Creatio Academy

App branding and publishing

Brand and publish mobile apps built on Mobile Creatio

Version 8.0

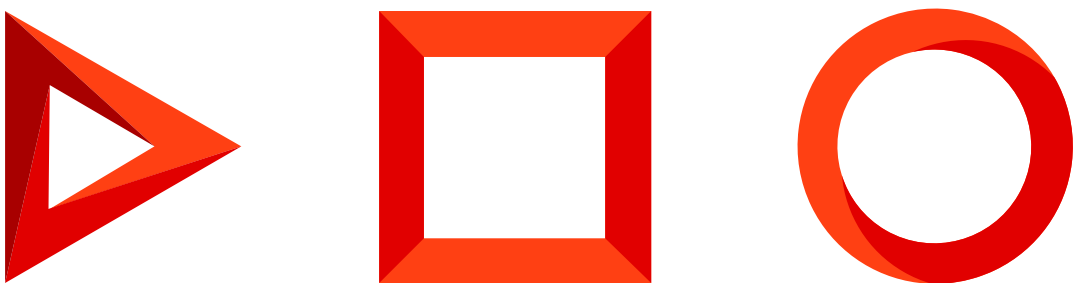


Table of Contents

Brand and publish mobile apps built on Mobile Creatio

Easy

You can use your logos and names to brand a mobile app built on Mobile Creatio using the **SDKConsole utility**.

Set up the SDKConsole utility

In general, the procedure comprises the following **steps**:

1. Perform the preliminary setup.
2. Install and set up the SDKConsole utility.
3. Run the SDKConsole utility.

1. Perform the preliminary setup

1. **Ensure you can publish the app**. You must be enrolled into Apple Developer program to publish your app on iOS and have a Google Play developer account to publish the app on Android. Learn more on Apple and Android websites: Apple Developer Program, Google Play Console.
2. **Enable Firebase Cloud Messaging to send push notifications**.
   a. Sign in to https://firebase.google.com/.
   b. Click [*Go to console*] in the top right.
   c. Create a project in the console.
   d. Add your Android and/or iOS app to the project.
   e. Download the config files for the app and save them for later. Specify the path to these files in the utility settings using the `google_service_info_file` property.
   f. Retrieve the server API key. To do this, click ⚙ in the top left of the Firebase project dashboard → [*Project Settings*] → [*Cloud Messaging*] tab → [*Project credentials*] → [*Server key*].
   g. Save the server API key to the `[PushNotificationService]` table of the Creatio database. For example, you can do it using the following SQL query.

      SQL query

      ```
      UPDATE PushNotificationService SET Settings = '{"url":"https://fcm.googleapis.com/fcm/","ap
      ```

      `Some_Api_Key` is your API key.
3. Download **and install Java development kit version 8**.

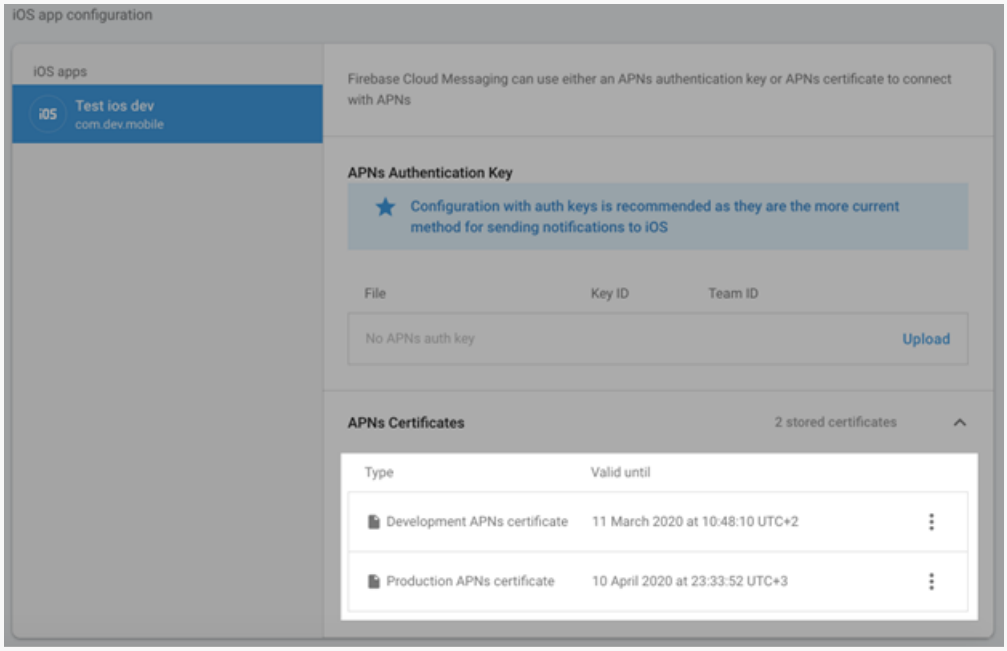
4. **Register your app with App Store Connect (iOS only)**. To do this, follow the procedure in the official Apple documentation.
5. **Install the APNs certificates into Firebase (iOS only)**. iOS projects require you to install the development and production APNs certificates. To do this:

   a. Open the certificate configuration page: Certificate.

   b. Add and install development and production `Apple Push Notification service SSL` certificates into your Mac.

   You also need to generate, download and install provisioning profiles.

   Upload the *.p12 files exported from [*Keychain Access*] in the Firebase settings on the [*Cloud Messaging*] tab.



6. **Install J2ObjC (iOS only)**. The native functionality of the Mobile Creatio application is partly written in Java. The J2ObjC utility for Java code to Objective-C translation is required for shared use on iOS. Learn more about the utility in the official Google documentation. To **install J2ObjC**:

   a. Download the J2ObjC release version archive (2.0.5) to a Mac.

   b. Unpack the archive into your user home directory (`MacintoshHD/Users/MyUser /`). The unpacked archive must contain the `dist` directory.

   c. Rename the unpacked archive directory to `j2objc`.

2. Install and set up the SDKConsole utility

1. **Contact Creatio support** (support@creatio.com) and specify the email address that is or will be associated with your GitLab account. The support team will send you a signup link. After the signup, you will be able to access the SDKConsole project.
2. **Sign up for GitLab**. Open the link you received from the Creatio support and follow the instructions. If you

sign in with a third-party service, make sure that you have a password set up for your GitLab account. To do this, click the profile icon in the top right → [*Edit profile*] → [*Password*].

3. **Install the SDKConsole utility**.

   Install the SDKConsole utility **on Mac**

   Install the SDKConsole utility **on Windows**

4. **Update the SDKConsole utility**.

   a. Back up the `SDK.config` file that contains the user settings. That way you will not have to reconfigure the utility.

   b. Download the utility archive from the Git repository.

   c. Unpack the archive into your utility folder.

   d. Move the `SDK.config` backup to the utility folder.

3. Run the SDKConsole utility

1. **Configure the SDKConsole utility**. Before you run the utility, configure the settings in the `SDK.config` file. Make sure that you do not use a single backslash (`\`) in your file paths. Learn more about the utility settings in a separate article: SDKConsole utility settings.

   Example of the `SDK.config` file

```
{
    "name": "Creatio beta",
    "web_resources_path": "res/web",
    "tasks": ["prepare", "build", "deploy"],
    "use_extended_logging": true,
    "server_url": "https://mysite.creatio.com/",
    "iOS": {
        "repository_path": "https://gitlab.com/bpmonlinemobileteam/ios.git",
        "source_path": "",
        "google_service_info_file": "",
        "launch_storyboard_image_path": "res//LaunchStoryboard.png",
        "app_identifier": "com.myapp.mobile",
        "app_icon_path": "../res/AppIcon.png",
        "version_number": "7.13.9",
        "build_number": "2"
        "app_store_login": "some@gmail.com",
        "certificate_path": "/Users/your_user_dir/ios_distribution.cer",
        "certificate_password": "private_key_password_of_certificate",
        "apple_2FA_specific_password": "apple_specific_password",
        "testflight_changelog": "My what's new"
    },
    "Android": {
```

```
        "build_type": "debug",
        "repository_path": "https://gitlab.com/bpmonlinemobileteam/android.git",
        "source_path": "",
        "google_service_info_file": "",
        "package_name": "com.myapp.mobile",
        "version_number": "1.1.1",
        "build_number": 2,
        "native_resources_path": "res/android/res",
        "key_file": "C:/hybrid/platforms/android/androidappkey",
        "store_password": "android_app_distribution_password",
        "key_alias": "some_key_alias",
        "key_password": "key_password"
    }
}
```

2. **Run the SDKConsole utility**.

   Run the SDKConsole utility **on Mac**

   Run the SDKConsole utility **on Windows**

Description of the solutions for typical errors

View the solutions for typical errors in the table below.

The solutions for typical errors

Error description	
The `Unable to determine Android SDK directory` error on Mac	1. Open Terminal. 2. Run the following commands at the terminal: `echo "export ANDROID_HOME=~/Library/Android/sdk;export PATH=${PATH}:$AN`
The `Couldn't find the specified scheme 'bpm'online'. Please make sure that the scheme is shared…` error when running the build for an iOS project on Mac	Re-run the build. If this does not help, take the following steps: 1. Open the iOS project (`BPMonlineMobile.xcworkspace`) in XCode. 2. Select the current build scheme. If the `[bpm'online]` scheme is not selected, select it 3. Click the current scheme once more and select [*Edit scheme…*] in the list. 4. Select the [*Shared*] checkbox. 5. Close XCode. 6. Run the `./build` command at the Terminal.
Remove the `permission denied` message for my build in the Mac Terminal	Run the following command at the terminal: `chmod -R +x build`

The `function fs.copyFileSync is undefined` error during the build on Mac	Run the following commands at the Terminal:
	`brew link --overwrite node`
	`brew postinstall node`

The `Unable to determine Android SDK directory` error on Windows	Specify the `ANDROID_HOME` environment variable where you need to provide the path to th

SDKConsole utility parameters CLI

Beginner

`name`

Name of your application.

`web_resources_path`

Path to the directory that contains the resources used in the app, i. e., the logo and background on the login page.

`tasks`

Actions the utility executes. This is a string array where you can specify a combination of the tasks.

Available values

`prepare`	Preparation/rebranding of your iOS/Android project. This step makes all the necessary changes. You will get a finished project you can publish in AppStore and Google Play.
`build`	Build the project. You will get an assembled * .ipa iOS app file and\or * .apk Android app file.
`deploy`	Publish the app to TestFlight. iOS only.

`use_extended_logging`

Show detailed logs in the terminal when the utility is running. The recommended value is `true`. If set to `false`, the terminal displays only the currently executed step without details.

`server_url`

Default server. The server URL will be automatically specified on the login page when you log in to the app for the first time.

`repository_path`

Path to the GitLab repository that hosts the original Android/iOS project.

`source_path`

Path to the local Windows/Mac directory where the original Android/iOS project is located. If you specify this parameter the utility uses it in place of the `repository_path` parameter.

`google_service_info_file`

Path to the `GoogleService-Info.plist` (iOS) or `google-services.json` (Android) file required to connect to the Firebase push notification service.

`version_number`

App version in the following format: `0.0.1`.

`build_number`

Build number (string). Always update the build number before you perform the `deploy` task.

`launch_storyboard_image_path`

Path to the image displayed when the application starts (2732x2732 px). iOS only.

`app_identifier`

A unique app ID, for example, `com.myapp.mobile`. This is the Bundle ID specified when you registered the app in App Store Connect. iOS only.

`app_icon_path`

Path to the app icon (1024x1024 px). This is a master image the utility uses to generate the required icons for current iOS devices. iOS only.

`app_store_login`

Account (Apple ID) required to connect to App Store Connect / TestFlight. iOS only.

`certificate_path`

Path to the distribution certificate required when publishing to TestFlight. iOS only.

`certificate_password`

Certificate password. To restore the password, contact the certificate author. iOS only.

`apple_2FA_specific_password`

Specific password. iOS only.

Currently, all Apple accounts support two-factor authentication. To enable third-party services to connect to Apple services, the `app-specific passwords` were added. To get a specific password:

1. Open the https://appleid.apple.com/#!&page=signin URL while signed in to your Apple account.
2. Open the [*Security*] section → [*Generate password...*] command.
3. Follow the instructions to get a new password generated.

`testflight_changelog`

Description of the published changes to TestFlight (what's new). The description is published for the primary app language set in App Store. iOS only.

`build_type`

Build type. Android only.

Available values

`debug`
`release`
`release-unsigned`

`package_name`

A unique app ID, for example, `com.myapp.mobile`. Android only.

`native_resources_path`

Path to app resources, such as the app icon and the startup image. Structure the contents of this directory similarly to the `res` folder in the Android project. The directory can contain subdirectories that have drawable, drawable-xhdpi, and other icons. Android only.

`key_file`

Path to the key file (keystore) required to sign the app. Learn more about signing apps in the official Android documentation. Android only.

`store_password`

Password for the keystore required to sign the app. Android only.

`key_alias`

The key alias. Android only.

`key_password`

The password of the alias from the `key_alias` parameter in the keystore. Android only.