Access to your content

Table of Contents

Subscriptions

Subscriptions for iOS & Android

You can limit access to your content so only users with subscriptions purchased in the App or Play Store can see the content. Twixl Publisher offers 2 types of subscriptions: Standard and All Access Subscriptions. Non-subscribed users clicking on content that requires a subscription, will see the paywall where they can buy a subscription.

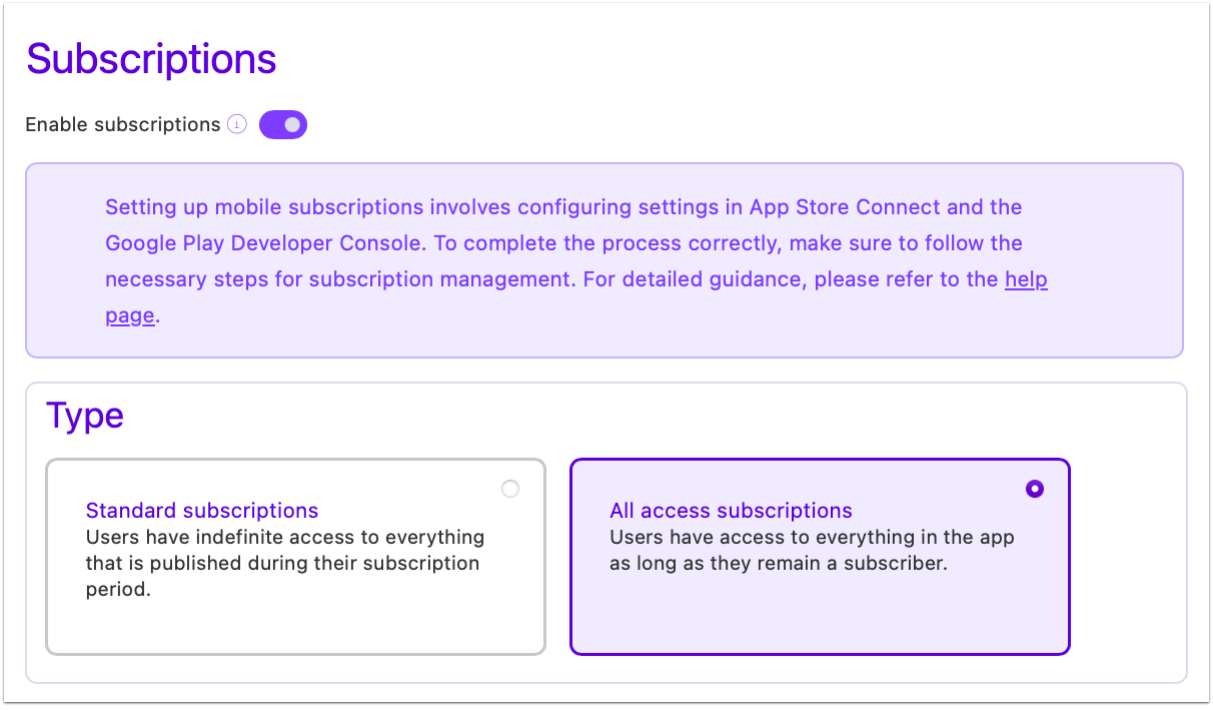
The subscriptions option in Twixl can be set via:

- Platform > Menu > Access > Subscriptions > Enable Subscriptions

1. Standard or all access subscriptions

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

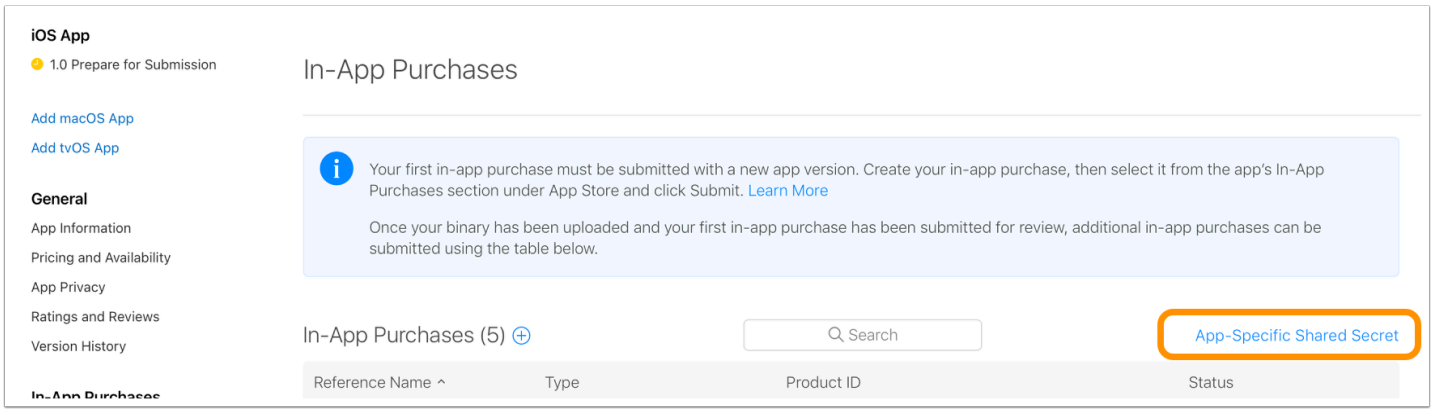
- **Standard Subscription**: this is the traditional type of subscription where you get access to new collections or PDF content items (i.e. issues) that are added during the period of your subscription.
- **All Access Subscription**: this can be compared to e.g. Spotify or Apple Music type of subscriptions: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



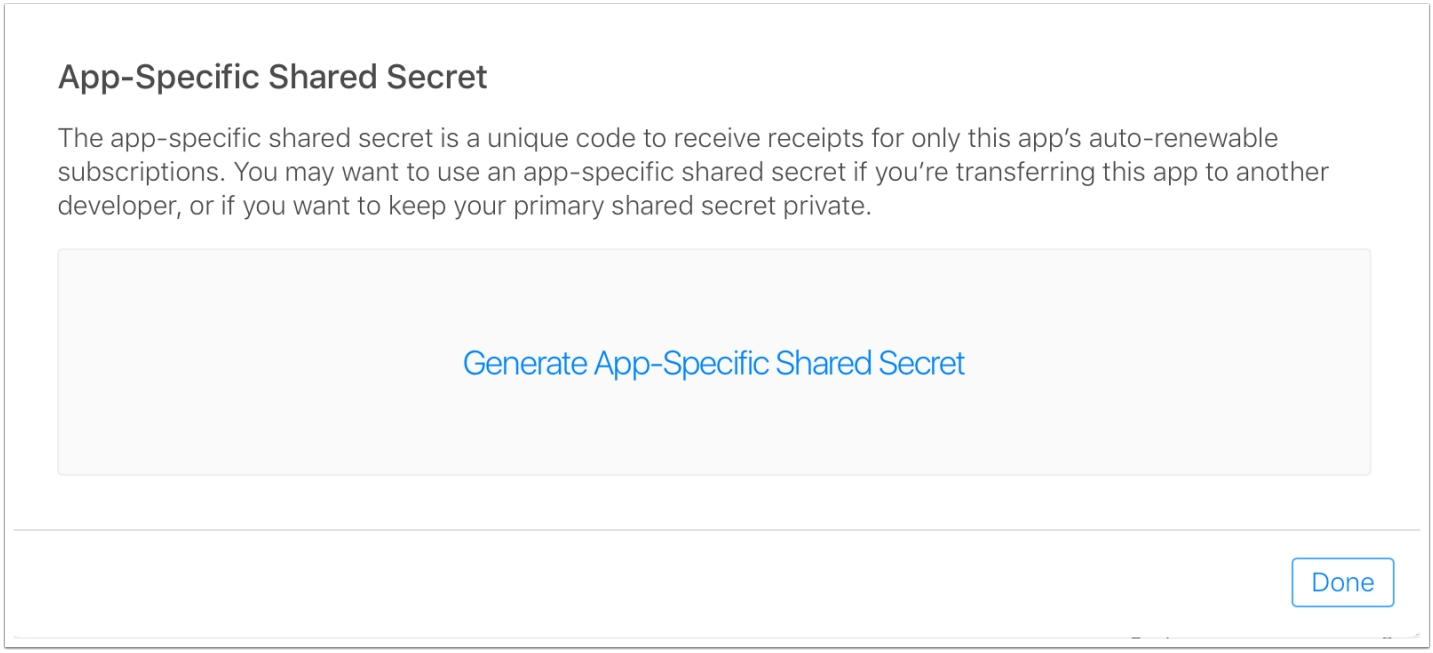
2. Subscriptions on iOS

To be able to offer subscriptions for iOS users of your app, you first need to generate an app-specific "Shared Secret". For Android users, this procedure is a lot easier (see 4.2).

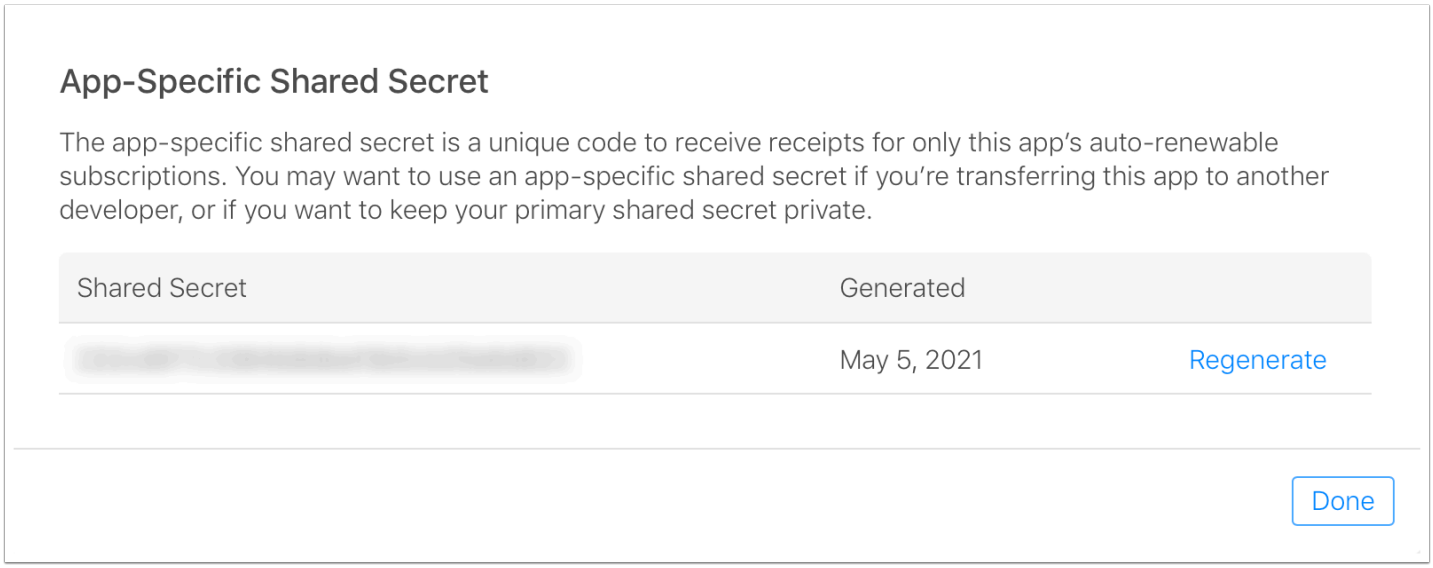
1. Go to https://appstoreconnect.apple.com -> My Apps -> Select your app -> Manage In App Purchases -> Select '**App-Specific Shared Secret**'.



2. Click '**Generate App-Specific Shared Secret**'.



3. Copy the Shared Secret.

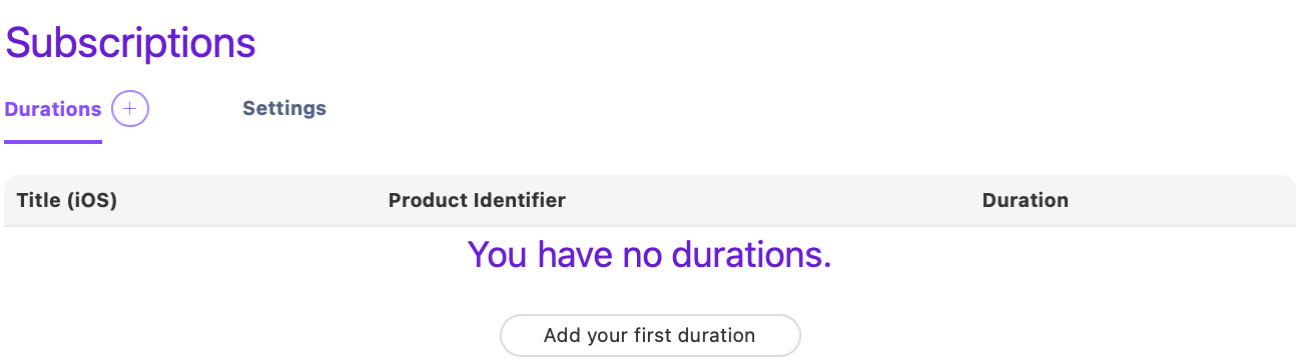


4. On the Twixl platform > Menu > Access > Subscriptions > iOS Specific > Paste the Shared Secret

If you're only creating an Android build, you can leave this field empty. When saving the page, a pop-up will ask if you don't want to add an App-Specific Shared Secret for iOS Subscriptions. You can confirm at this point.

Once you have saved the setup, two tabs will appear:

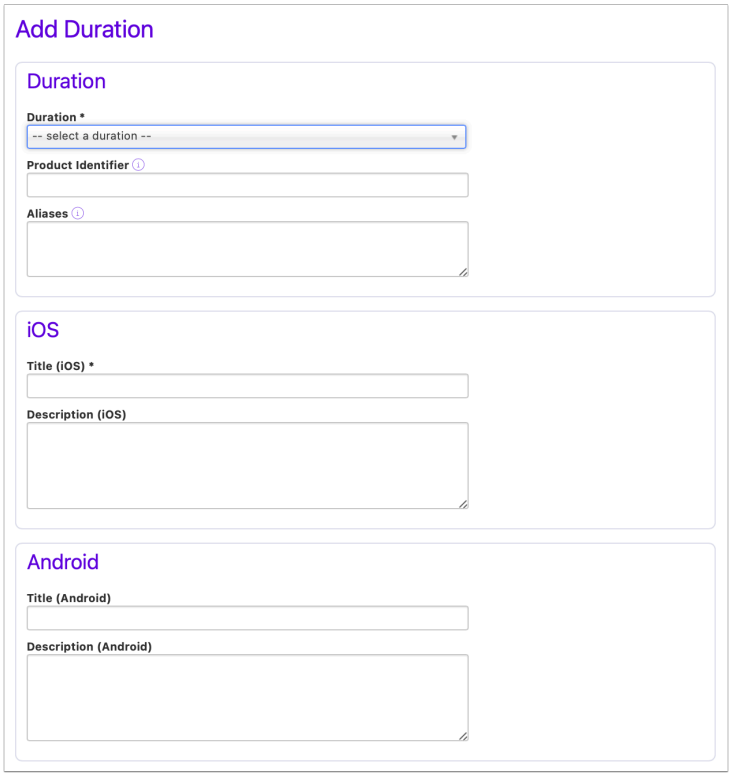
- Durations: Enter one or multiple intervals of subscriptions here.
- Settings: If you want to go back to your initial setup.



3. Adding a duration

To create a duration for an auto-renewable subscription, click the plus icon or "Add your first duration".

- **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`
- Aliases: Add aliases of product identifiers that will be treated the same. These aliases must be comma-separated.
- **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.



4. Adding a subscription

4.1. 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`

4.2. 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`

When you have opted for **Standard Subscriptions**, new subscribers will automatically receive the most recent paid collection or PDF that is published in your app.

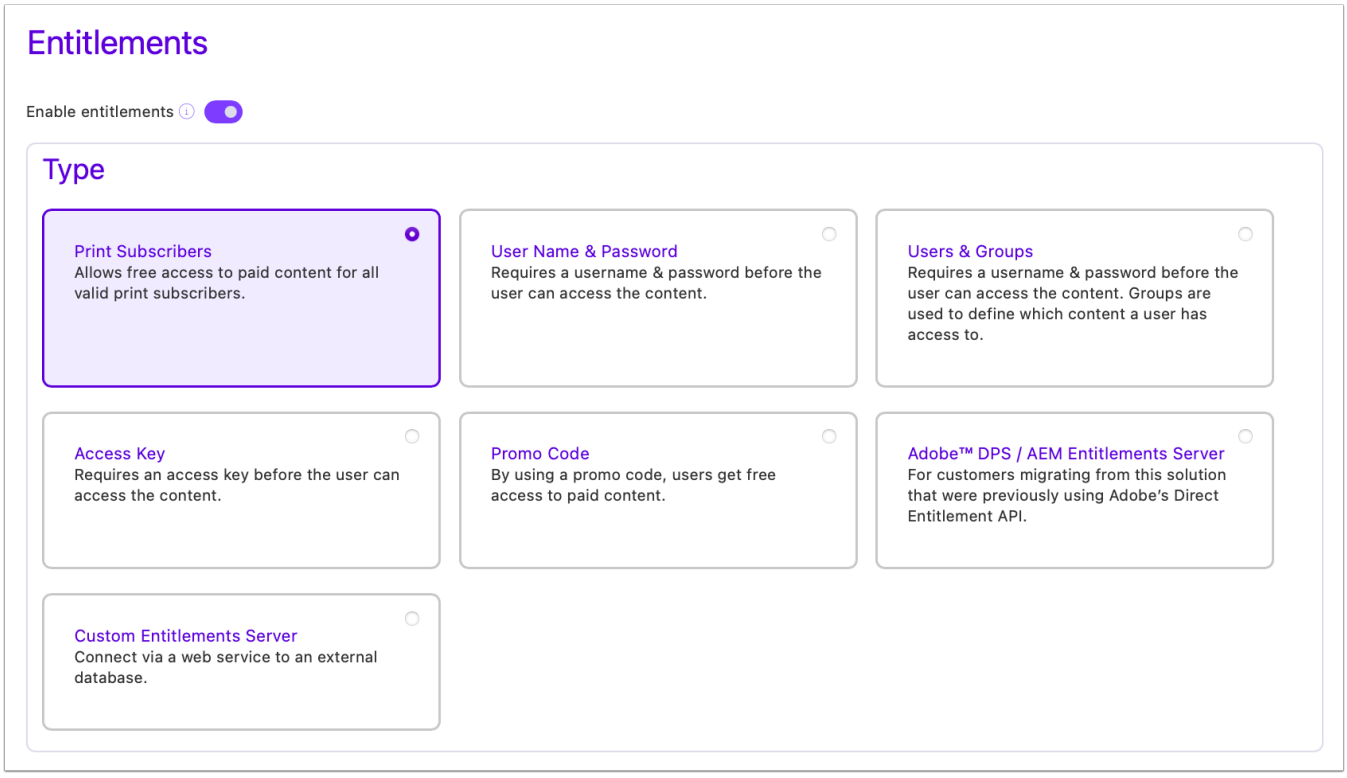
Entitlements

Using the Entitlements option

By using entitlement in your app, you can determine which content a reader/user can have access to. In the world of magazines, this is most frequently used to allow print subscribers to get free access to the digital editions, by logging in with a user name and a password. But entitlement is also frequently used by companies that want employees or partners to login to an app that can be distributed both internally or via the App Stores.

Via Platform > Menu > Access > Entitlements > Enable Entitlements

1. Configuring entitlements for your app



When navigating to the detail of your app, select Entitlements from the menu. You can then select one of the preconfigured scenario's:

1. **Print Subscribers**: for providing print subscribers free access to digital content - users can still see and purchase the different collections (issues) and also purchase a subscription through the App Store
2. **User name & password**: for a restricted access app, i.e. without a user name and password, users will not be able to access any content). This scenario also has a

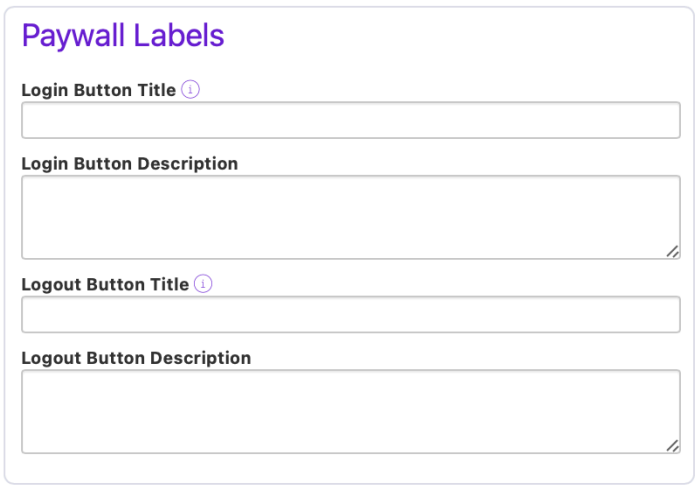
special default user called 'No Entitlements' that allows you to determine whether certain types of content will be displayed anyway when a user is not logged in.
3. **Users & Groups**: define access to your collections based on group access privileges - users belong to a group, collections can be visible in one or more groups.
4. **Access Key**: can be used for providing extra content to anyone with a valid access key
5. **Promo Code**: allows readers to redeem a promo code and get an issue (collection) for free that normally needs to be purchased
6. **Adobe DPS/AEM Direct Entitlement Server**: useful for customers migrating from this solution that were previously using Adobe's Direct Entitlement API.
7. **Custom Entitlements Server:** connect via a web service to an external database. See this article.

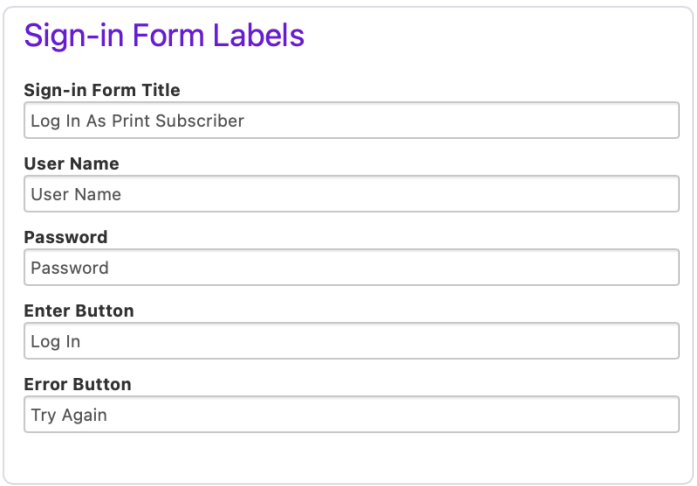
Once you selected a specific scenario, you will also be able to configure different fields depending on your initial selection.

Auto logout: If you enable this option, the app will require a re-login every time the user quits the app. A re-login will also be required when your mobile device goes into sleep mode. This is useful for those apps that require additional safety (e.g. banking apps, internal apps...).

Labels: You can define what texts will appear in:

- **Paywall:** Next to the purchase options, the user will also be able to login or logout. You can define the texts and information the user will see here.
- **Sign-in form:** Depending on the entitlement selection, this section defines the texts on the form where the user will login, logout, create an account, demand a new password, enter a promo code...



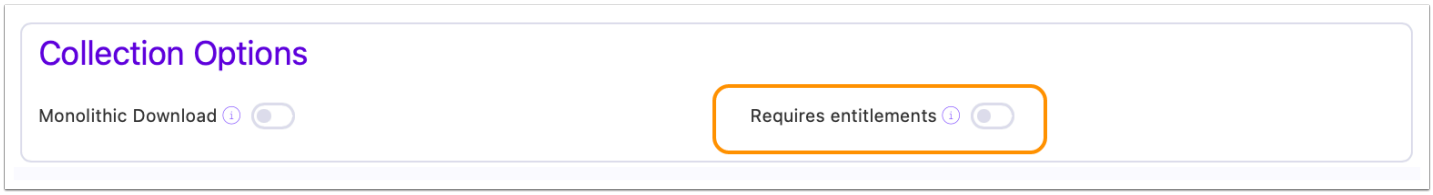


2. Collection Options

Once you have set the entitlement scenario in Access > Entitlements, you can enable this feature in a collection so this collection can only be entered by an entitled user.

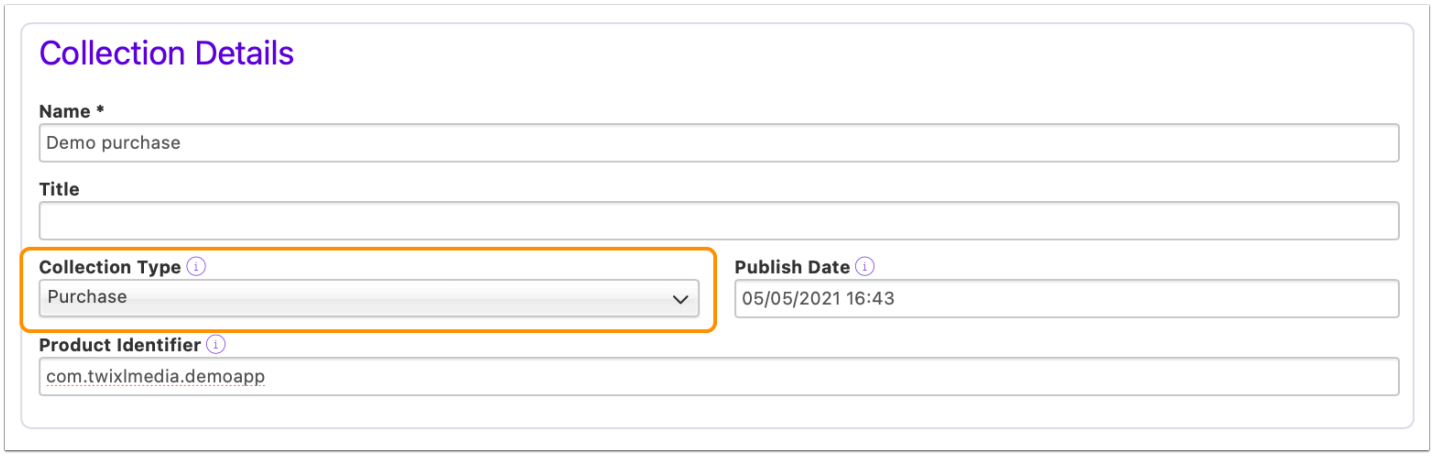
Enable entitlements via:

Collection > Collection Setting > Collection Options > Requires entitlements



2.1. Free of charge or in-app purchase

Each collection is set by default as *Free of charge*. You can also make it an in-app purchase. In that case, you need to make sure that this in-app purchase is also defined on your Apple and/or Google developer account using the same product identifier.



When a reader navigates to a collection that has been defined as an in-app purchase, it will trigger the paywall to be displayed. The paywall can display a number of different options:

- the standalone purchase of a collection (or '*issue*')
- the purchase of a subscription (if you defined any)
- a *Login* button: tapping this will select the entitlement sign-in form to be displayed
- You can easily define the text that will be shown in the Purchase Info.

2.2. Requires entitlement

If this setting is enabled for a collection, the entitlements sign-in form will be triggered when an unentitled user wants to navigate to that collection. If a reader then logs in with a valid user name and password, he will get access to the collection.

IMPORTANT NOTE:

If you check *Requires Entitlement* for the root collection, then the login form will be displayed on startup, and anyone who's not entitled will not be able to access any content in the app.

3. PDF Content Items

PDF Content Items are something special: while they are indeed a type of *Content Item,* for the purpose of *Entitlements* they are treated as a *Collection*. **As such, you can set the Entitlements options for a PDF Content Item**.

SOME TIPS:

- PDF Content Items have a **Product Identifier** (used for Entitlements and In-App Purchases). All other Content Items don't have Product Identifiers.
- PDF Content Items can - as a result - be offered as a **purchase**.

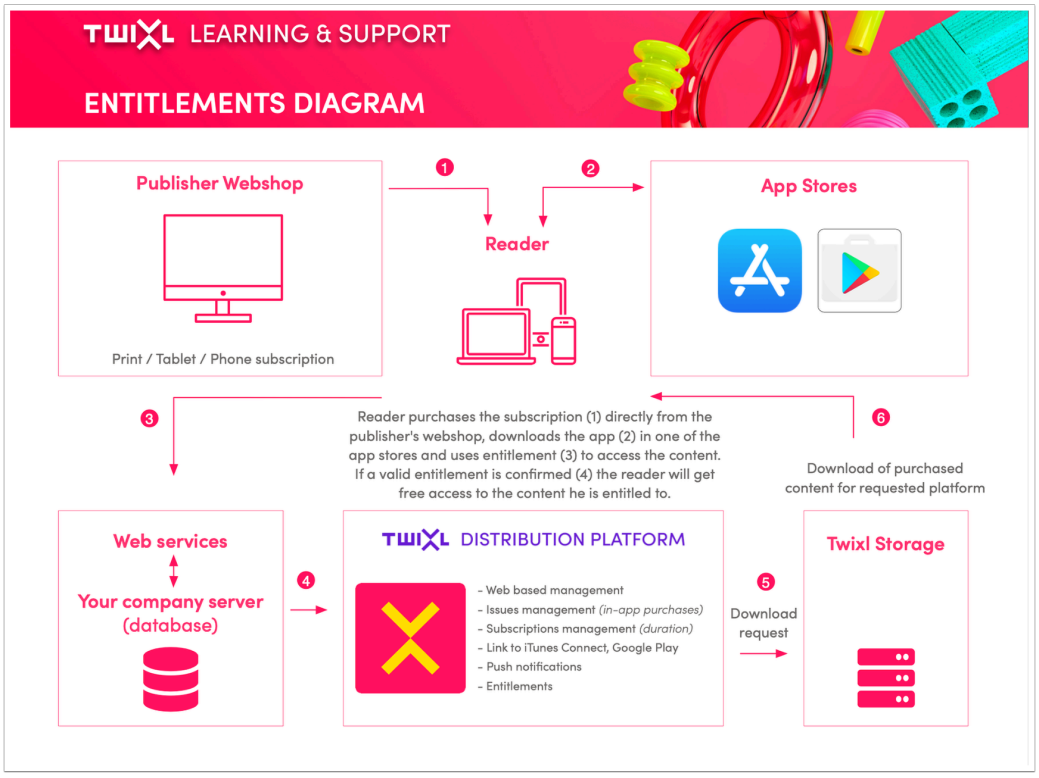
- PDF content Items are also checked against the **Entitlements** to see if they need to be offered for free or not.
- PDF Content Items are not shown in **Detail Mode** of a Collection. They are being presented in their own special Detail View.

4. Related articles

- Integrating a Custom Entitlement Server

Entitlement Diagram: how it works

This diagram explains in a nutshell how Entitlements work.



You can also download this diagram:

Twixl Entitlements Diagram

Integrating a Custom Entitlements Server

For some customers, the built-in entitlement scenario's don't always offer exactly what they are looking for... others require a connection to an external data source. In those cases, a custom entitlement server can be an alternative.

The implementation and deployment of a custom entitlement server is something that can be offered by Twixl or by Twixl's solution partners, and is based on the technical information below.

Via Platform > Menu > Access > Entitlements > Enable Entitlements > Custom Entitlement Server

If you are looking for the Twixl dev team to implement a custom entitlement server, then please provide us a much details as possible about the project via this form.

To use a Custom Entitlements Server in your app, you will need an external web application that implements the Entitlements API.

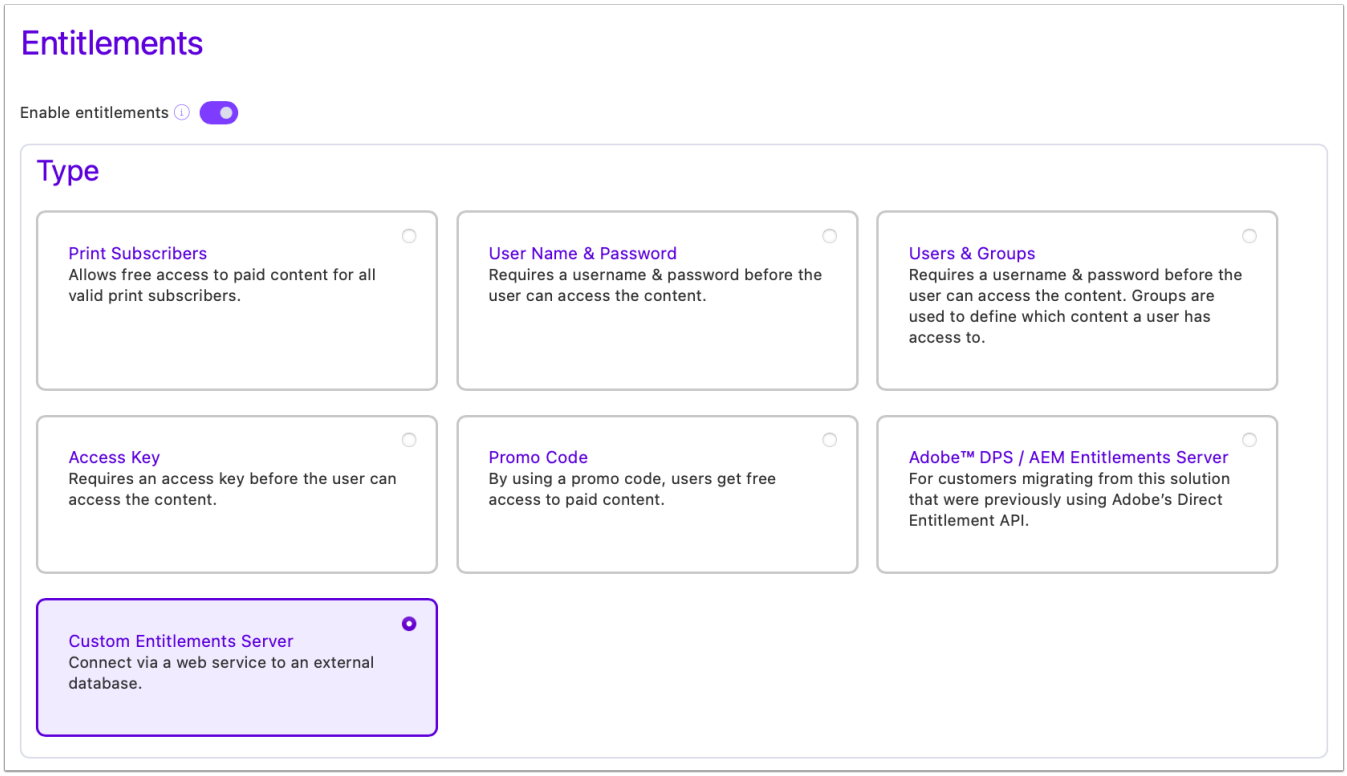
1. Terminology

There is some specific terminology you need to be familiar with before using entitlements:

- **Entitlements Server**: the external web application that implementes the Entitlements API.
- **Token**: a unique identifier for the user on the entitlements server. This can e.g. be a print subscriber number, a region, ... This token will be used to verify which content a user has access to.
- **Device UDID**: in a Twixl app, a device is identified by a unique identifier that stays the same, even when you reinstall the application on the same device. This can be used to track a specific device. Note that this information remains completely anonymous.

2. Configuration

Via Platform > Menu > Access > Entitlements > Enable Entitlements > Custom Entitlement Server



First of all, make sure you have done the necessary configuration on the Twixl platform, as described in the Entitlements documentation.

3. API

3.1. Sample implementations

You can download example implementations of an Entitlements Server SDK in the following server-side programming languages:

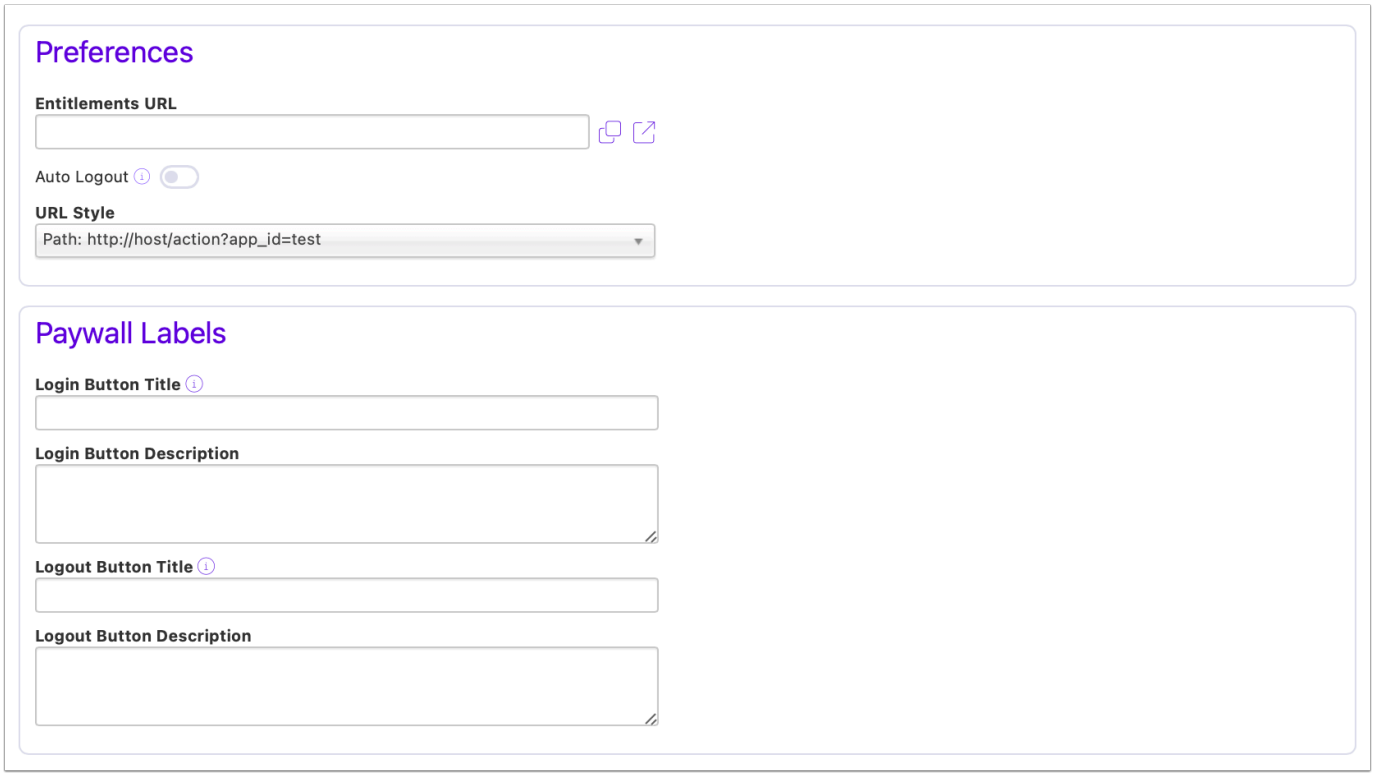
- ASP.NET
- Java
- PHP

3.2. Syntax and payload

More info about the syntax and the payload can be found in the API documentation on the Twixl platform.

3.3. Entitlements Server URLs

Depending on the server side implementation of the Entitlements Server, there are two different ways the URLS can be constructed. This can be configured on the Twixl platform.



3.4. URL Styles

The first style is passing the action in the query string parameter called “do”. This can be done by selecting the following url style in the build settings.

```
https://<url>?do=signin_form&amp;app_id=test</url>
```

Click to copy

This will generate URLs in the following style:

```
https://<url>?do=<action>&amp;param=value</action></url>
```

Click to copy

If you prefer the action to be a part of the url itself, you can choose the following url style in the build settings:

```
https://<url>/signin_form?app_id=test</url>
```

Click to copy

This will generate URLs in the following style.

```
https://<url>/<action>?param=value</action></url>
```

Click to copy

> **IMPORTANT NOTE:**
>
> For security reasons it is obviously a requirement to use the https protocol for your connections. Note that this requires that you have a secure SSL certificate installed on your web server.

3.5. signin_form

- **HTTP method:** `GET / POST`
- **POST Parameters:** None
- **Description:** This API call shows the signin page. Using this page, you can e.g. ask users for their print subscribers credentials or allow them to choose e.g. a region from a list of regions. When you use a link to trigger the signin action, you can specify the link as follows:

```
<a href="?do=signin&amp;region=uk">Select UK</a>
```

Click to copy

When you use a form, there are a few things you need to take into account:

- If the signin_form is fetched by the browser client it should contain method POST, otherwise it should use GET. The reason for this is that the mobile apps need to 'hijack' the signin call (and they will convert it to POST) but the browser client needs to send the credentials by POST to make it secure.
- The browser client will fetch the signin_form with a ref=browserclient parameter. Check for this parameter to return a form with a POST method.
- The form should call the action `signin` (using a hidden form field).

The example HTML code for a form can be:

```
<form method="GET">
    <input type="hidden" name="do" value="signin">
    <input type="text" name="email">
    <input type="password" name="password">
</form>
```

Click to copy

When the user clicks the link or submit the form, the application will intercept the URL and parse the query string. It will then add the following parameters to the dictionary of keys and values:

- **`app_id`**: the unique application identifier of the app (e.g. `com.twixlmedia.myfancyapp`)
- **`app_version`**: the version number of the application (e.g. `1.0`)
- **`udid`**: the unique Twixl identifier of the device (e.g. `3efad737b4d845ffa6ddc4d484b279e9`).

3.6. signin

- **HTTP method:** POST
- **POST Parameters:**
  - **`app_id`**: the unique application identifier of the app (e.g. `com.twixlmedia.myfancyapp`)
  - **`app_version`**: the version number of the application (e.g. `1.0`)
  - **`udid`**: the unique Twixl identifier of the device (e.g. `3efad737b4d845ffa6ddc4d514b279e9`).
  - all parameters from the signin_form action.
- **Description:**
  - This is the call that will check if the user is entitled or not. If the user is entitled, it should generate a so-called "entitlement token". This is a unique ID identifying the user and will be used by Twixl Publisher to identify the user for all other entitlement requests.
  - The signin call should always return JSON data. There are two results that are possible, depending on the result.
  - If the user is entitled, the following JSON structure should be returned:

```
{"token": "the entitlement token"}
```

Click to copy

If the entitlement fails, the following JSON structure should be returned:

```
{"error": "a message explaining why the user is not entitled"}
```

Click to copy

3.7. signin_succeeded

- **HTTP method:** `POST`
- **POST Parameters:**
  - **`token`**: The token that was returned in the signin action.

- **Description:** This is the view that is shown when the signin returned a valid token (which means that the user is entitled). You can either show an HTML page with more details based on the token passed as a query string parameter. If you want to provide a link that allows the user to close the popup window, you need to use the following URL:

```
<a href="tp-close://self">Close Me</a>
```

Click to copy

If you want to automatically close the popup when the signin succeeds, you can also perform an HTTP redirect to the `tp-close` URL:

```
header('Location: tp-close://self')
```

Click to copy

3.8. signin_error

- **HTTP method:** `POST`
- **POST Parameters:**
  - **`error`**: The error that was returned in the signin action.
- **Description:** This action is executed when the signin action returns an error message. This can be used to e.g. indicate that the user is not known or not entitled. The actual error message will be passed as a query string parameter.

3.9. entitlements

- **HTTP method:** `POST`
- **POST Parameters:**
  - **`requested_identifier`**: the product identifier of the requested issue (will only be filled in when the user does a purchase. It will be empty when the user requests the contents of the kiosk).
  - **`app_issues`**: the list of issues defined for the app encoded as a json string
  - **`app_id`**: the unique application identifier of the app (e.g. `com.twixlmedia.AvantGand`)
  - **`app_version`**: the version number of the application (e.g. `1.0`)
  - **`token`**: the token that was returned in the signin action
  - **`product_identifiers`**: a json string containing the list of product identifiers the application is about to show.
- **Description:** This action is called every time the application wants to show the list of issues or when a user tries to purchase an issue. The post request will contain a parameter called “product_identifiers” which is a JSON string containing the list of product identifiers the application is about to show. To convert these to a real list, you

need to parse the JSON string. In PHP, this can be done using the json_decode function:

```
$product_identifiers = json_decode($_POST['product_identifiers']);
```

Click to copy

The result of the entitlements call should be a JSON structure with the following content:

```
{
    "token":"my-entitlement-token",
    "entitled_products":
        [
            "com.myapp.product1",
            "com.myapp.product2"
        ],
    "mode":"hide_unentitled"
}
```

Click to copy

The parameter `entitled_products` should contain the list of issue identifiers to which the user is entitled.

The parameter `mode` defines how the Twixl Distribution Platform will interpret the results:

- **`purchase_unentitled`**: make all the products to which you are entitled free if you didn't get them for free yet
- **`hide_unentitled`**: hides all the items to which you are not entitled from the app

In this call, you should also make sure you properly handle the case where the "token" is empty. This basically means that the user is not entitled.

3.10. force re-login of the user

When the /entitlements call returns a token __token_expired__, it will force the app to clear the token, thus requiring the user to login again.

Note that this feature requires the Twixl app to be built with **version 15+**.

Custom Entitlements Servers Inquiry Form

For some of our customers, the built-in entitlement scenario's don't always offer exactly what they are looking for... others require a connection to an external data source. In those cases, a custom entitlement server may offer an alternative. The implementation and deployment of a custom entitlement server is something that can be offered either directly by Twixl or by Twixl's solution partners.

If you are looking to implement a custom entitlement server, then please provide us a much details as possible about the project via the form below.

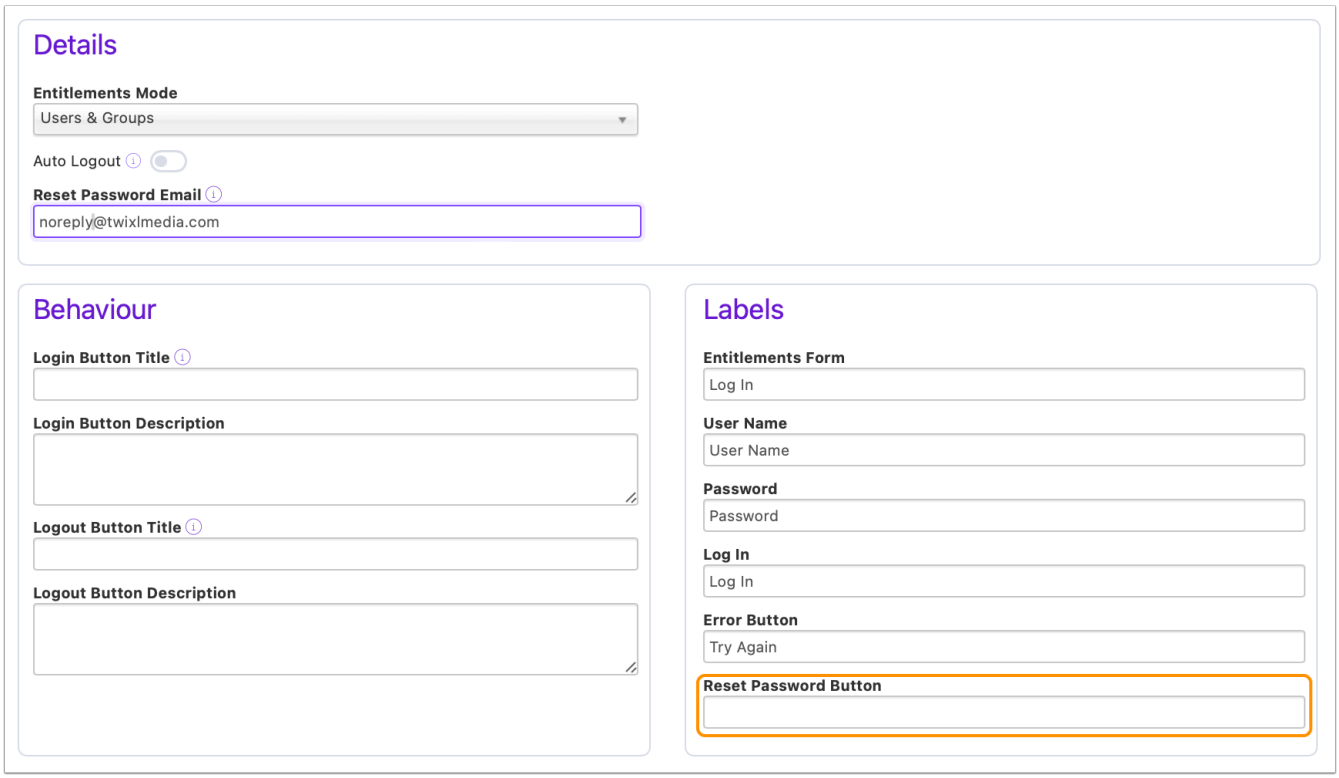
Yes, I'm interested in the Custom Development of a **Custom Entitlements Server**

Using "reset password" for the "Users & Groups" scenario

Please note that the "password reset" feature can be used only with the entitlement scenario "Users & Groups". This feature is not enabled by default, so If you want to use this functionality in your app, please create a support ticket.

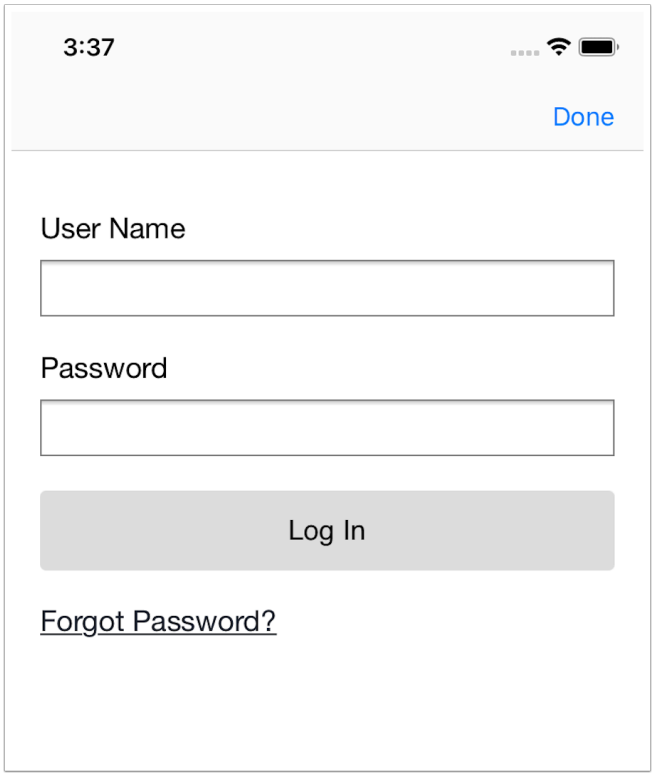
1. How to configure

When you select the Entitlement scenario "Users & Groups", under "Labels", you'll see an field appear that can be used to define the text that is shown for the "Forgot password" link. If you leave it empty, the link will simply not show up.

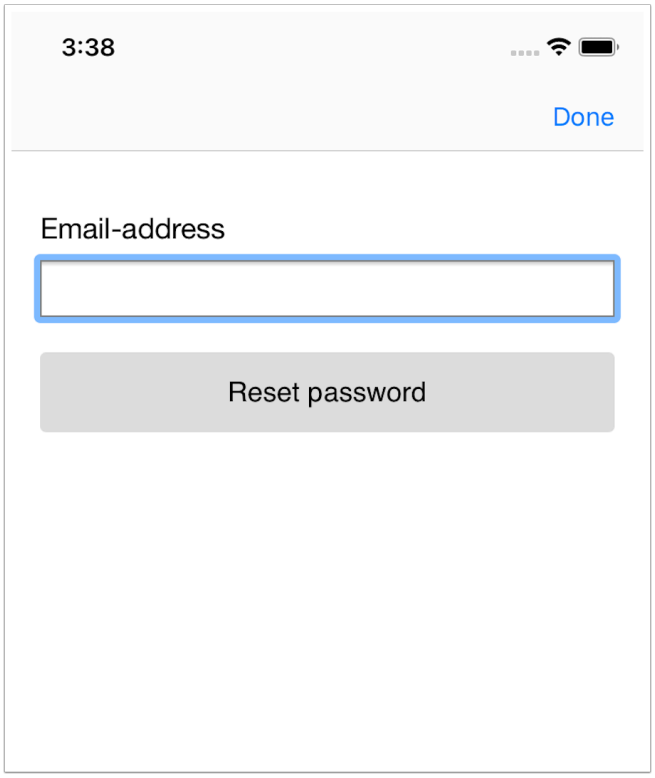


2. Reset flow

When a user is in the app and the entitlement login dialog is shown, you'll see a link that allows you to reset your password:



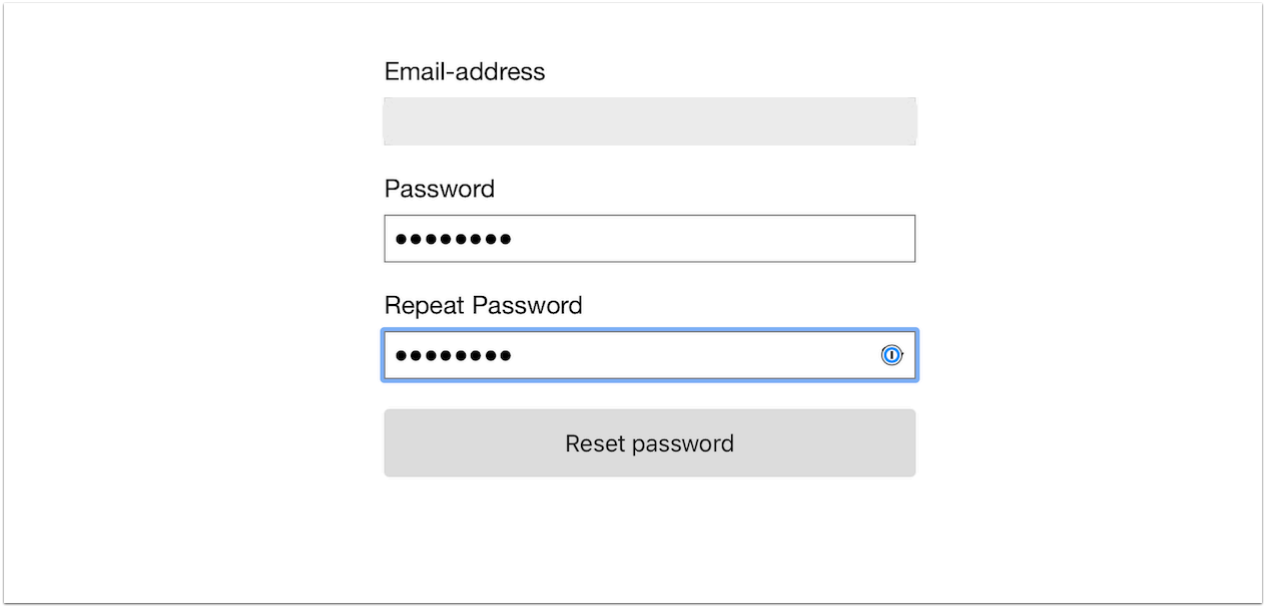
Tapping "Forgot Password?" takes you to a page where you can enter your email (or whatever you used as the label for the "username" on the platform):



You can then fill in the username / email and tap "Reset password". This will trigger that an email is sent with a link to the form that allows you to reset the password.

The "from" address is the address that has been configured as the support email in the app settings. The title in the mail is the name of the app as configured on the platform.

Tapping the 'Reset your password' link takes you to the following form which allows you to reset the password:



After resetting the password, a message explains you that you can now login to the app with this new password.

In-app purchases

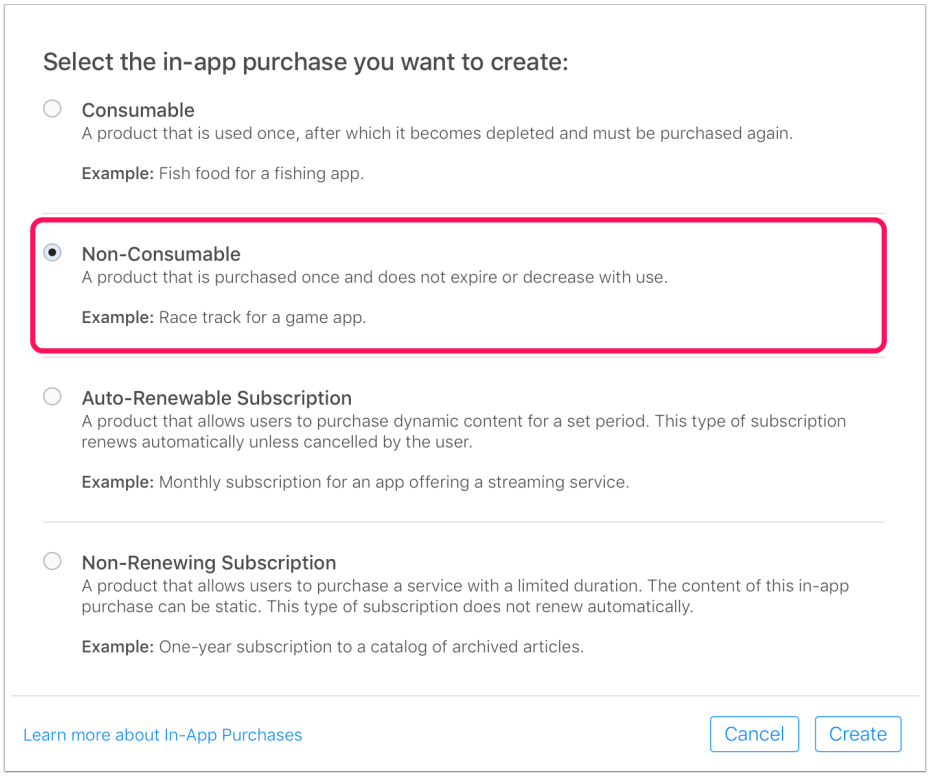
In-app purchases for iOS apps

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

1. Adding in-app purchases in AppStore Connect

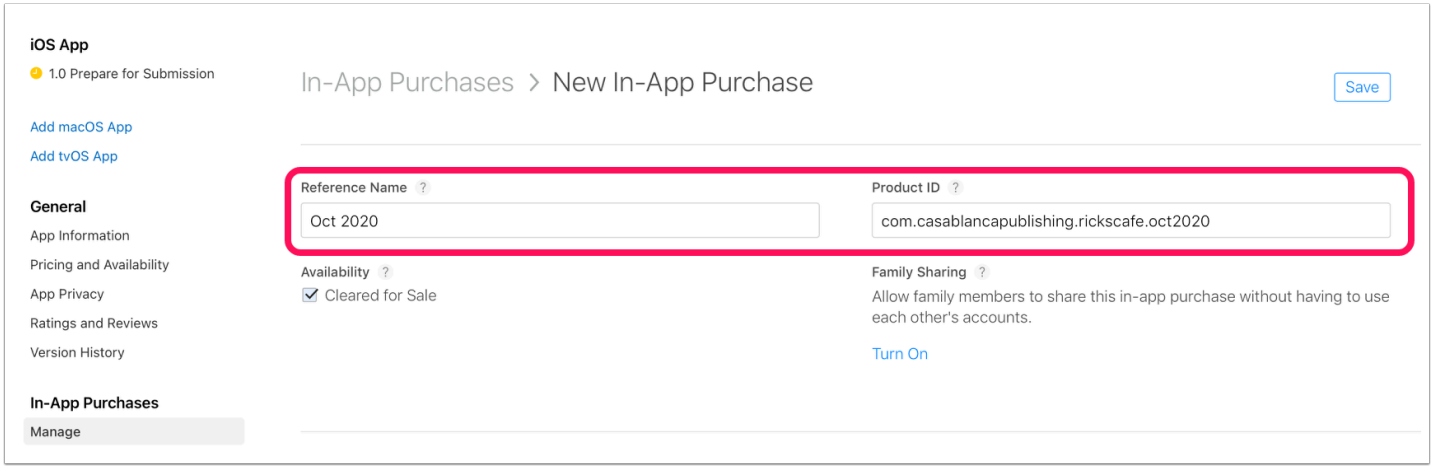
In your app, contain may be offered for free, or some or all content may be available as an in-app purchase.

- 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

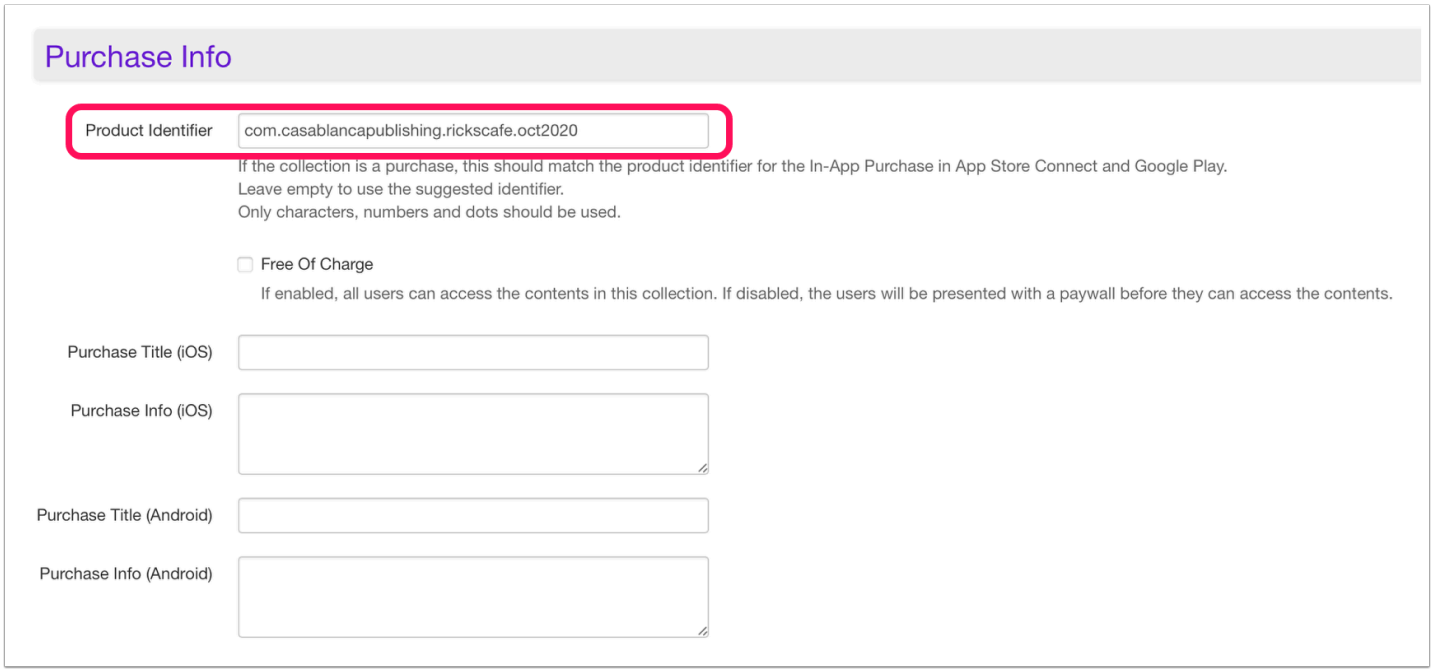
4. Provide in-app purchase information for at least one language.



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

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

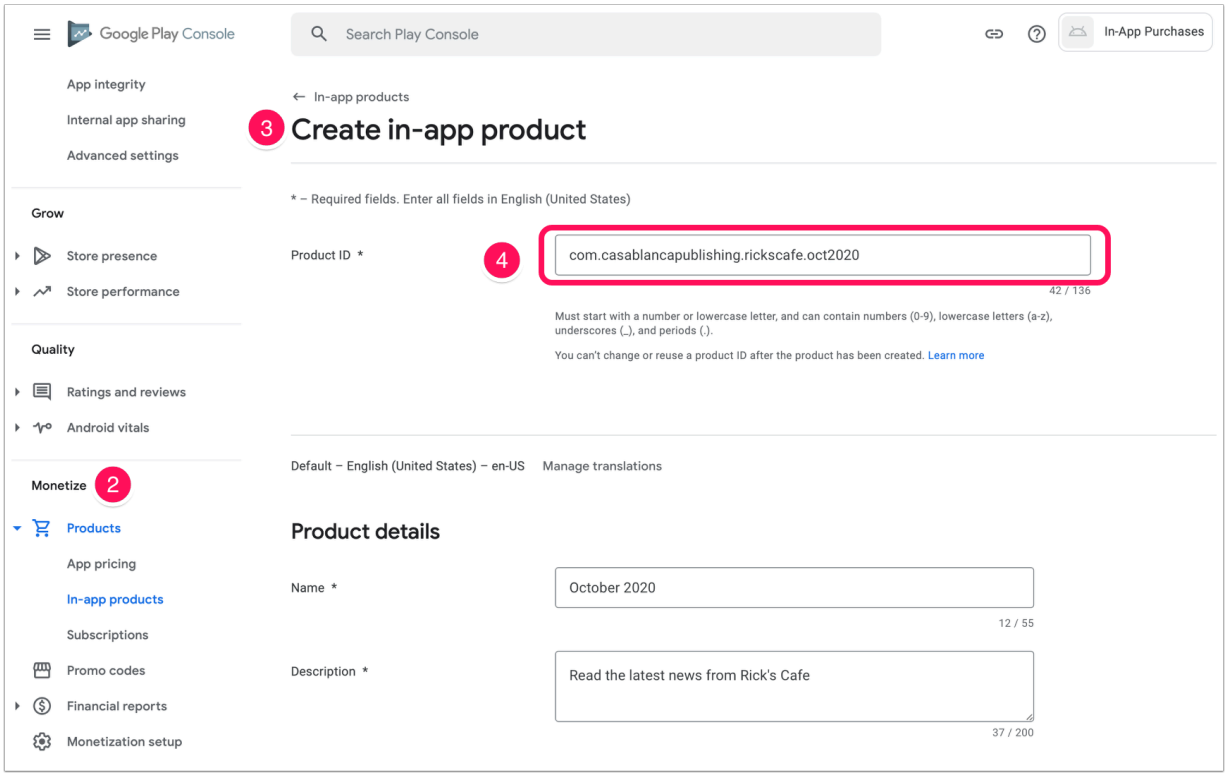
In your app, contain may be offered for free, or some or all content may be available as an in-app purchase.

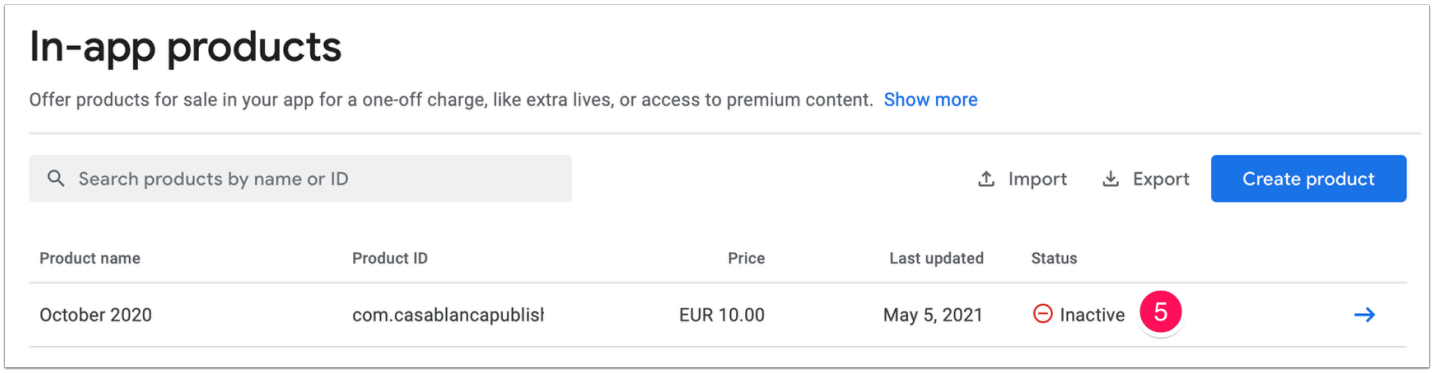
If your app offers only free content, there's nothing else to configure but adding your content to the Twixl platform.

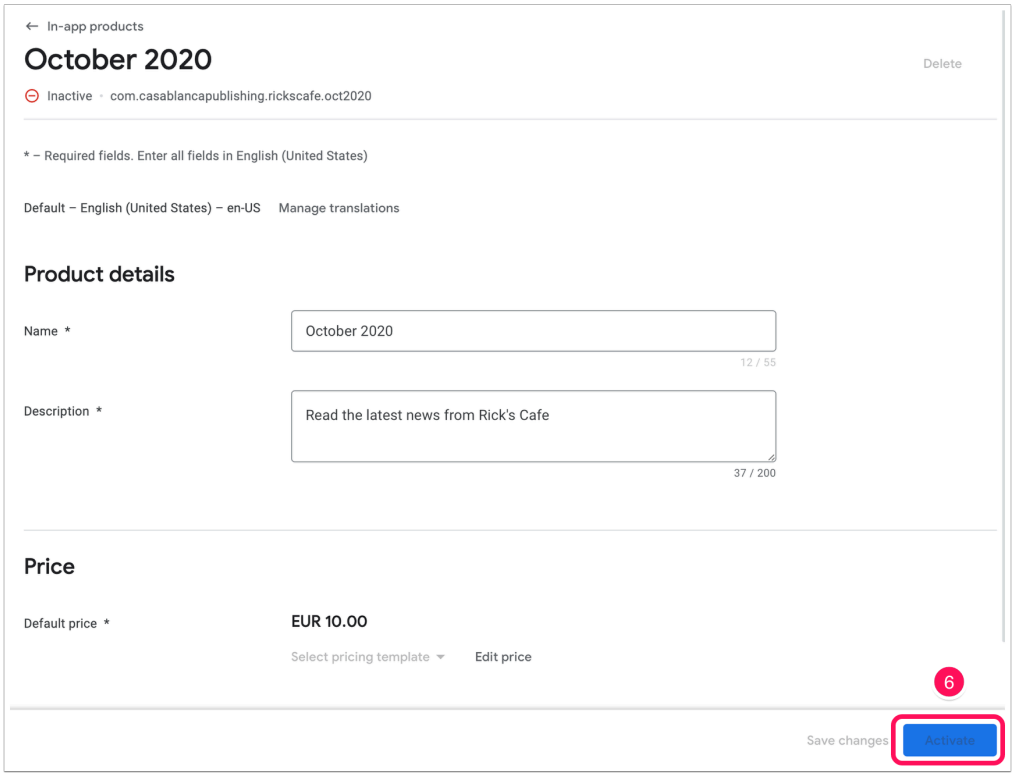
If you want to make collections or PDF content items available as an in-app purchase, you will need to create a separate in-app purchase entry for each of the collections you want to offer:

- **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

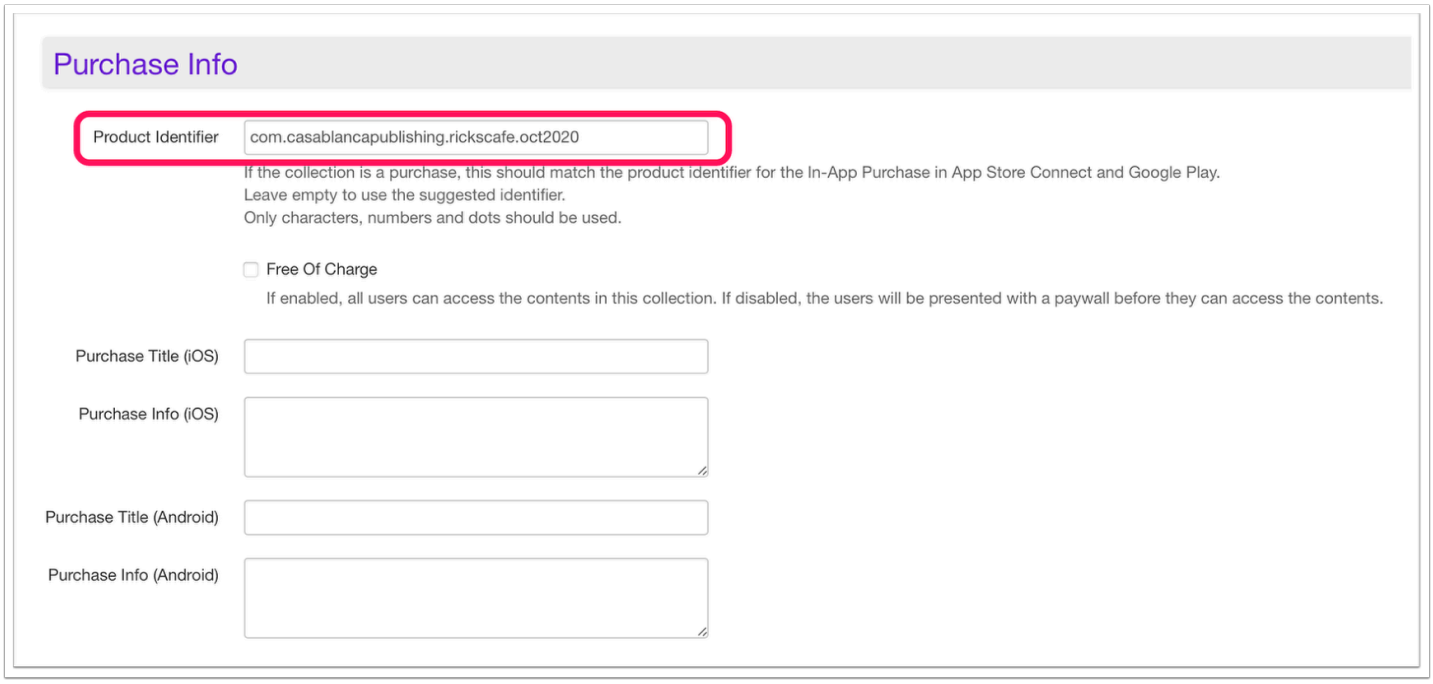

In-app products
Offer products for sale in your app for a one-off charge, like extra lives, or access to premium content. Show more
Search products by name or ID
Import
Export
Create product
Product name
Product ID
Price
Last updated
Status
October 2020
com.casablancapublisl
EUR 10.00
May 5, 2021
Inactive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

Free previews: Metered Access

If your app has purchasable content and you want to offer a free preview of that content to the app user, you can use the metered access feature to make your life easier. Note that this feature requires TP19 or higher.

Principle

The metered access feature makes it easier to create links between preview and paid collections. The app will check whether the user has access to either the preview or the purchased collection. As a publisher you have complete control over the content that will be shown to the user because the linking is performed between collections. These collections can contain most types of Twixl content items*.

Workflow

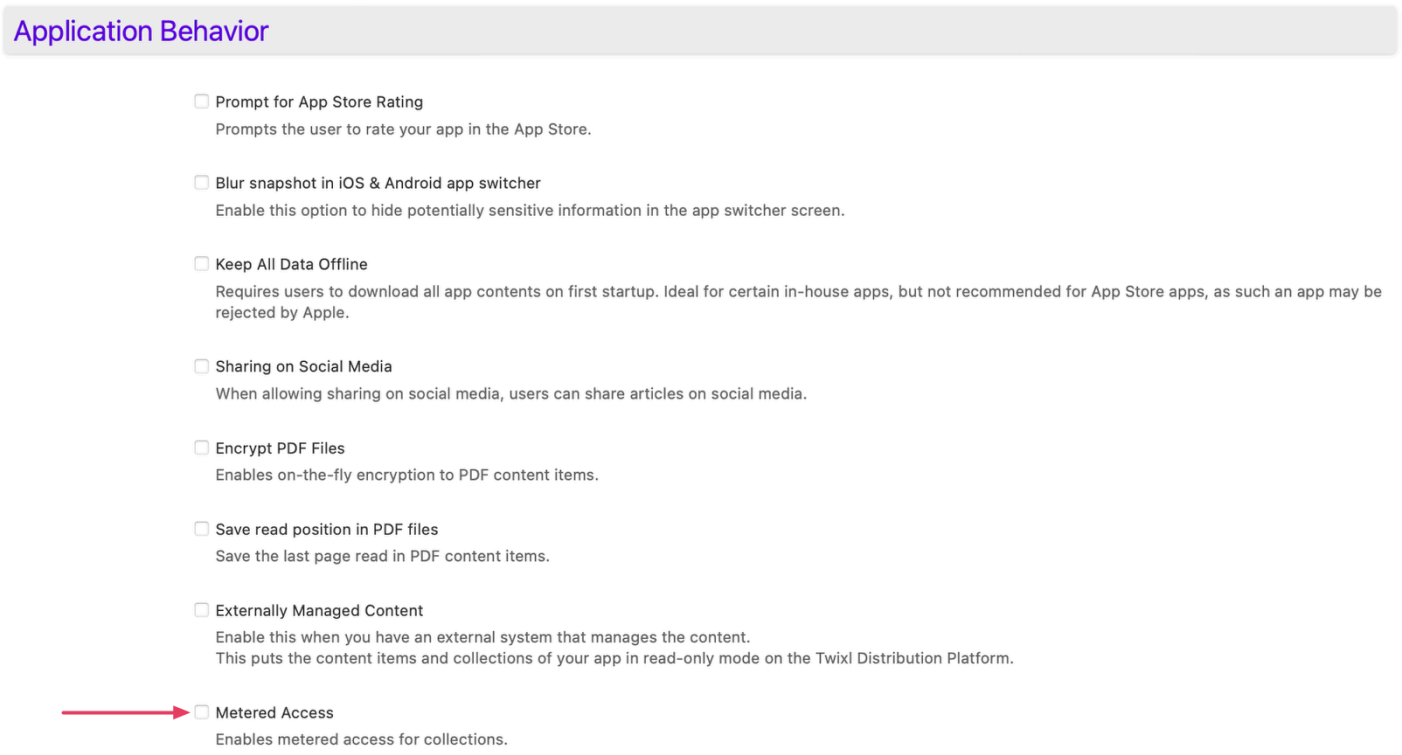
1. Enabling the metered access feature (app setting)
2. Defining collections as purchase or preview
3. Linking collections to a preview or purchase collection
4. Adding content to the collections
5. Adding purchase button

1.Enabling the metered access feature

To be able to use the metered access feature, you have to enable this function in the app settings.

How:

Go to: Menu > Design > Configuration > Application Behavior > check the 'Metered Access' box > Save

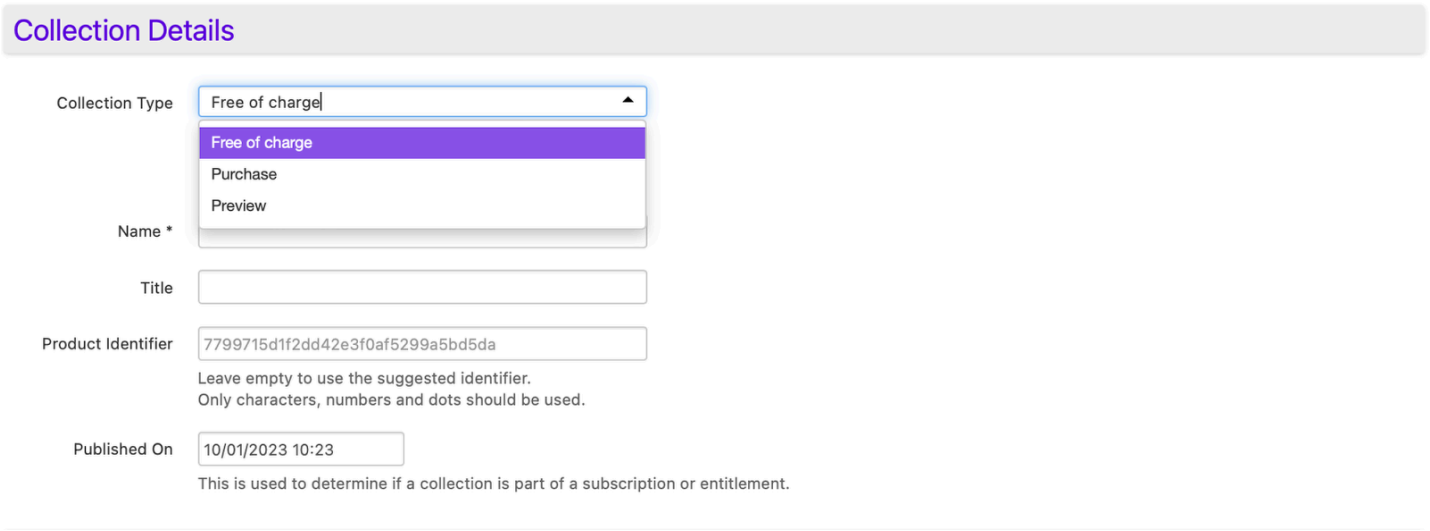


⚠ Note that enabling metered access cannot be reversed. So always make sure to test this in a development app first!

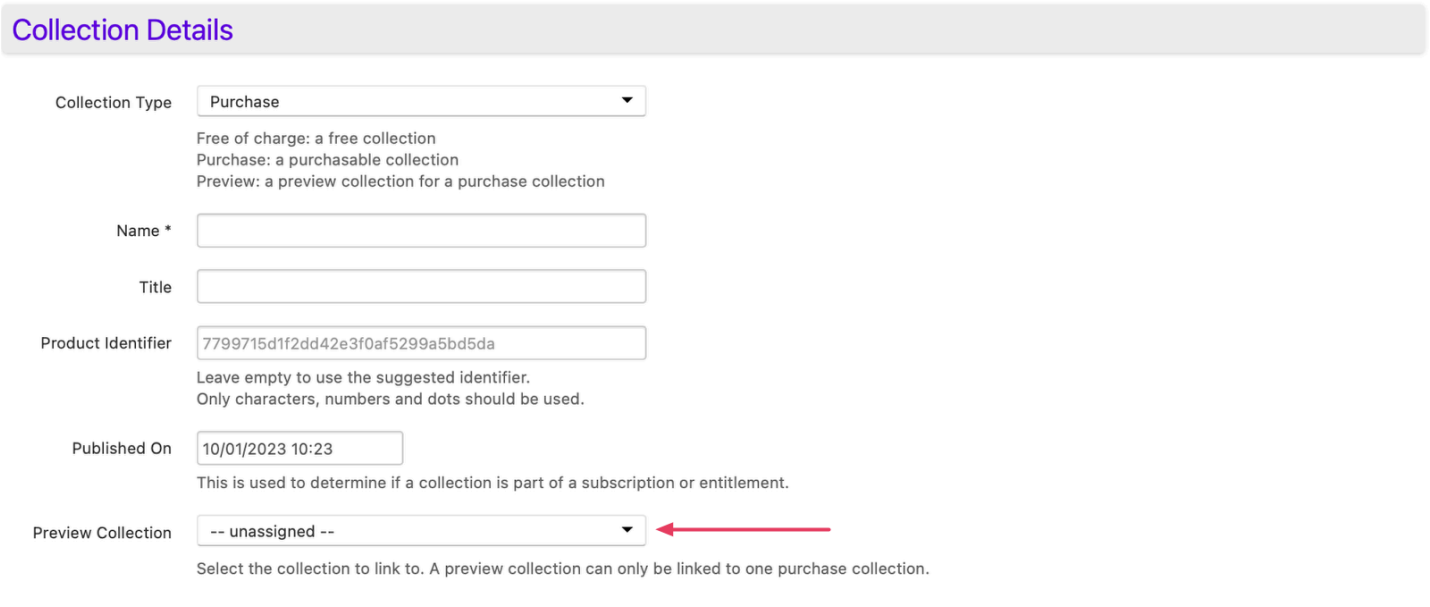
2. Defining collections as purchase or preview

Once the metered access feature has been enabled, an extra 'preview' option appears when creating or editing a collection. When creating or editing a collection you have 3 options:

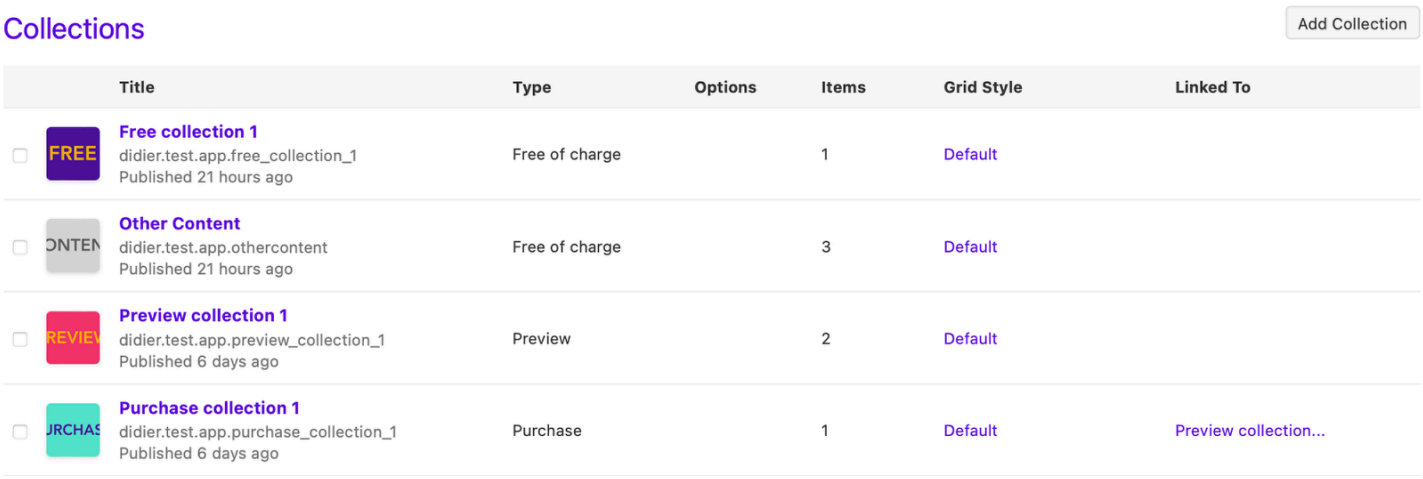
- Free of charge
- Purchase
- Preview



When a collection is marked as Preview, you will be able to link it to an existing Purchase Collection in Collection Details while creating the collection. A collection marked as Purchase will offer the possibility to link to an existing Preview collection via the dropdown. Linking the collection is not mandatory. You can link to a collection afterwards too. You can complete all collection fields as usual and save it.



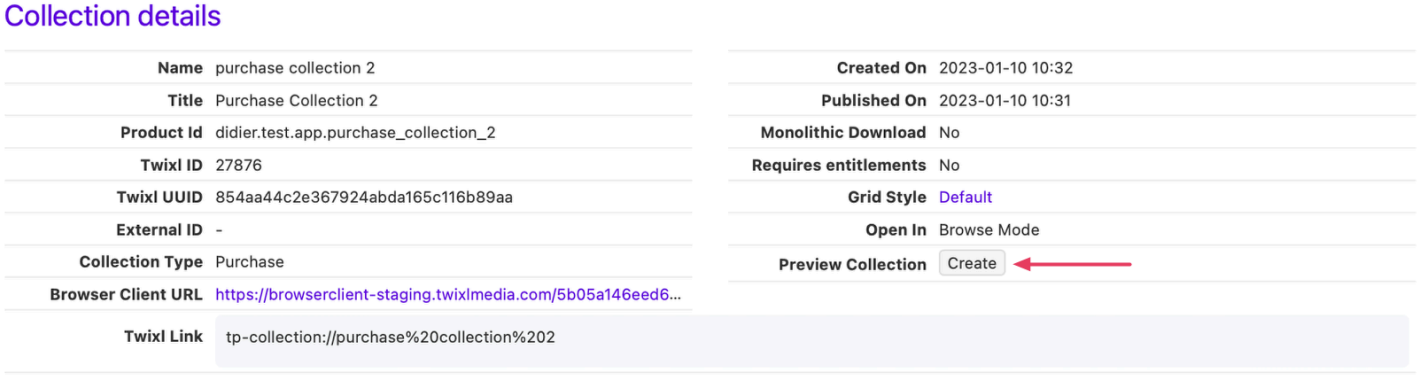
Whether a collection is free, preview or purchase is shown in the Type column in collection overview in Home or via Main > Collections. If there is already a collection linked to this collection, will also be shown in the 'Linked To' column in this overview.



3. Linking collections to a preview or purchase collection

Some possibilities to create links between collections:

- Via edit in a collection and select the matching collection
- By creating a new collection and selecting the matching collection
- By clicking on 'Create' in the Collection Details of a collection and creating a new collection



4. Adding content to the collections

In Preview collections you can add your content as usual. For preview collections, the type of content items that are allowed are:

- HTML Article
- Twixl Article
- PDF
- Image
- Movie
- Vimeo Movie
- YouTube Movie

- Inline Web Viewer
- Embedded Web Viewer
- Placeholder
- Weblink

This means you are free to provide preview content that maximises purchase attraction.

5. Adding purchase button

How you integrate and design the purchase button is also completely customisable. If you create a Twixl article (InDesign), you can integrate the "tp-paywall://" scheme via the Twixl Plugin in your design. This will call the paywall function and once the content has been purchased, the app will only show the purchase collection. After the purchase of a collection has been completed, its preview collection will no longer appear.

Other content types will require to add a web link containing a "tp-paywall://" scheme.

Click here for more information on these custom url schemes.

Final check:

Before the go live of your publication, make sure both your content items both in the purchase and the preview collection are in the published status.