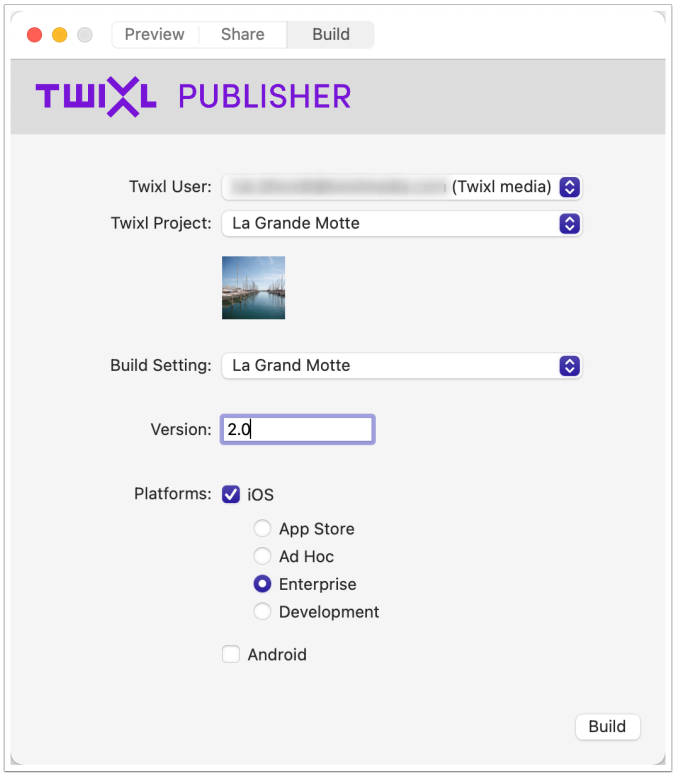
In the App ID Configuration on developer.apple.com, you'll need to check 'Secure Website' in the Deployment Details. Currently this information is optional, but this may become a requirement in the future.



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).

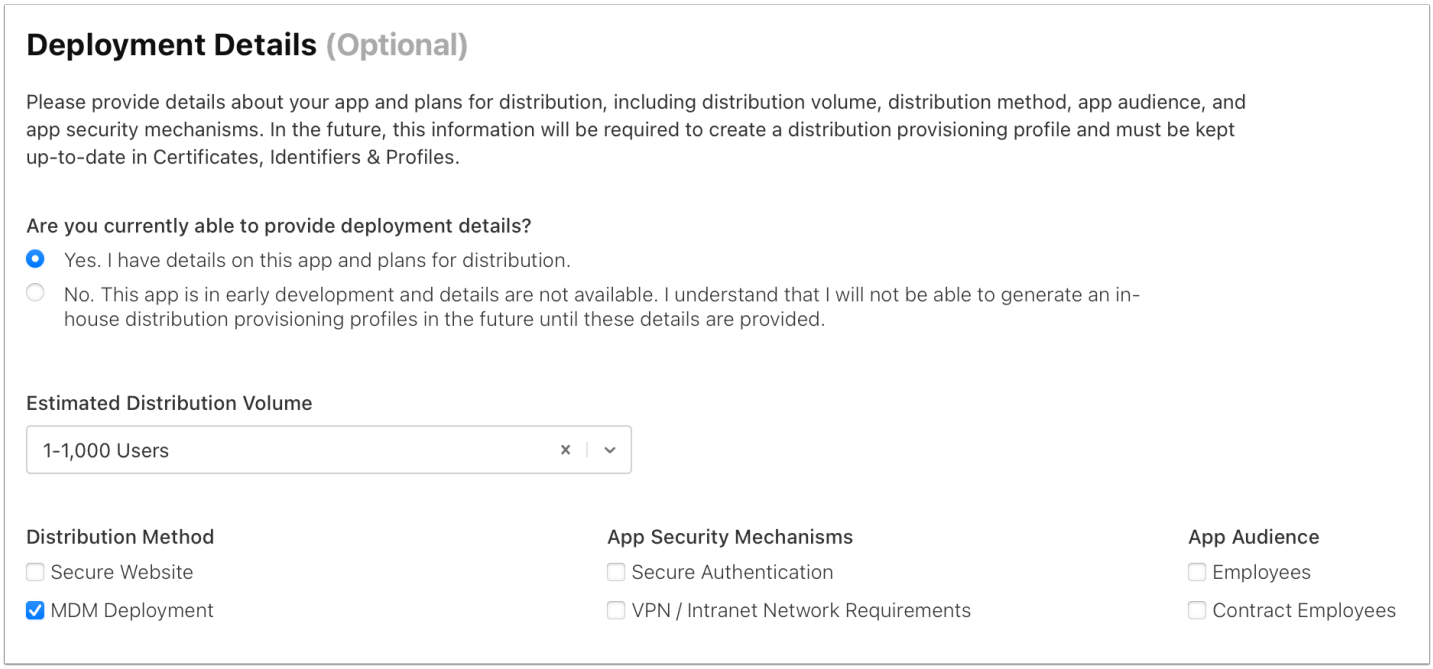
Three files will be generated when you create a build:

- `[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 the app easily without the need for iTunes syncing. 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 distribution, 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.

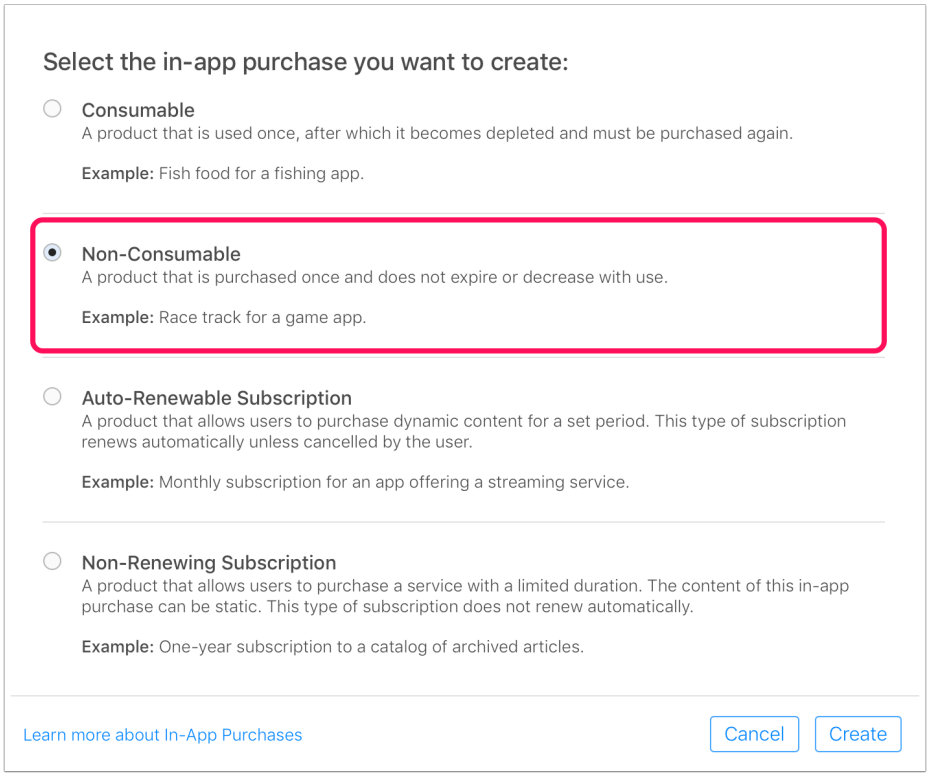
In-app purchases for iOS apps

How to configure in-app purchases for your Twixl app.

1. Adding in-app purchases in AppStore Connect

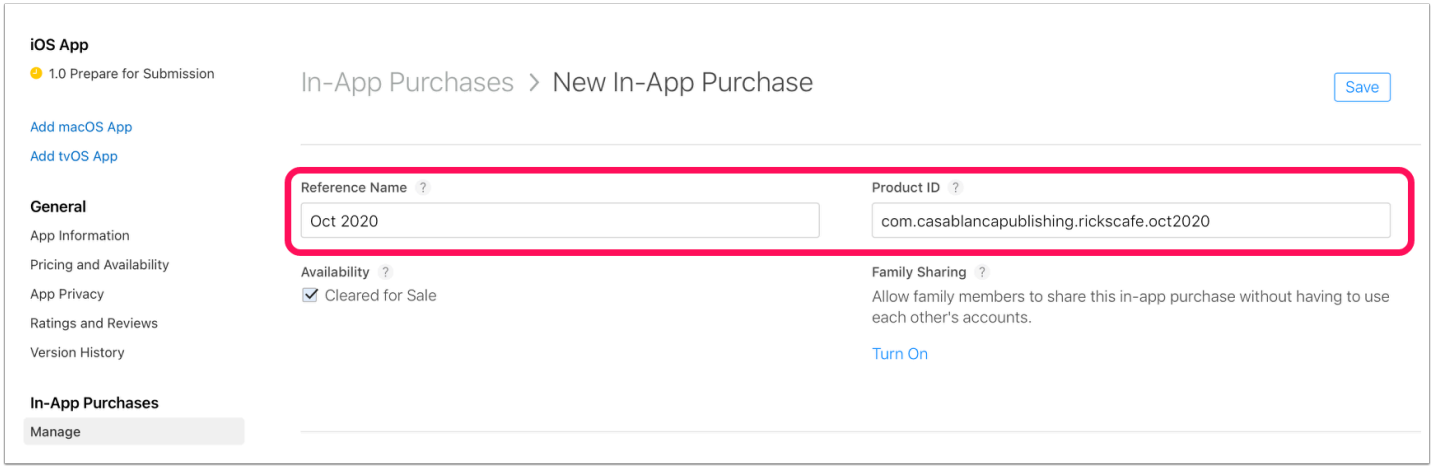
In your app, contain may be offered for free, or some or all content may be available as an in-app purchase.

- If your app offers only free content, there's nothing else to configure but adding your content to the Twixl platform.
- If you want to make collections or PDF content items available as an in-app purchase, you will need to create a separate in-app purchase entry in AppStore Connect for each of the collections you want to offer. Note that the in-app purchase type for collections needs to be 'Non-Consumable'.



1. Enter a reference name (that will be used for reporting purposes only), e.g. October 2020
2. In the Product ID field, enter a unique identifier that will be used for reporting. It can be composed of letters and numbers. Usually this will also be a reverse DNS name like the app identifier: e.g. `com.casablancapublishing.rickscafe.oct2020`
3. Set pricing for the collection by selecting a price tier

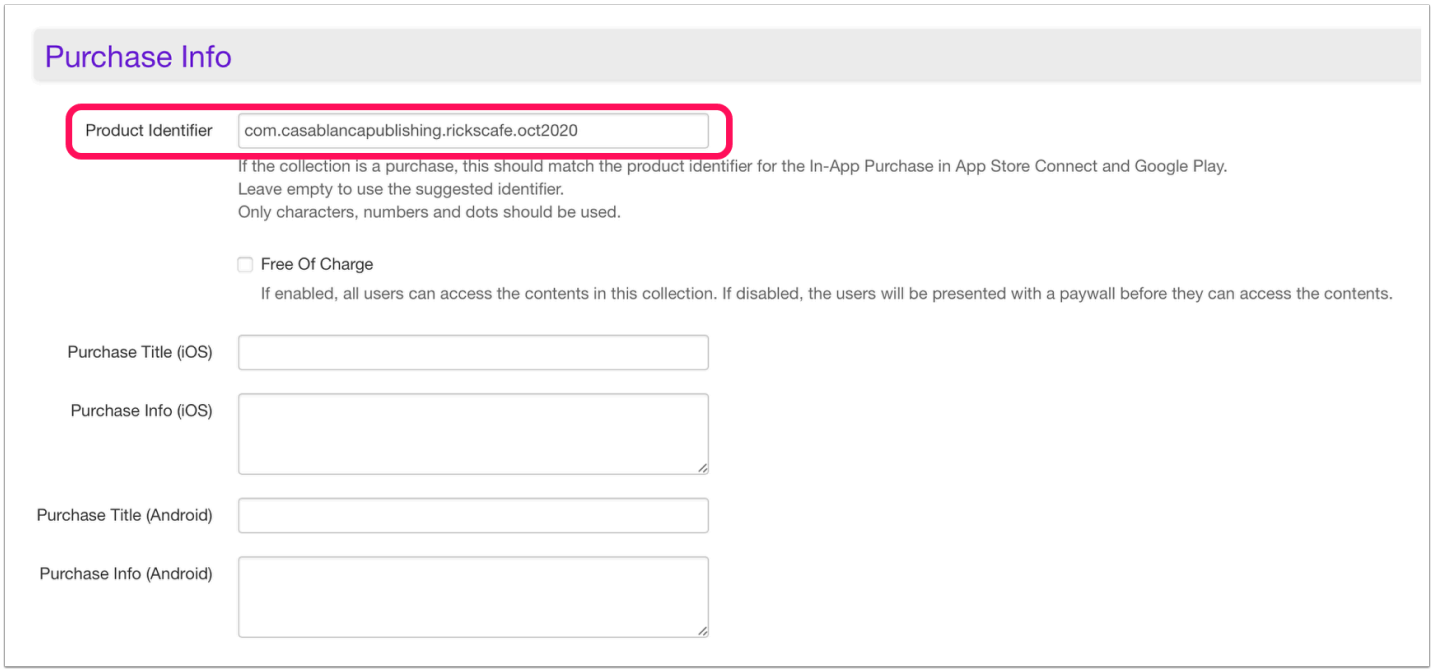
4. Provide in-app purchase information for at least one language.



2. Adding In-App Purchases

You can either add Collections or PDF Content Items as in-app purchases. For each of the in-app purchases defined in AppStore Connect, you also need to add a collection or PDF on the Twixl platform, and add the same identifier under 'Purchase Info' for your collection/PDF.

You can also add the title and extra info that will be displayed in the paywall.



IMPORTANT:

Make sure the product identifier in the Twixl platform matches exactly with the one you defined in AppStore Connect.

3. In-app purchase testing

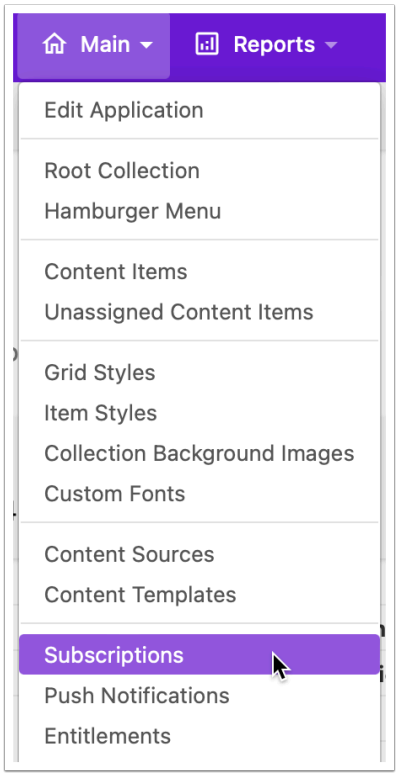
In order to test in-app purchases with an Ad Hoc build of your app, Apple provides a sandboxing environment. To do this:

1. Build and install an Ad Hoc version of your app.
2. Create a test user in AppStore Connect.
3. Logout from the App Store for your regular account.
4. Try to make a purchase and login with the test user you created.
5. You should be able to make the test purchase in the testing ('sandbox') environment.

4. Adding subscriptions in your app

For more info about this, see Working with subscriptions for iOS & Android

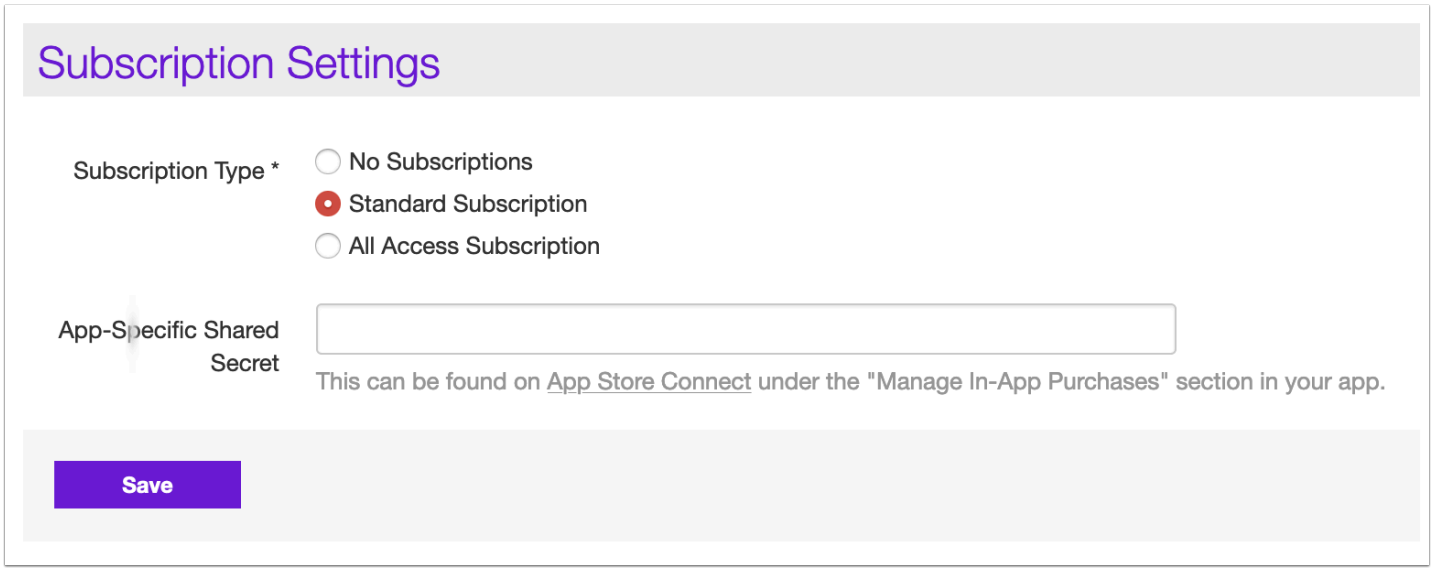
Subscriptions for iOS & Android



1. Standard or all access subscriptions

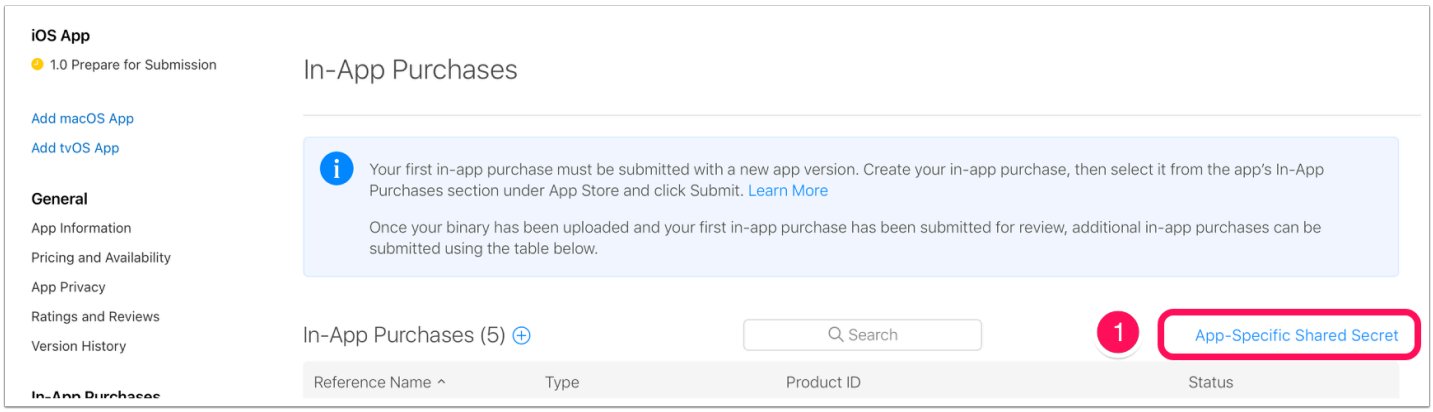
If you want to offer a subscription in your app, there are two options:

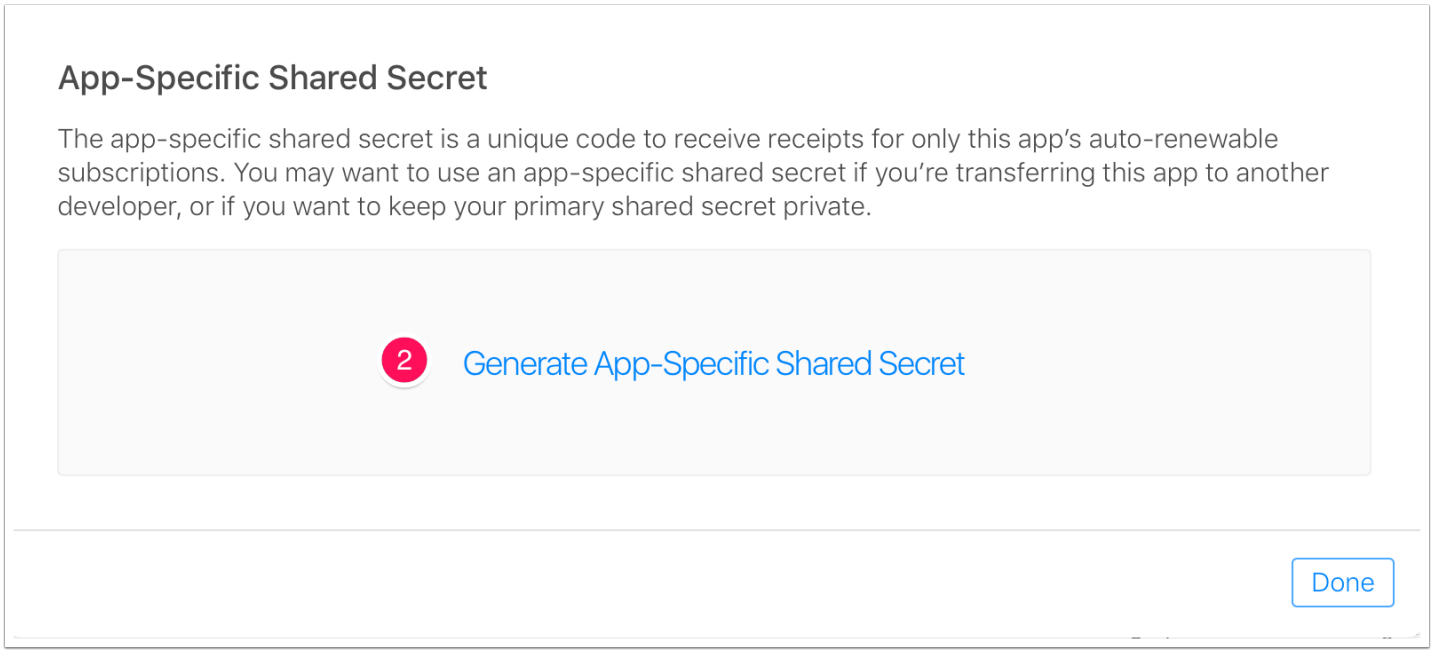
- **Standard Subscription**: this is the traditional type of subscription where you get access to new collections or PDF content items (i.e. issues) that are added during the period of your subscription.
- **All Access Subscription**: this can be compared to e.g. Spotify or Apple Music type of subscriptions: you get access to all content in the app, as long as your subscription remains valid.

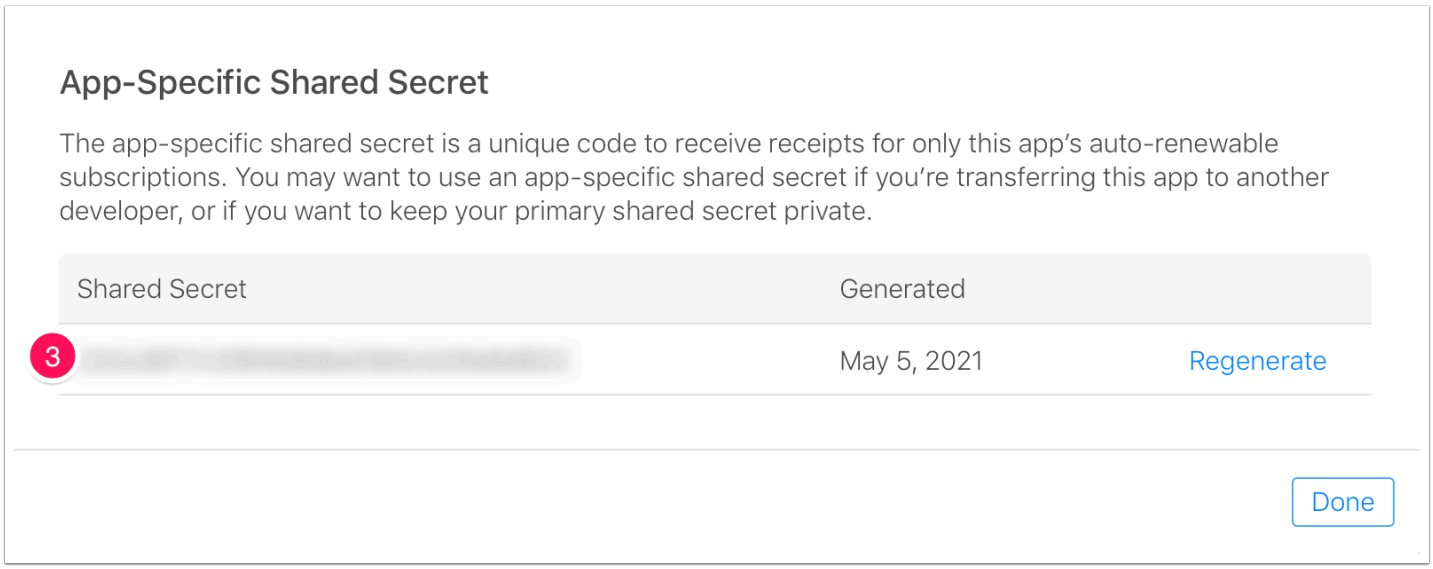


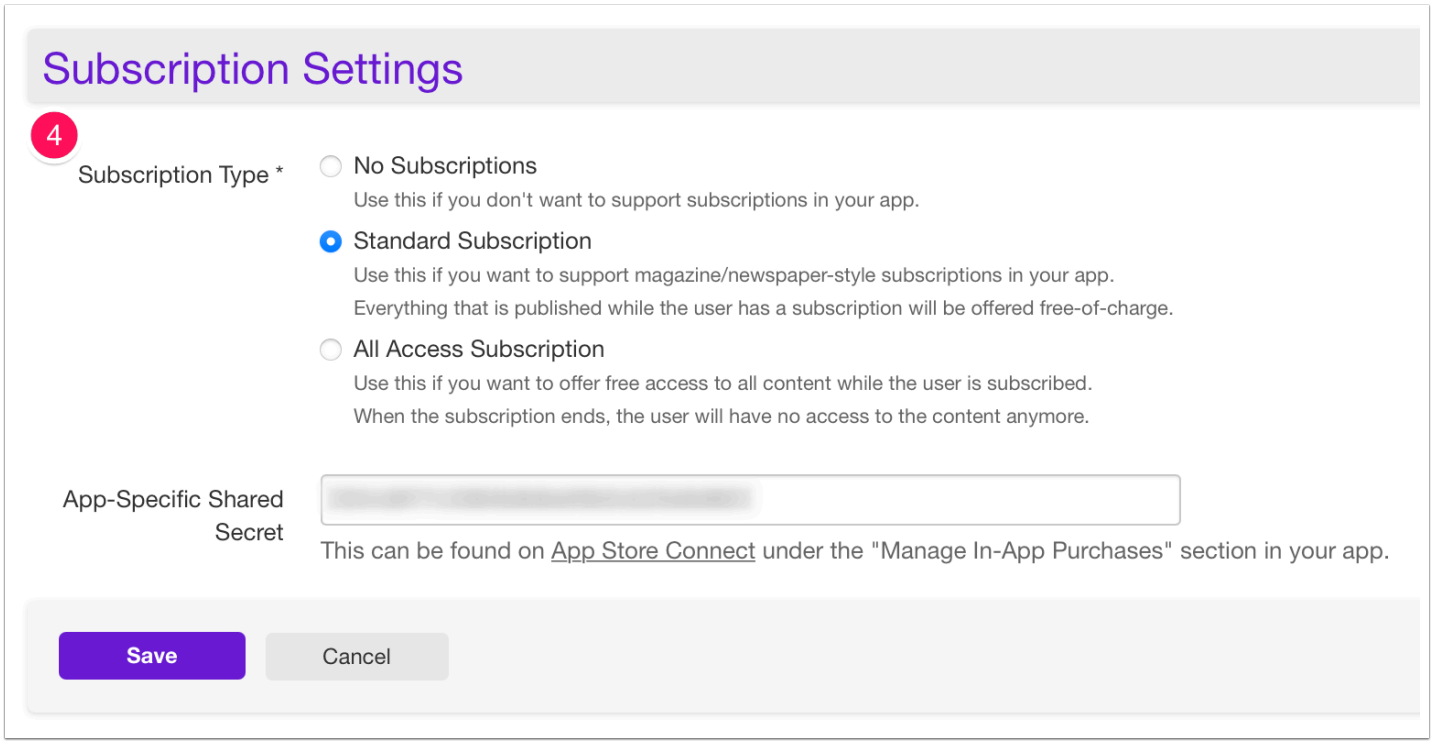
1.1. Generate App-Specific Shared Secret (iOS only)

1. Go to https://appstoreconnect.apple.com -> My Apps -> Select your app -> Manage In App Purchases -> Select '**App-Specific Shared Secret**'.
2. Click '**Generate App-Specific Shared Secret**'.
3. Copy the Shared Secret.
4. On the Twixl platform, select **Subscriptions** from the app's menu.
5. Edit the subscription settings, select the type you want to offer, and paste the shared secret in the corresponding field.




App-Specific Shared Secret
The app-specific shared secret is a unique code to receive receipts for only this app's auto-renewable subscriptions. You may want to use an app-specific shared secret if you're transferring this app to another developer, or if you want to keep your primary shared secret private.
2
Generate App-Specific Shared Secret
Done


App-Specific Shared Secret
The app-specific shared secret is a unique code to receive receipts for only this app's auto-renewable subscriptions. You may want to use an app-specific shared secret if you're transferring this app to another developer, or if you want to keep your primary shared secret private.
Shared Secret
Generated
3
May 5, 2021
Regenerate
Done



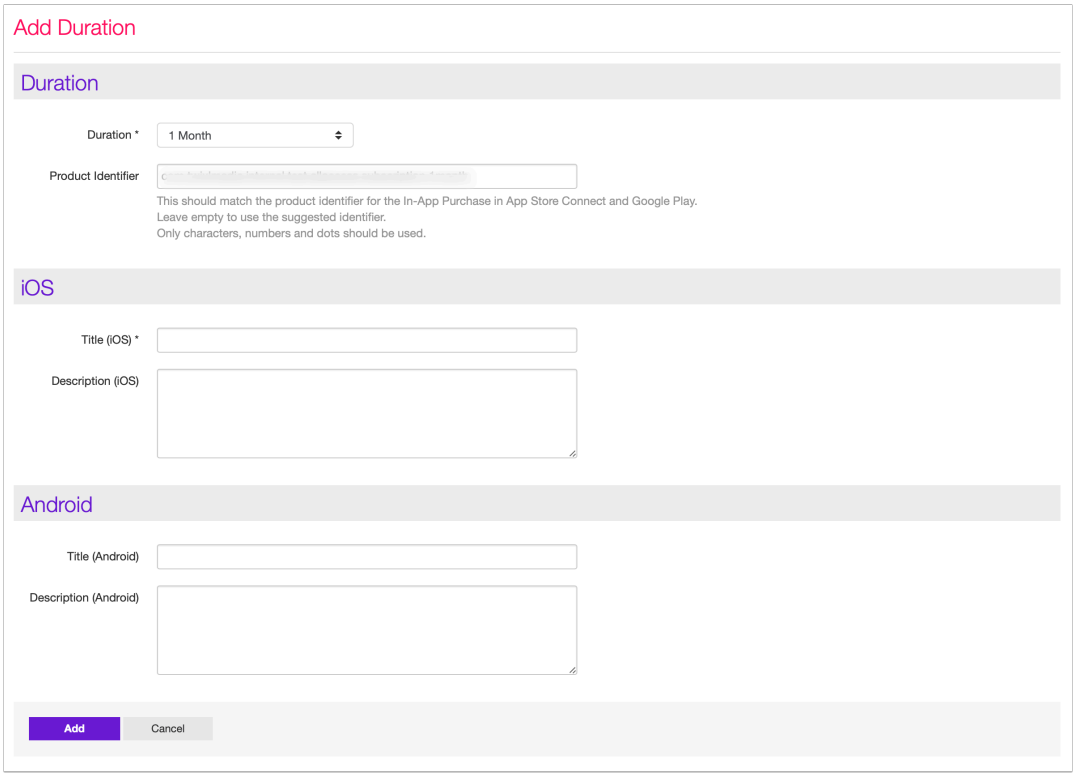
1.2. Latest content

New subscribers will automatically receive the most recent paid collection or PDF added to your app (for standard subscriptions only).

2. Adding a duration

To create a duration for an auto-renewable subscription, you need to specify the following details:

- **Duration:** The interval of the subscription. On iOS, it is either 1 week, 1 month, 2 months, 3 months, 6 months or one year.
- **Product identifier:** A unique reverse-DNS for this duration. This product identifier has to be the same as the one entered in AppStore Connect / Google Play Developer Console, for example `com.mycompany.mymagazine.subscription.1month`
- **Title (iOS & Android):** The name a reader will see in the list of subscriptions on the device.
- **Description (iOS & Android):** The description a reader will see in the list of subscriptions of the device.



3. Adding a subscription

3.1. In App Store Connect (iOS)

1. Go to https://appstoreconnect.apple.com -> **Manage Your Application** -> **Select your application** -> **Manage In-App Purchases** -> **Create New**
2. Choose the correct type of subscription (usually Auto-Renewable) and create a new subscription family or select an existing subscription family.
3. Just like on the Twixl platform, you have to add a duration for the subscription. You need to use the same product identifier & matching duration as on the Twixl platform, for example `com.mycompany.mymagazine.subscription.1month`

3.2. In the Google Play Developer Console (Android)

1. Go to https://play.google.com -> navigate to the In-App Products for your app.
2. Create the same subscription with the same product identifier & matching duration as on the Twixl platform, for example
   `com.mycompany.mymagazine.subscription.1month`

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/

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

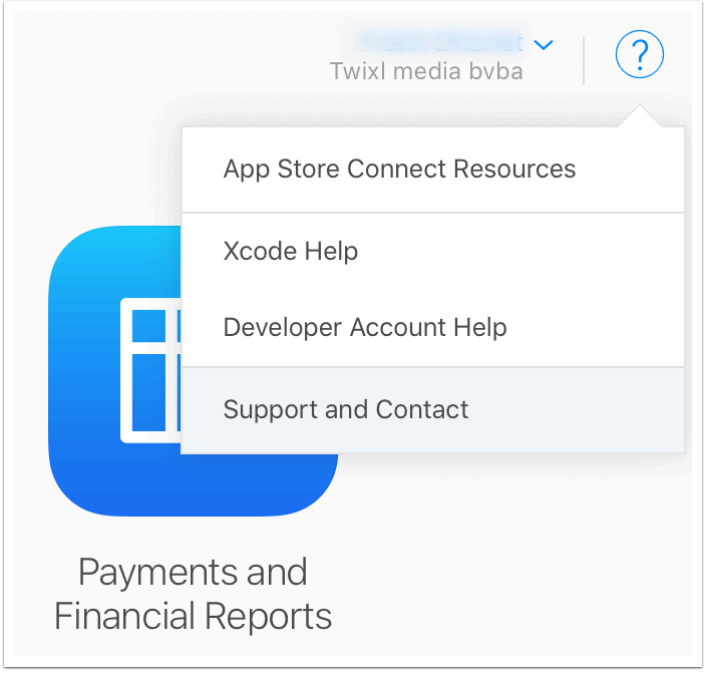
Asking Apple for an "expedited" review

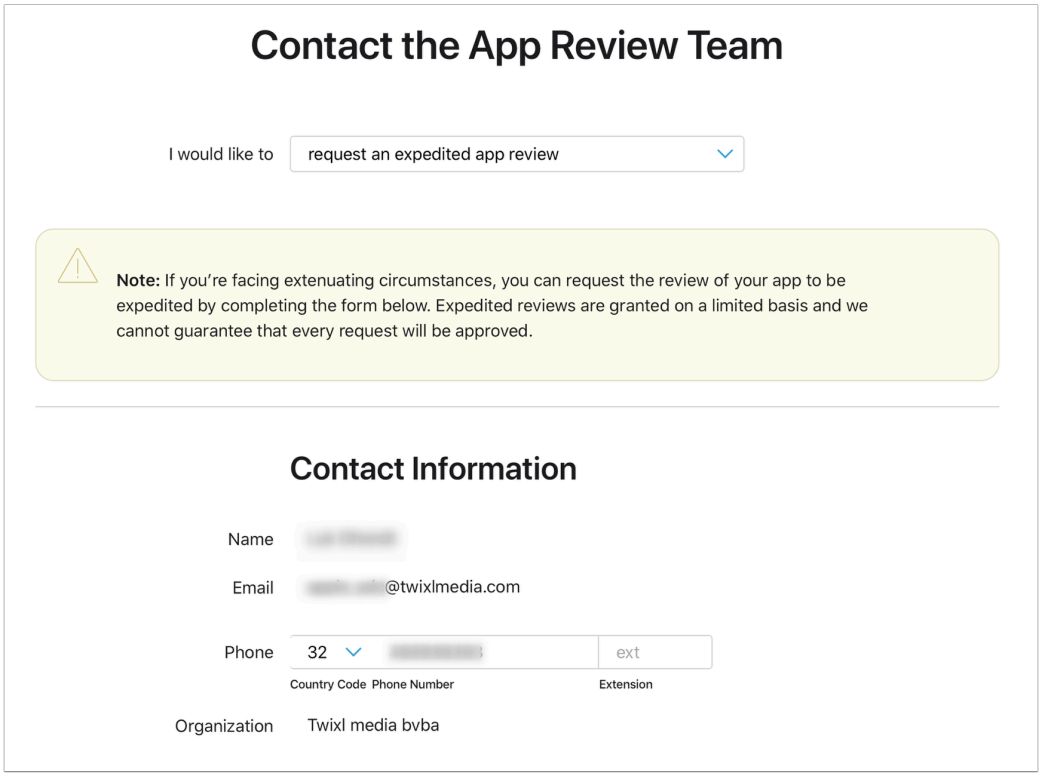
If you ever run into a problem with an app you published in Apple's App Store, and want to update it as fast as possible, you can ask the Apple review team for an "expedited review" explaining the problem.

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

Android Specific

Deploying an Android test build

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

1. Preview on a device

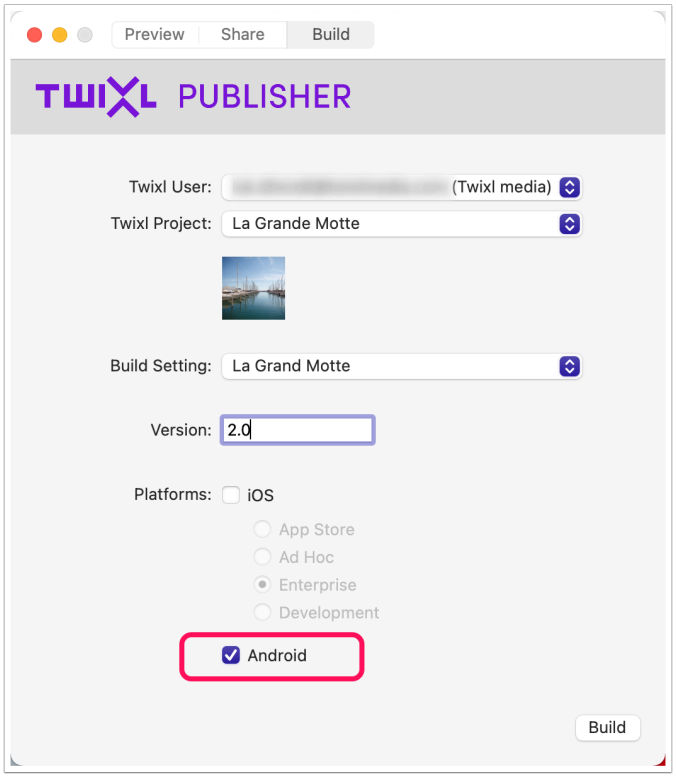
Preview on a device is the quickest way to get content on your tablet or phone, and allows you to preview in the Twixl app.

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.

Deploying an Android app in Google Play

> ⚠ **IMPORTANT INFO ABOUT GOOGLE PLAY APP SIGNING**
>
> Read this important information about Google Play App Signing here, before you start uploading apps to Google Play.

1. Requirements for Android apps

- Android Developer Account
- Artwork

2. Uploading to 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).

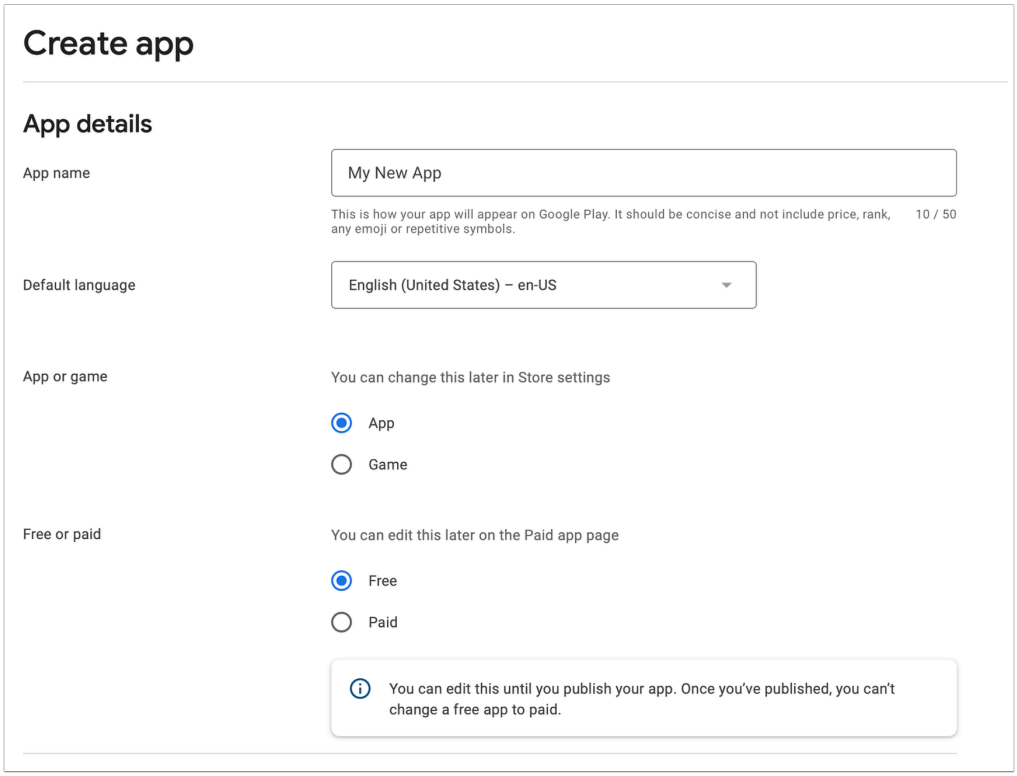
`.apk` : this is the original extension that will become deprecated as of August 2021 for new apps, and November 2021 for updates of existing apps.

`.aab` : this is the new format that will be required as of August 2021 for new apps, and November 2021 for updates of existing apps.

We recommend you move to the Android App Bundle with your next update.

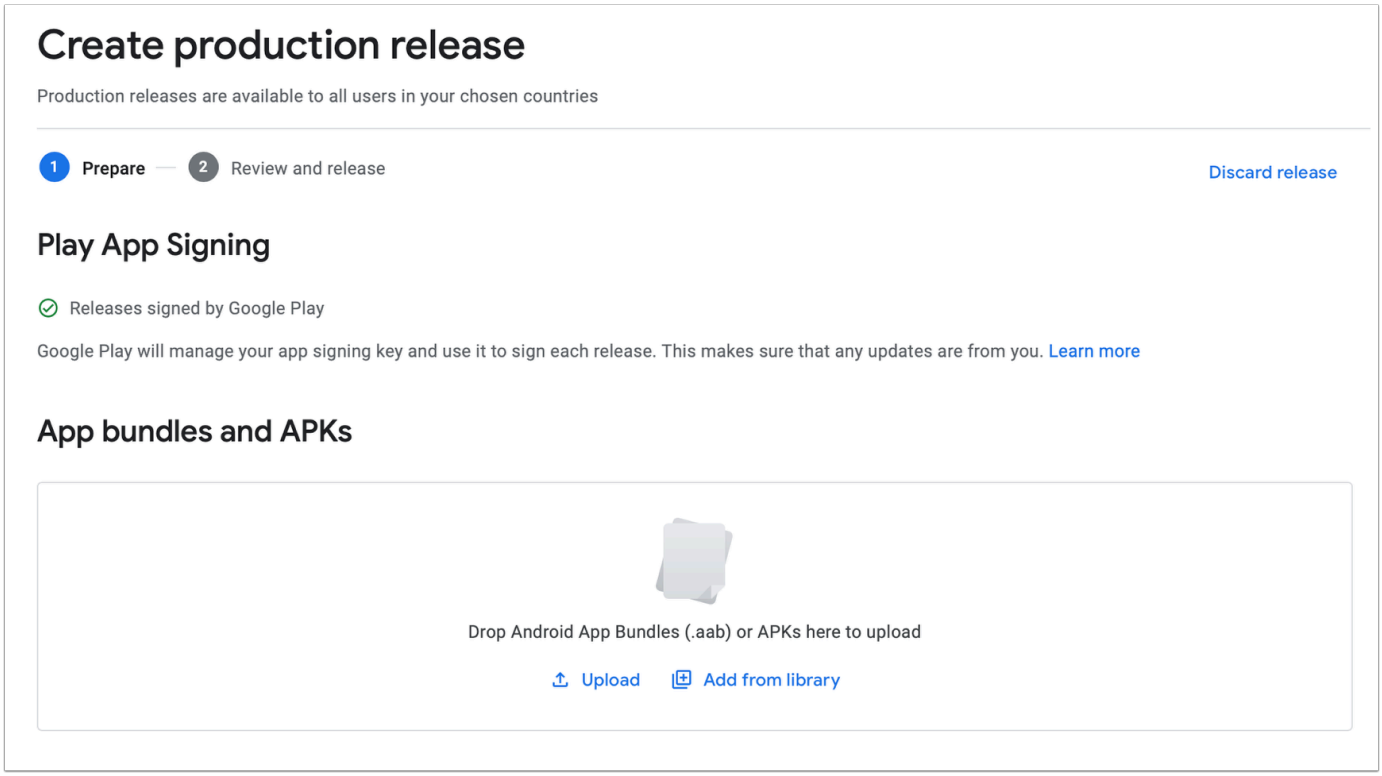
> ⚠ **.apk** remains the format that you can use for sideloading an app on an Android device, or for in-house distribution.

2.1. Creating a new application



When you login to your Android Developer Account, select '**Create Application**'.

2.2. Create New Release



In the sidebar, under 'Release', select 'Production', then click 'Create New Release'.

Now you can upload your `.aab` (or `.apk`) file.

3. Add artwork

When the file has been uploaded, you should also upload your artwork. A couple of things are required:

2 application screenshots:

These should be any of the following sizes:

- 320 x 480, 480 x 800, 480 x 854, 1280 x 720, 1280 x 800
- 24-bit PNG or JPEG (no alpha)

High Resolution Application Icon:

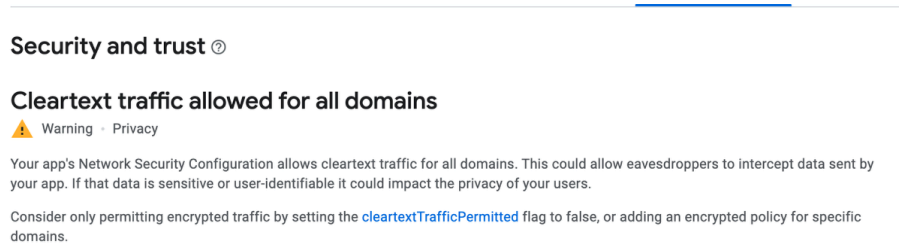
- 512 x 512 pixels
- 32-bit PNG or JPEG
- Maximum: 1024 Kb

The final steps to deploy your application to Google Play are: click '**Publish**' and '**Activate**' your application. It can take a couple of hours before your app will become available in the Google Play store.

4. Android Keystore

For more details about the Android Keystore, please check this article. If you are migrating your app from another platform, check out this paragraph specifically.

> ⚠ Note that 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.

In-app purchases for Android apps

How to configure in-app purchases for your app.

1. Adding in-app purchases in the Google Play Developer Console

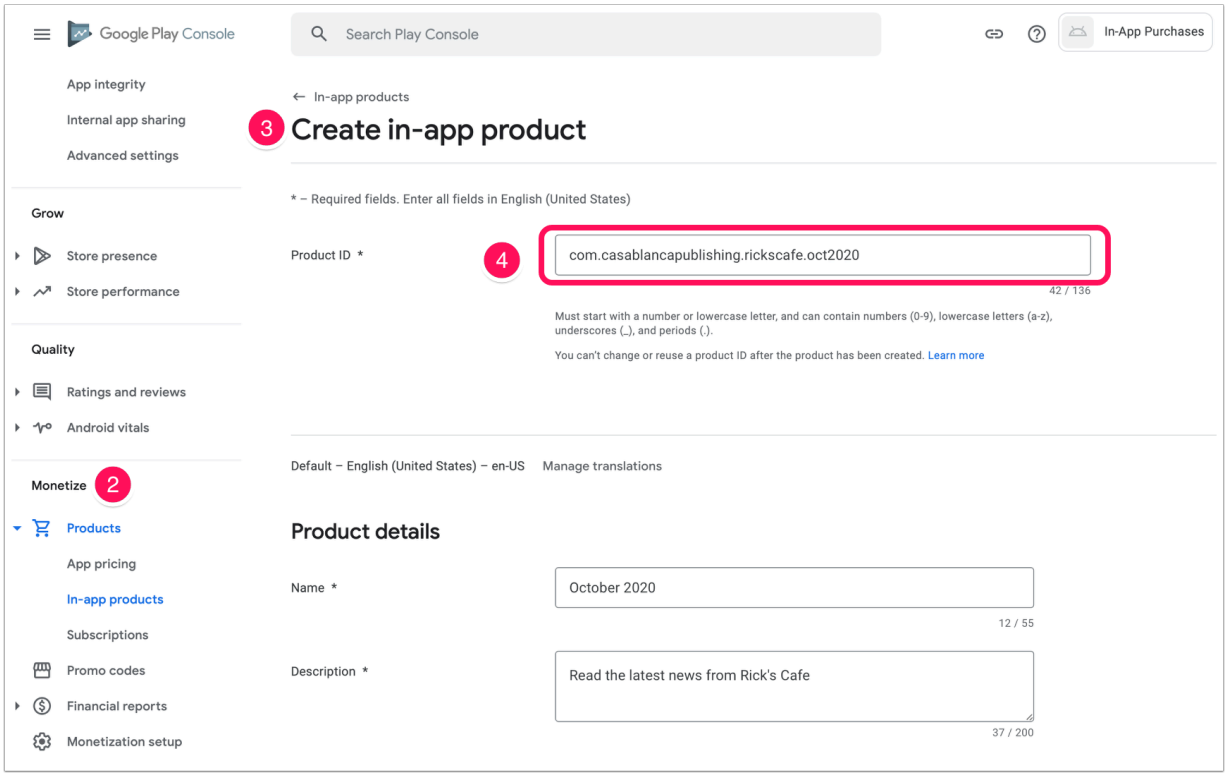
In your app, contain may be offered for free, or some or all content may be available as an in-app purchase.

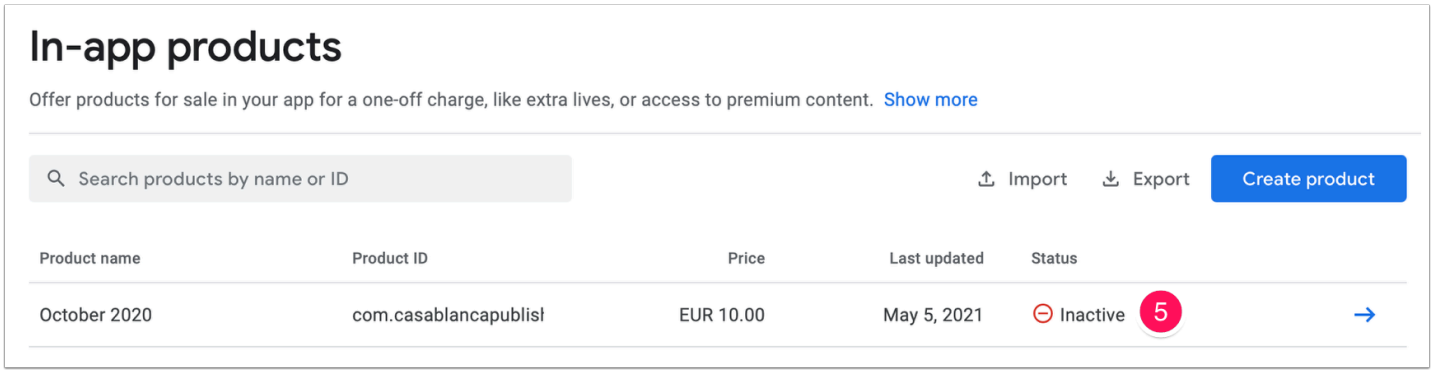
If your app offers only free content, there's nothing else to configure but adding your content to the Twixl platform.

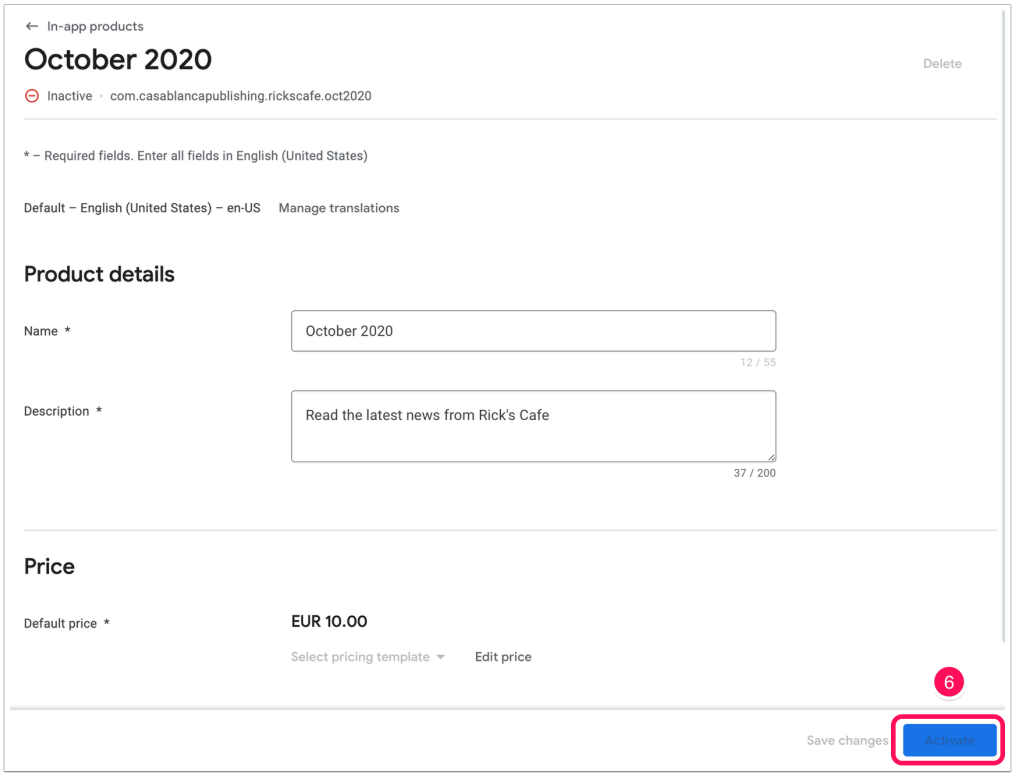
If you want to make collections or PDF content items available as an in-app purchase, you will need to create a separate in-app purchase entry for each of the collections you want to offer:

- **Google Play Developer Console** : here you will also set the price for the in-app purchase
- **Twixl platform** : here you will have your different paid collections or PDF content items.

1. Go to https://play.google.com/apps/publish and select your app from the list of apps
2. In the section **Monetize**, choose **In-app Products**.
3. Now select **Create Product**.
4. Add the **product ID**, **name** and **description**, set the price, and Save the information.
5. The in-app purchase has been saved, but it is still inactive at this point.
6. Select the in-app purchase you just created once more, and you'll see you can now activate it.


Google Play Console
Search Play Console
In-App Purchases
App integrity
Internal app sharing
Advanced settings
Grow
Store presence
Store performance
Quality
Ratings and reviews
Android vitals
Monetize
2
Products
App pricing
In-app products
Subscriptions
Promo codes
Financial reports
Monetization setup
In-app products
3
Create in-app product
* – Required fields. Enter all fields in English (United States)
Product ID *
4
com.casablancapublishing.rickscafe.oct2020
42 / 136
Must start with a number or lowercase letter, and can contain numbers (0-9), lowercase letters (a-z), underscores (_), and periods (.).
You can't change or reuse a product ID after the product has been created. Learn more
Default – English (United States) – en-US
Manage translations
Product details
Name *
October 2020
12 / 55
Description *
Read the latest news from Rick's Cafe
37 / 200

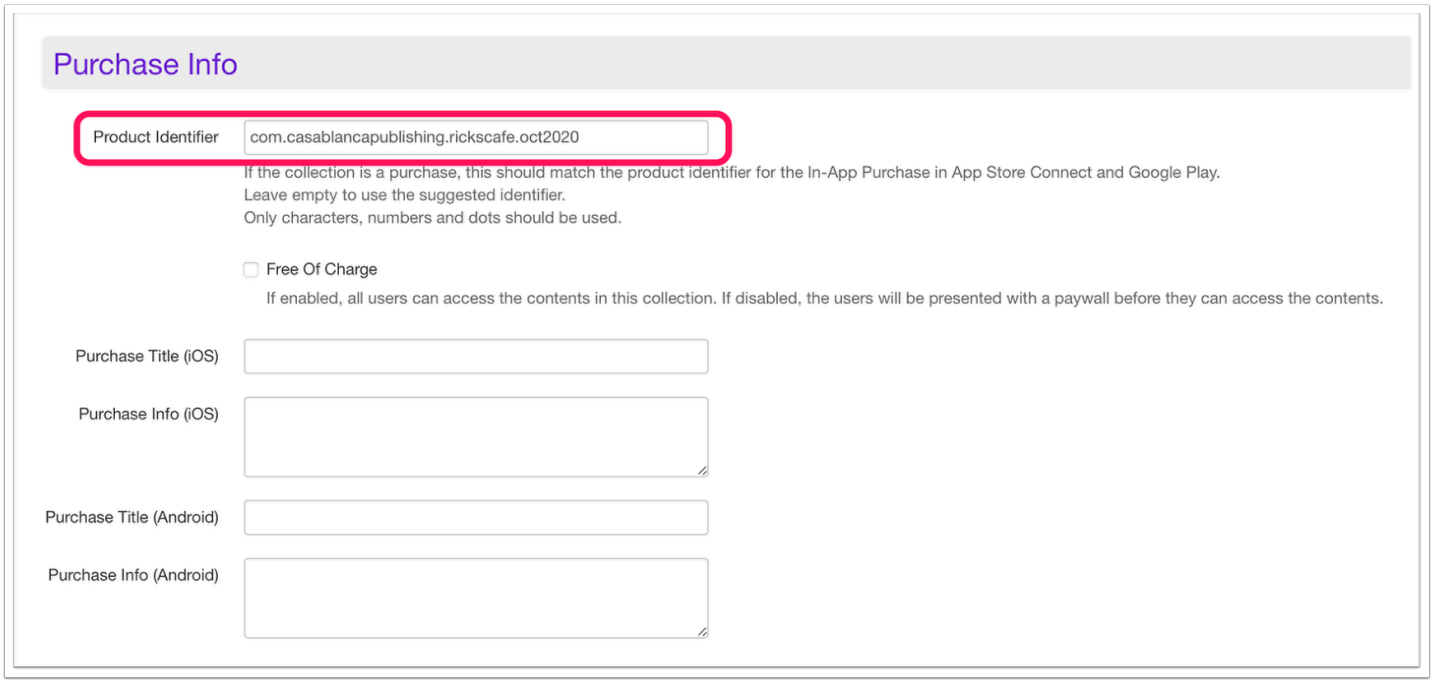

In-app products
Offer products for sale in your app for a one-off charge, like extra lives, or access to premium content. Show more
Search products by name or ID
Import
Export
Create product
Product name
Product ID
Price
Last updated
Status
October 2020
com.casablancapublisl
EUR 10.00
May 5, 2021
Inactive
5



2. Adding in-app purchases to the Twixl platform

For each of the in-app purchases defined in Google Play, you also need to have a collection or PDF content item on the Twixl platform. Add the same identifier under 'Purchase Info' for your collection/PDF.

You can also add the title and extra info (by platform) that will be displayed in the paywall.



IMPORTANT:

Make sure the product identifier in the Twixl platform matches exactly with the one you defined for the in-app Product in Google Play.

3. In-app purchase testing

After you added your in-app products for your app in Google Play, you can test your purchases:

1. Create a Google Play build of your app
2. Upload the app to Google Play as an alpha or beta
3. Create license test accounts for authorized users (in Developer Console, go to Settings > Account details, then in the License Testing section, add the addresses to the field "Gmail accounts with testing access"
4. Test your in-app purchases with one of the test accounts - these allow you to purchase any of your in-app products without being charged

IMPORTANT NOTE:

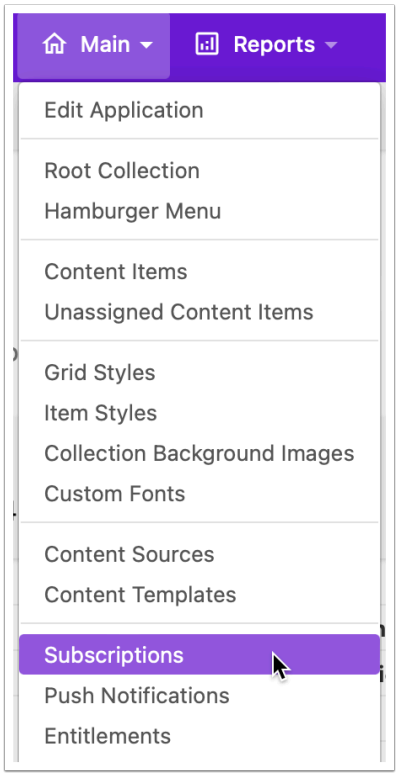
Sometimes you may get the error "This version of the application is not configured for billing through Google Play. Check the help center for more

information". Note that Google takes a while to process applications and update them to their servers (anywhere from a couple of hours to a day). So after uploading the APK on Google Play, you may have to wait a few hours before the in-app products will allow to be purchased.

4. Adding subscriptions in your app

See this article: Working with subscriptions for iOS & Android

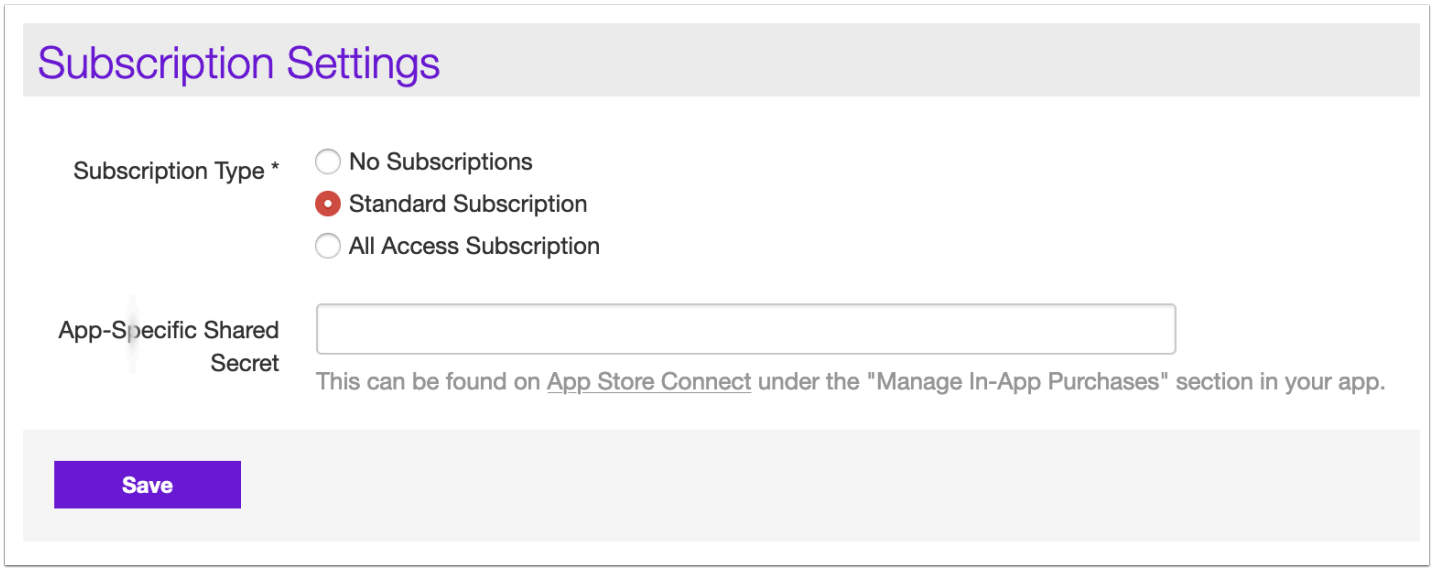
Subscriptions for iOS & Android



1. Standard or all access subscriptions

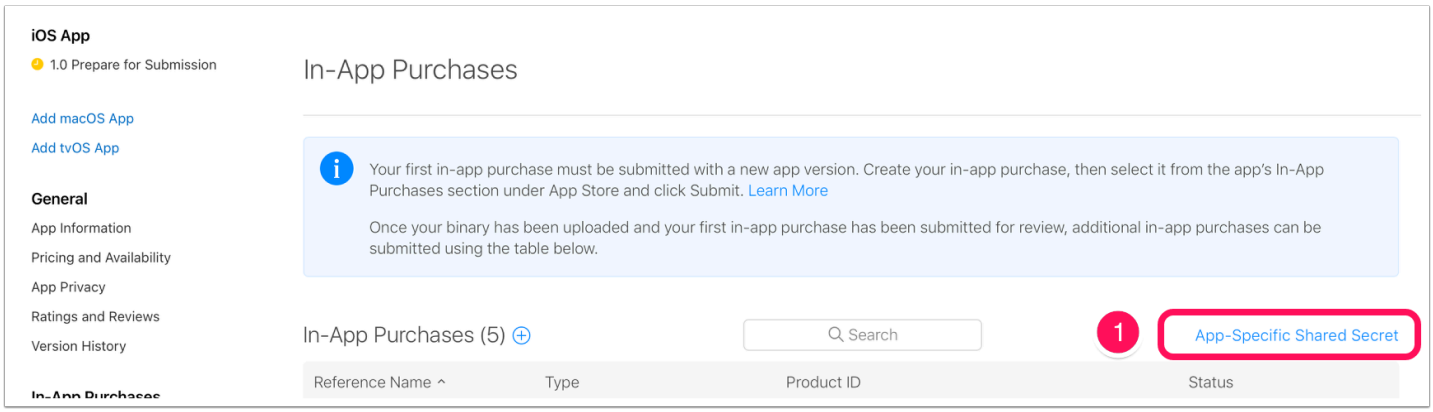
If you want to offer a subscription in your app, there are two options:

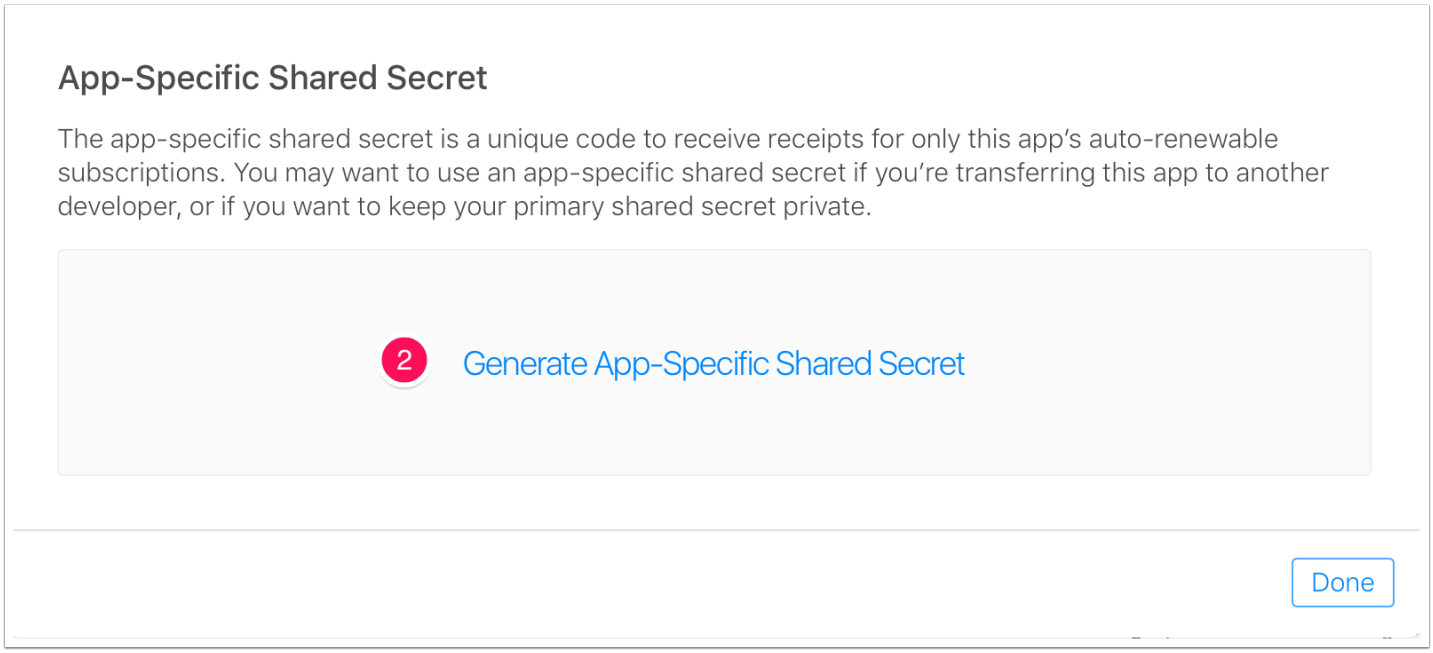
- **Standard Subscription**: this is the traditional type of subscription where you get access to new collections or PDF content items (i.e. issues) that are added during the period of your subscription.
- **All Access Subscription**: this can be compared to e.g. Spotify or Apple Music type of subscriptions: you get access to all content in the app, as long as your subscription remains valid.

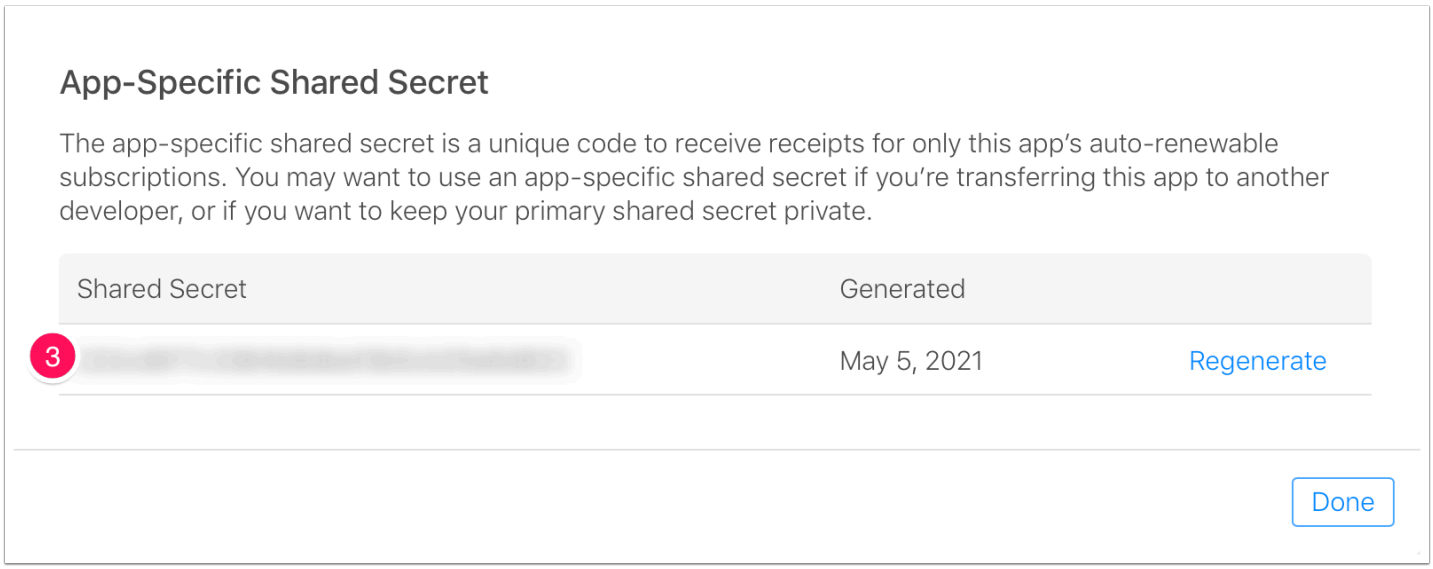


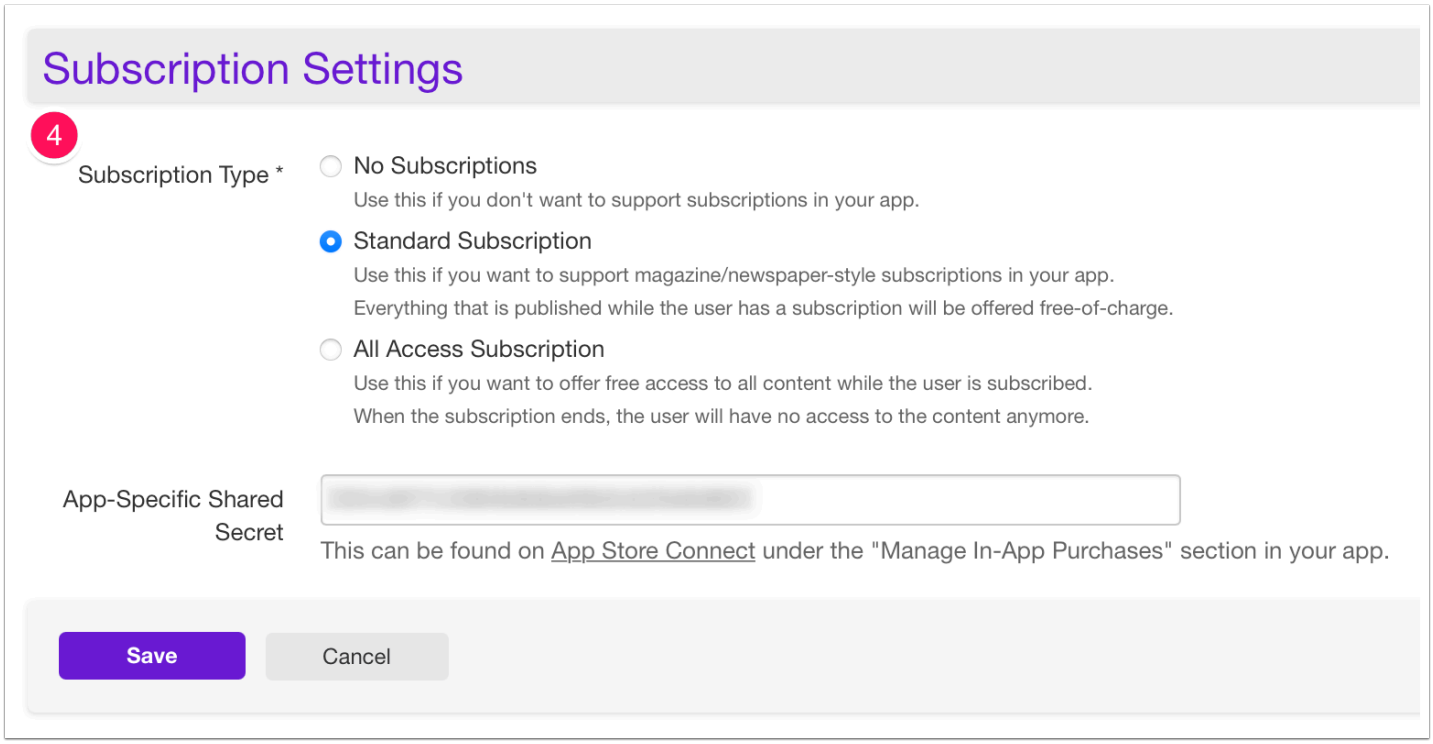
1.1. Generate App-Specific Shared Secret (iOS only)

1. Go to https://appstoreconnect.apple.com -> My Apps -> Select your app -> Manage In App Purchases -> Select '**App-Specific Shared Secret**'.
2. Click '**Generate App-Specific Shared Secret**'.
3. Copy the Shared Secret.
4. On the Twixl platform, select **Subscriptions** from the app's menu.
5. Edit the subscription settings, select the type you want to offer, and paste the shared secret in the corresponding field.




App-Specific Shared Secret
The app-specific shared secret is a unique code to receive receipts for only this app's auto-renewable subscriptions. You may want to use an app-specific shared secret if you're transferring this app to another developer, or if you want to keep your primary shared secret private.
2
Generate App-Specific Shared Secret
Done


App-Specific Shared Secret
The app-specific shared secret is a unique code to receive receipts for only this app's auto-renewable subscriptions. You may want to use an app-specific shared secret if you're transferring this app to another developer, or if you want to keep your primary shared secret private.
Shared Secret
Generated
3
May 5, 2021
Regenerate
Done



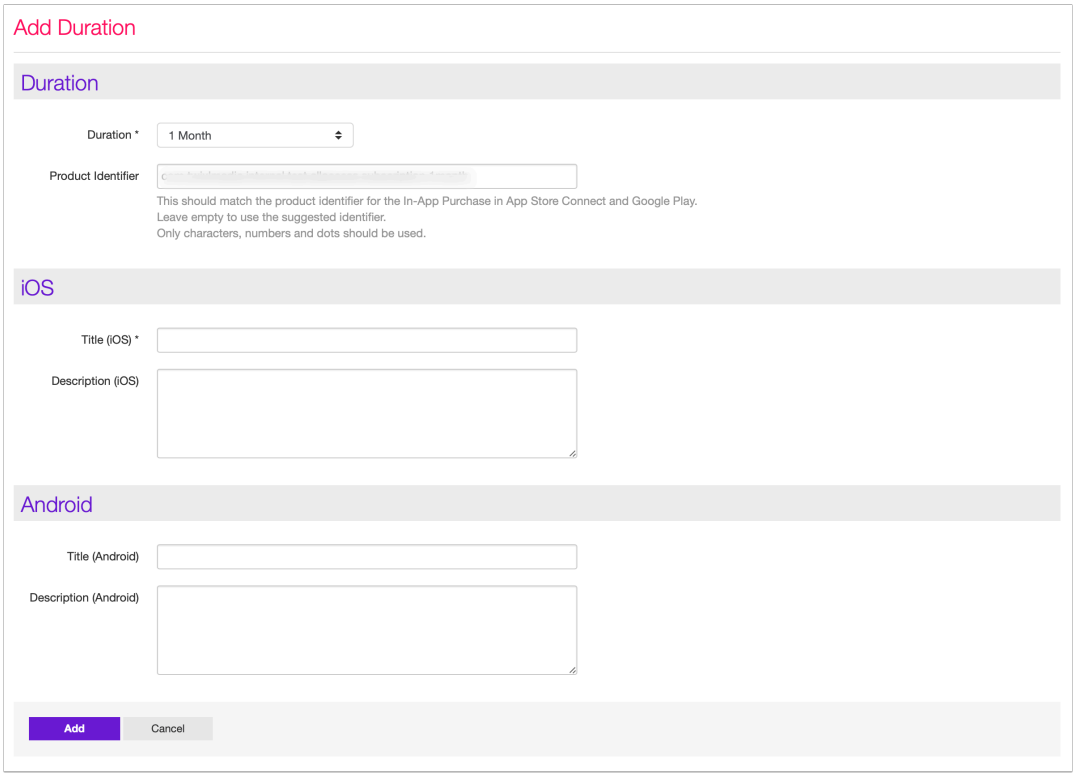
1.2. Latest content

New subscribers will automatically receive the most recent paid collection or PDF added to your app (for standard subscriptions only).

2. Adding a duration

To create a duration for an auto-renewable subscription, you need to specify the following details:

- **Duration:** The interval of the subscription. On iOS, it is either 1 week, 1 month, 2 months, 3 months, 6 months or one year.
- **Product identifier:** A unique reverse-DNS for this duration. This product identifier has to be the same as the one entered in AppStore Connect / Google Play Developer Console, for example `com.mycompany.mymagazine.subscription.1month`
- **Title (iOS & Android):** The name a reader will see in the list of subscriptions on the device.
- **Description (iOS & Android):** The description a reader will see in the list of subscriptions of the device.



3. Adding a subscription

3.1. In App Store Connect (iOS)

1. Go to https://appstoreconnect.apple.com -> **Manage Your Application** -> **Select your application** -> **Manage In-App Purchases** -> **Create New**
2. Choose the correct type of subscription (usually Auto-Renewable) and create a new subscription family or select an existing subscription family.
3. Just like on the Twixl platform, you have to add a duration for the subscription. You need to use the same product identifier & matching duration as on the Twixl platform, for example `com.mycompany.mymagazine.subscription.1month`

3.2. In the Google Play Developer Console (Android)

1. Go to https://play.google.com -> navigate to the In-App Products for your app.
2. Create the same subscription with the same product identifier & matching duration as on the Twixl platform, for example `com.mycompany.mymagazine.subscription.1month`

Google Play App Signing

> ⚠ **IMPORTANT:**
>
> **Google Play App Signing** support has been added as of **Twixl Publisher 15.4**. This means that in order to enable this for your Android app, you'll need to create a build using 15.4 or higher.

1. About "Google Play App Signing"

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 secure way to store your app signing key that helps protect you if your key is ever lost or 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.

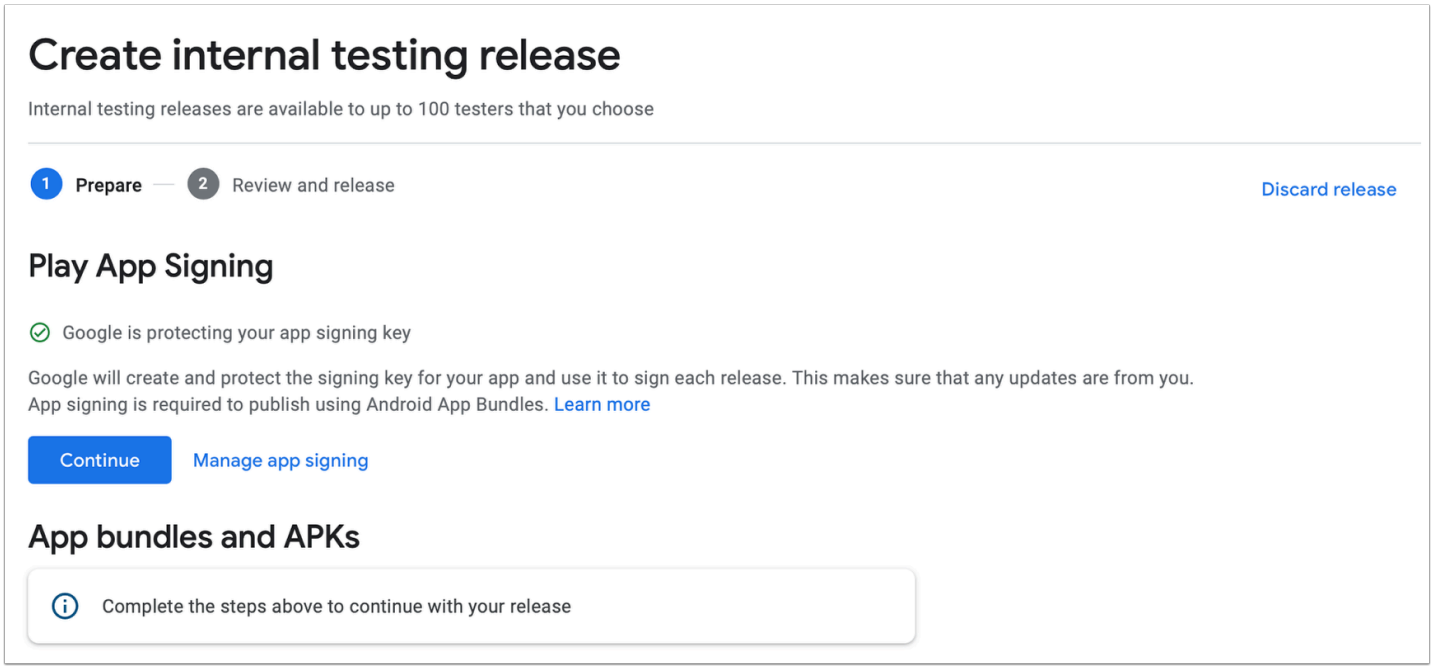
With Google Play App Signing, you can securely manage your app signing keys for new or existing apps. Keys are stored on the same secure infrastructure Google uses to store its own keys.

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 lose your keystore, you'll need to publish a new app with a new package name..

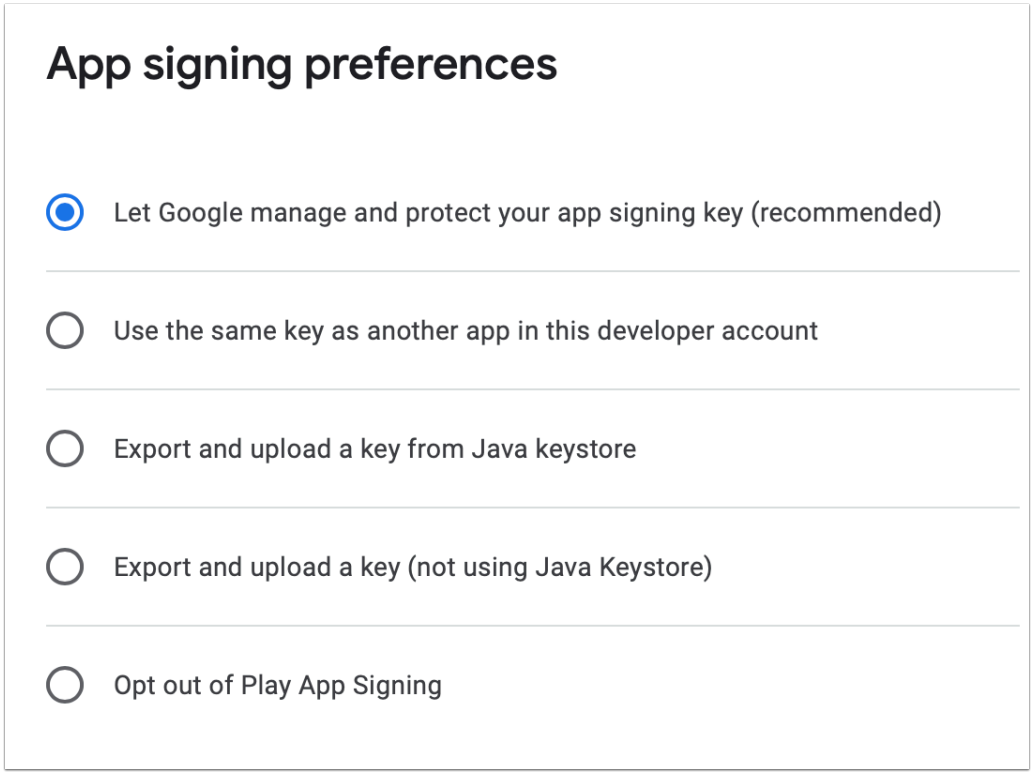
2. How to opt in to app signing for a new app

1. Open Play Console.
2. Select your app.
3. On the left menu, go to Release > Setup > App integrity.

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

3.1. Enroll in Play App Signing

1. Open Play Console.
2. Select your app.
3. On the left menu, go to Release > Setup > App integrity.
4. If you haven't already, review the Terms of Service and select Accept.

3.2. Send your key to Google

1. Locate your original app signing key. This is the file called 'app_signing_private_key.pepk' that is included in the build folder of your Android app (as of **Twixl Publisher 15.4** or higher).
2. Open Play Console.
3. Select an app.
4. On the left menu, go to Release > Setup > App integrity.
5. Select the export and upload option that best suits your release process and upload an the app signing key.

About the new Google Play Console

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

Don't use it (yet)!

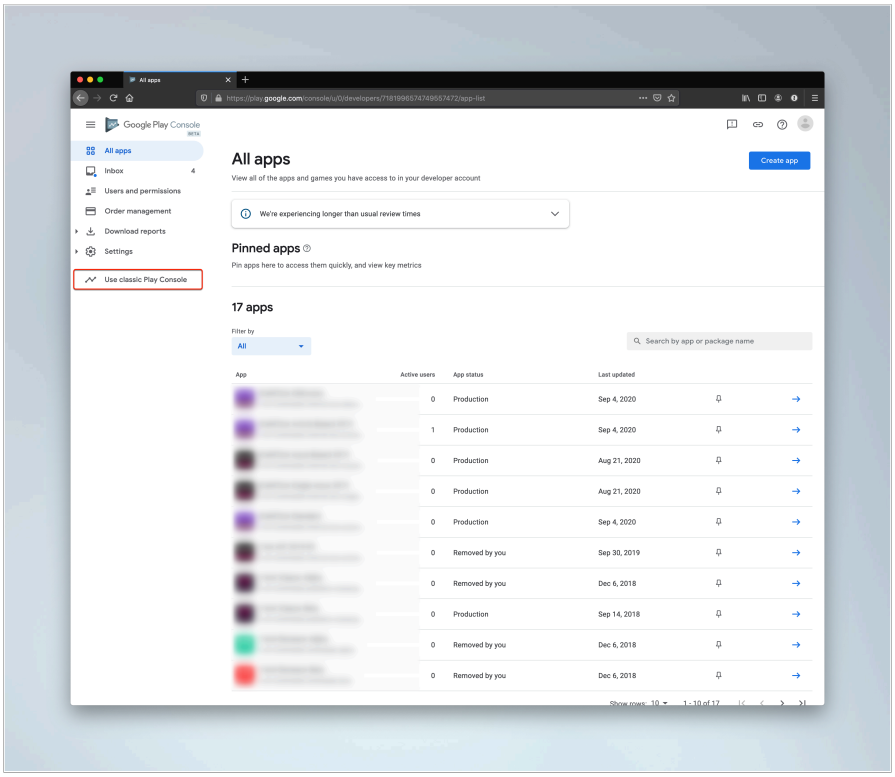
The new Google Play Console is supposed to introduce a new style and an improved experience, but there are just too many problems in this beta:

- 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: