Monaca CLI Tutorial

Monaca CLI is one of the local development environments which can be used to develop Monaca apps locally. Monaca CLI provides command line interface for using Monaca Cloud from your local PC. You can create new local projects, import or clone existing projects from Monaca Cloud to your local PC and start developing Monaca apps with any editors you prefer. Monaca CLI allows you to debug your app with inspector integration and remote build your projects without any setups locally.

Before getting started with this tutorial, you will need the following:

- install Monaca CLI with the following command (read more).

  ```
  $ npm install -g monaca
  ```

- [optional] have a smart mobile device (either iOS or Android) if you want to test on a real device.

- Part 1: Starting a Project
- Part 2: Running Monaca Debugger with Monaca CLI
- Part 3: Building Monaca App
- Part 4: Publishing Monaca App

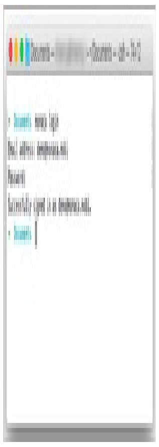
Part 1: Starting a Project

Step 1: Logging into Monaca

1. Open a Command Prompt window (for Windows) or Terminal window (for Mac) and type the following command:

```
$ monaca login
```

2. Then, you will be asked to enter email and password of your Monaca account.



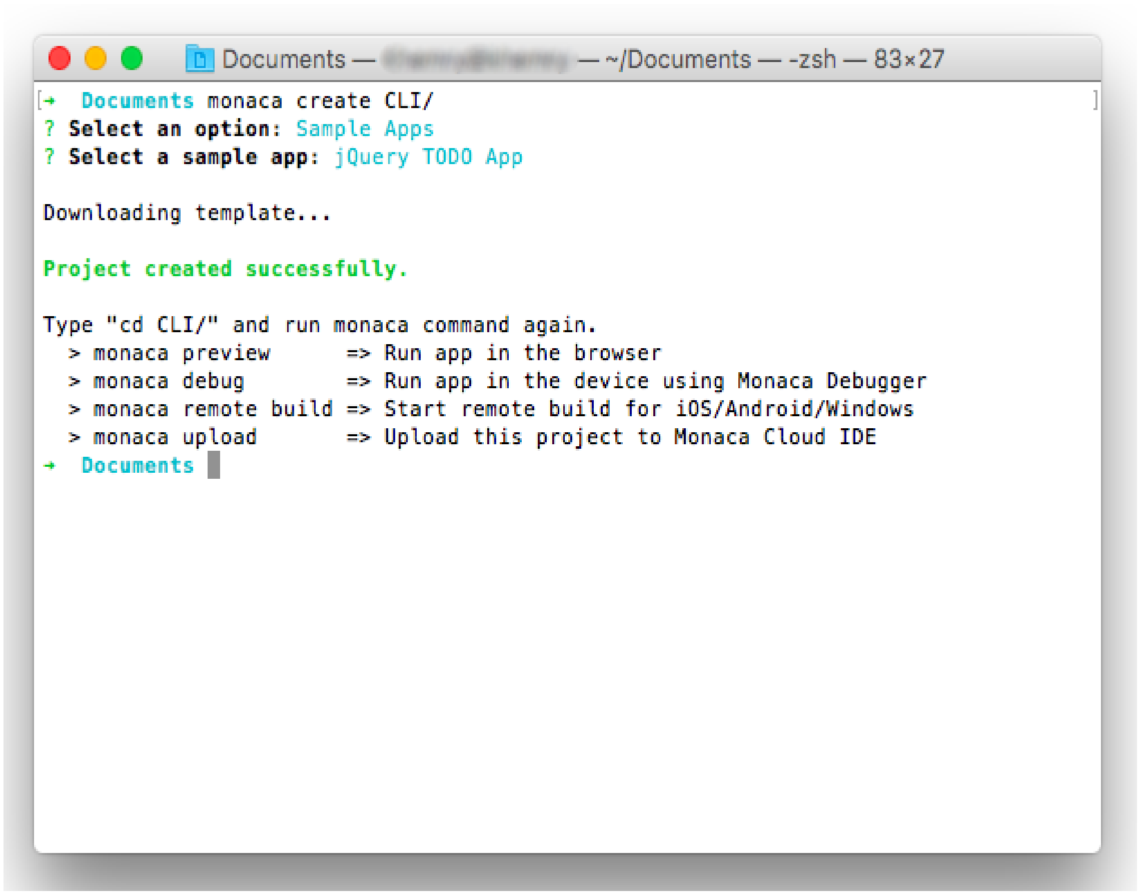
Please use monaca signup command if you need to sign up.

Step 2: Creating a New Project

1. Create a new project by using the command below:

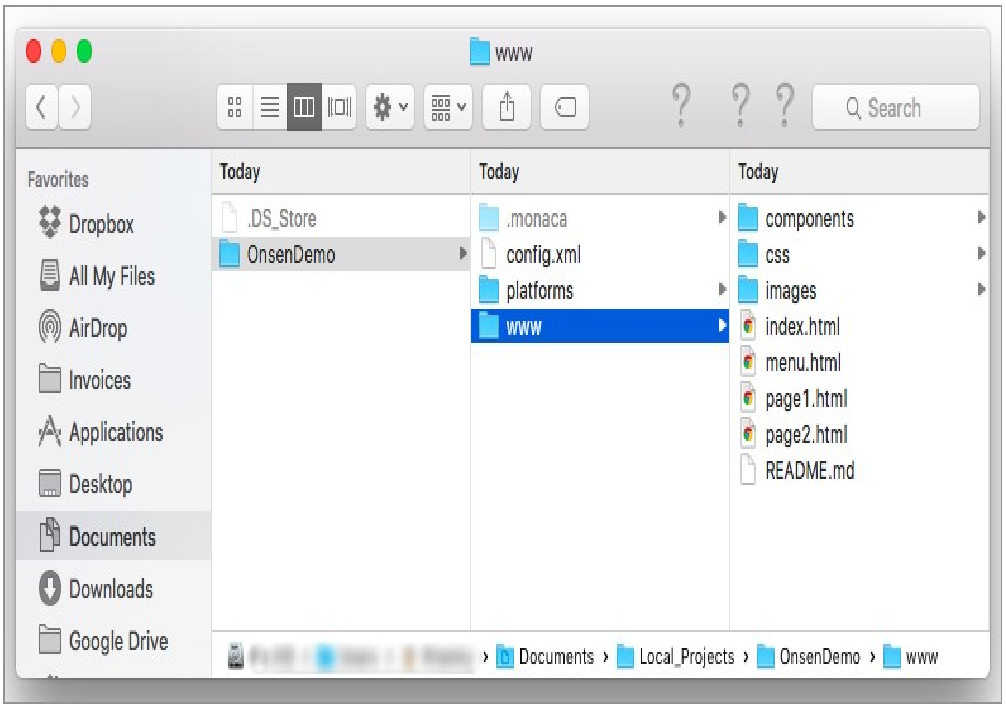
```
$ monaca create PROJECT_DIRECTORY
```

2. Then, you will be asked to choose either `Sample Apps` or `Templates` to create a new Monaca project. In this tutorial, let's choose the `Sample Apps` option. Then, select `jQuery TODO App`.



```
→  Documents monaca create CLI/
? Select an option: Sample Apps
? Select a sample app: jQuery TODO App

Downloading template...

Project created successfully.

Type "cd CLI/" and run monaca command again.
  > monaca preview      => Run app in the browser
  > monaca debug        => Run app in the device using Monaca Debugger
  > monaca remote build => Start remote build for iOS/Android/Windows
  > monaca upload       => Upload this project to Monaca Cloud IDE
→  Documents
```

3. After creating the project, the project's folder will be created on your PC. You can then use any local editors to develop your Monaca projects.



You can also use Monaca CLI to import or clone your existing projects from Monaca Cloud.

Part 2: Running Monaca Debugger with Monaca CLI

Monaca Debugger is a powerful application for testing and debugging your Monaca applications on real devices in real time.

When developing Monaca apps on your local PC, assuming that your local PC is successfully paired with Monaca Debugger, all changes made to your project files will be pushed into your Monaca Debugger as soon as you save those changes.

Before Getting Started

Please install Monaca Debugger on your device.

Available on the App Store GET IT ON Google play

Please refer to Monaca Debugger Installation for other platforms.

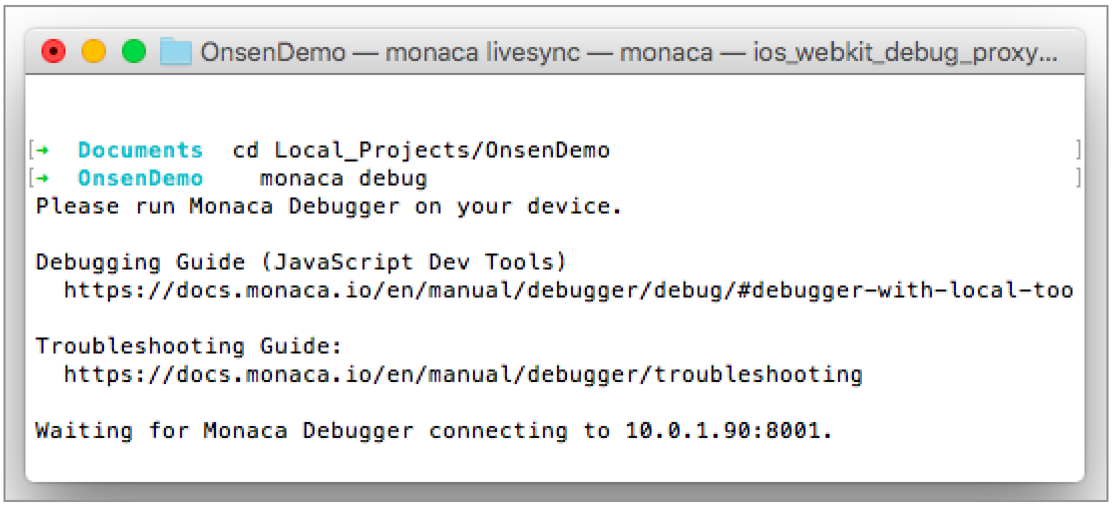
Step 1: Pairing Monaca Debugger with Local PC

In order to debug/test a local Monaca app, you need to pair local PC with Monaca Debugger.

Before connecting Monaca Debugger to the local PC, please pay attention to the following points:

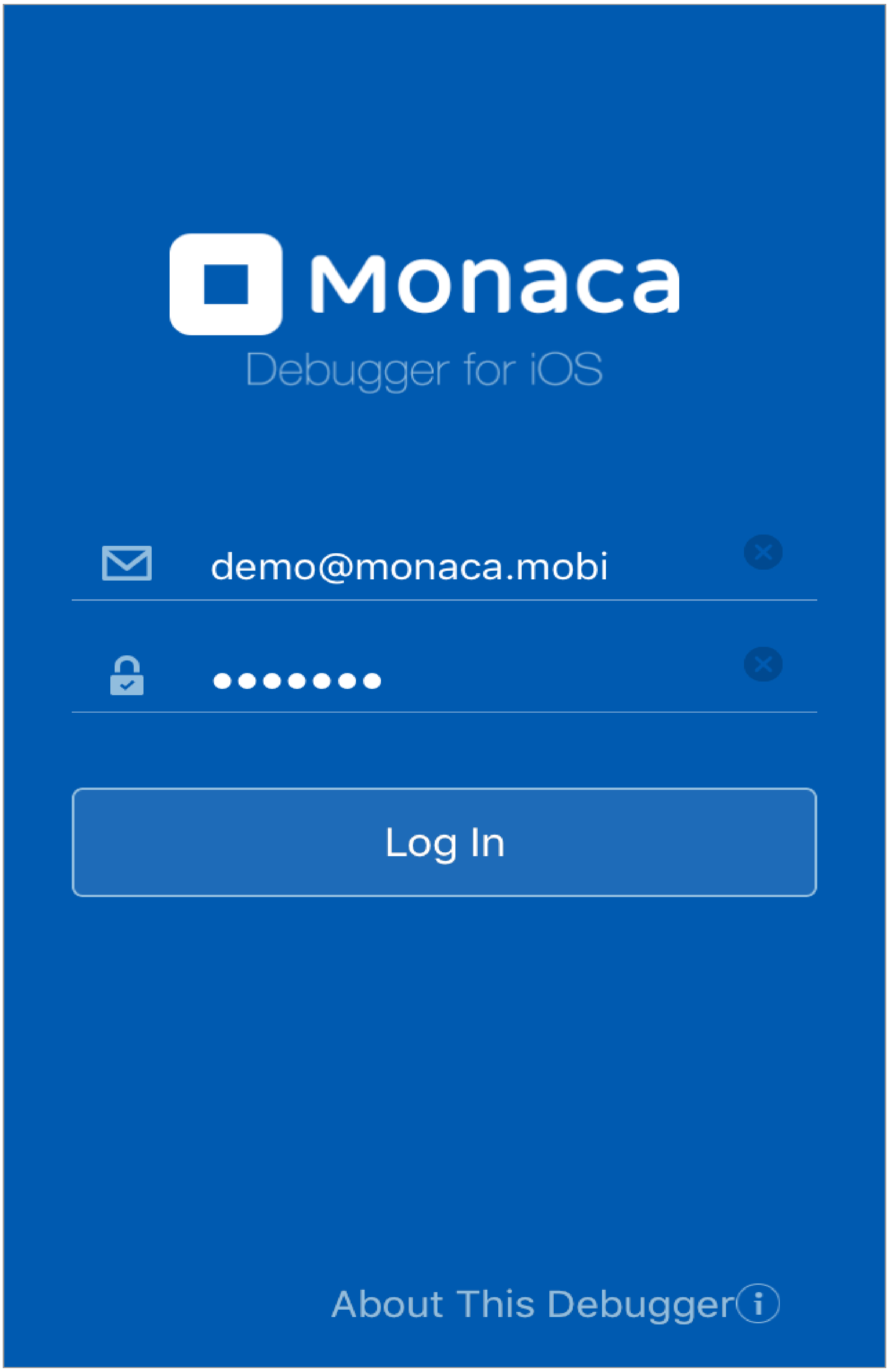
1. Monaca Debugger and the local PC have to connect to the same network connection (LAN or Wi-Fi).
2. Use the same Monaca account for either Monaca Debugger and the local PC.
3. Disable the local PC's firewall.

1. In the command window, navigate to your project folder and type monaca debug command to connect to your Monaca Debugger. Then, Monaca CLI will wait for requests from debugger.

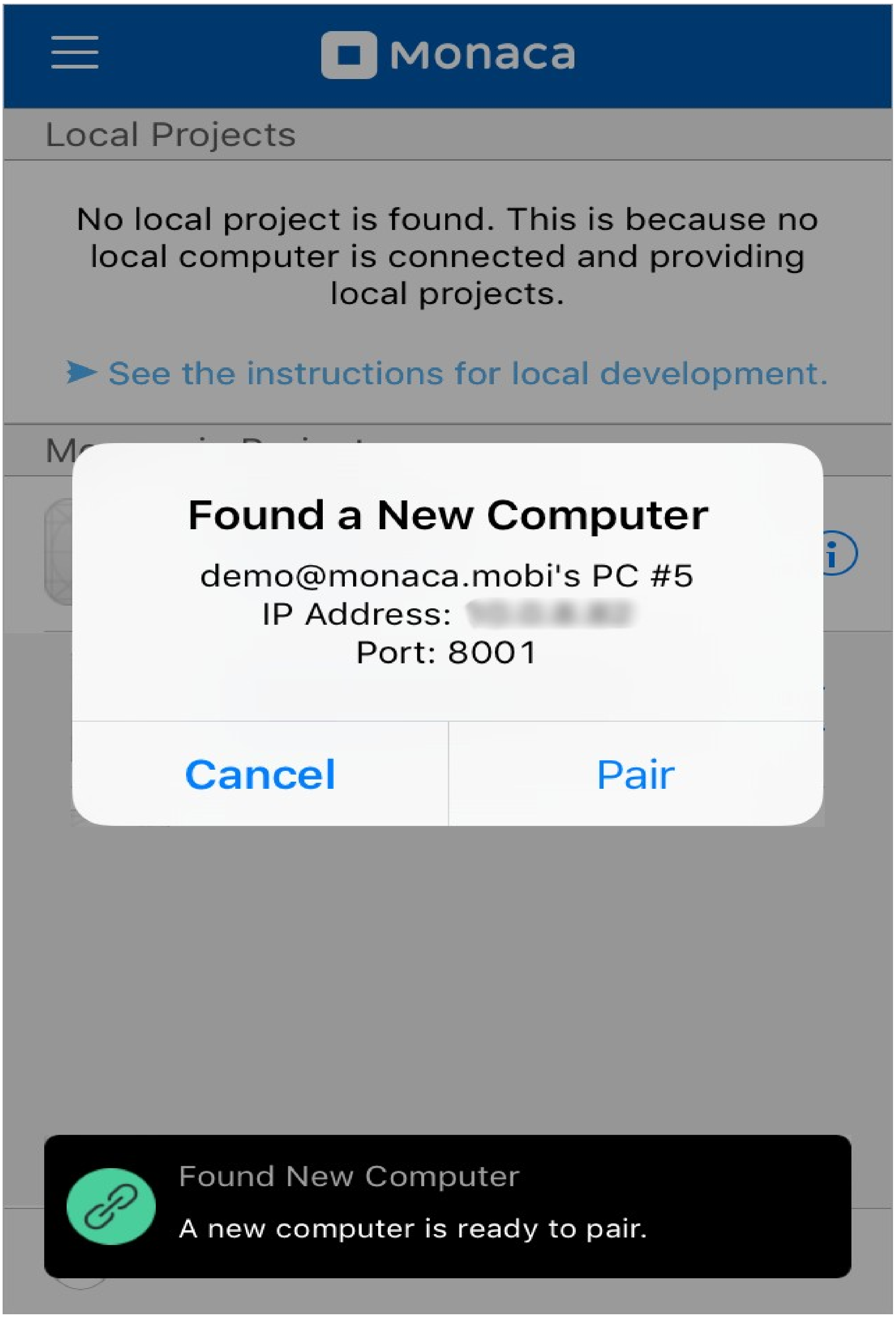


```
OnsenDemo — monaca livesync — monaca — ios_webkit_debug_proxy...

→  Documents  cd Local_Projects/OnsenDemo
→  OnsenDemo   monaca debug
Please run Monaca Debugger on your device.

Debugging Guide (JavaScript Dev Tools)
  https://docs.monaca.io/en/manual/debugger/debug/#debugger-with-local-too

Troubleshooting Guide:
  https://docs.monaca.io/en/manual/debugger/troubleshooting

Waiting for Monaca Debugger connecting to 10.0.1.90:8001.
```

2. Launch Monaca Debugger app and sign in using your Monaca account information. Make sure you are using the same account information you use for Monaca CLI earlier.



3. Then, a popup message prompting you to pair the debugger with the local PC will appear.



Local Projects

No local project is found. This is because no local computer is connected and providing local projects.

➤ See the instructions for local development.

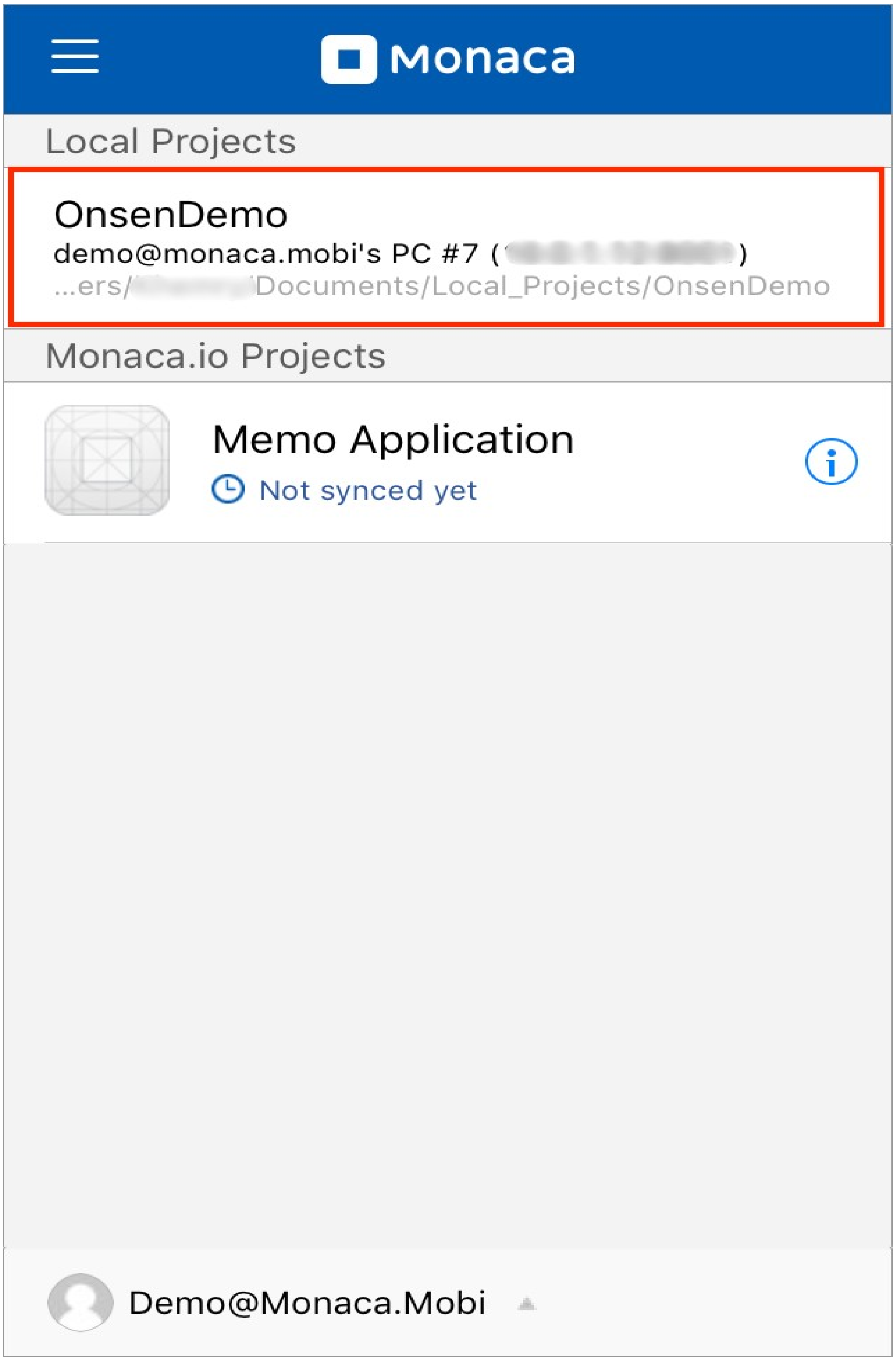
Found a New Computer

demo@monaca.mobi's PC #5
IP Address: [redacted]
Port: 8001

Found New Computer

A new computer is ready to pair.

4. If your pairing is successful, your local project name will appear under `Local Projects` in Monaca Debugger. However, if you fail the pairing, please refer to Fail to Pair Monaca Debugger.


Monaca

Local Projects

OnsenDemo
demo@monaca.mobi's PC #7 ([redacted])
...ers/[redacted]/Documents/Local_Projects/OnsenDemo

Monaca.io Projects

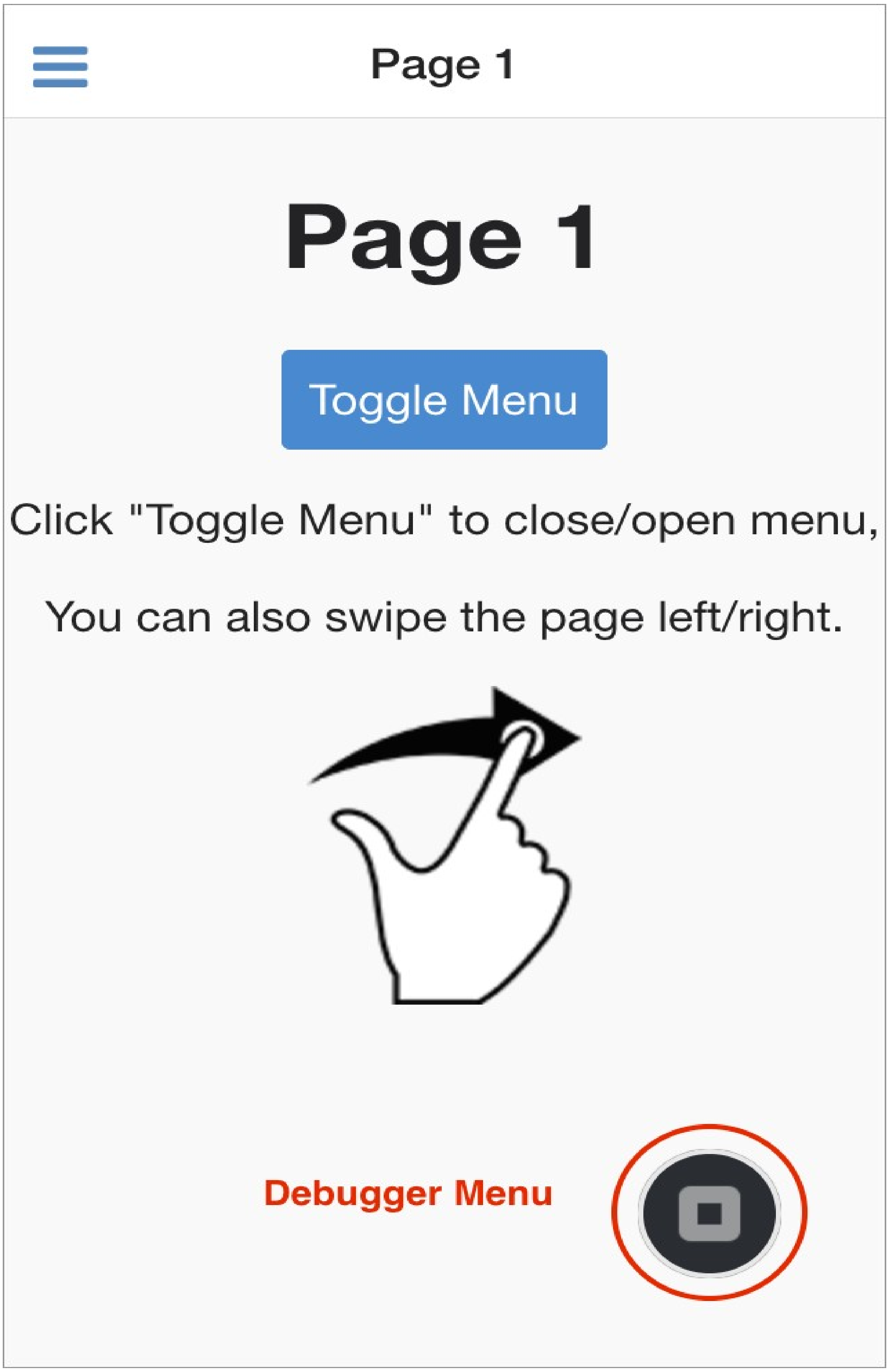
Memo Application
Not synced yet

Demo@Monaca.Mobi

In order to stop debugging and unpair the debugger, press `Ctrl+c`.

Step 2: Running a Project on Monaca Debugger

1. From the Local Projects list in Monaca Debugger, click on your project name to run it.
2. Now the project should be running as shown in the screenshot below. Use Back button within Debugger Menu button to go back to the Project List screen.



Page 1

Click "Toggle Menu" to close/open menu,

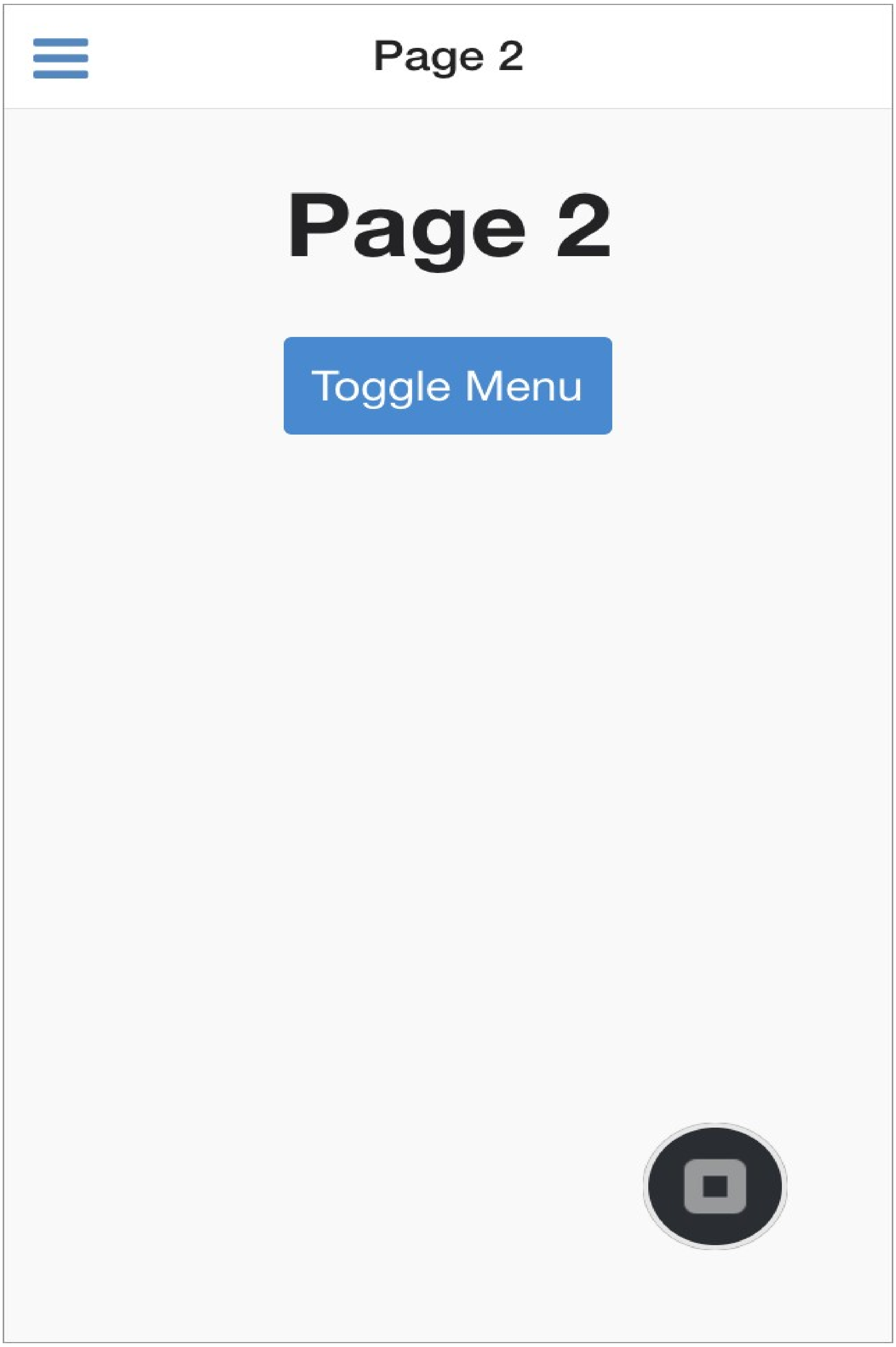
You can also swipe the page left/right.

Step 3: Real-time Update between Monaca CLI and Debugger

1. Run the project on the debugger.
2. You can now try making some changes to your file. For example, try changing the starting page of the sample app to Page 2. In order to do this, please open the `index.html` file. Then, change the `main-page` attribute (inside `<ons-sliding-menu>` tag) to `page2.html` and save your changes. The updated code should look like this:

```
...
<ons-sliding-menu
    var="app.slidingMenu"
    menu-page="menu.html"
    main-page="page2.html"
    side="left" type="overlay"
    max-slide-distance="200px">
</ons-sliding-menu>
...
```

3. If your PC is still connected to Monaca Debugger, it will automatically refresh the updates. Now your starting page should be Page 2.



Please refer to Debugger Functionalities to explore the other functions provided by Monaca Debugger.
You can also use USB debugging with Monaca CLI. Please refer to USB Debugging with Monaca .

That’s it! That’s how easy it is to use Monaca Debugger. Please try to make more changes to your project and see how it runs on the debugger. Enjoy developing with Monaca!

Next:

- Part 3: Building Monaca App

Part 3: Building Monaca App

In this page, we will cover the two following topics:

1. Building a Monaca App for iOS
2. Building a Monaca App for Android

For more information on how to build Monaca Apps for other platforms, please refer to Build.

Building a Monaca App for iOS

In this section, we will talk about how to create a Debug Build of your Monaca app for iOS which will be installed on a development device. For more information about other types of build, please refer to Types of Build.

Prerequisite

1. You must enroll in Apple Developer Program.
2. After enrolling in the program, you will be able to create the following items which are required to create a Debug build in Monaca:
    - `App ID` (see How to Register App ID)
    - `Development Certificate` (see How to Generate Certificates)
    - `Development Provisioning Profile` (see How to Create Provisioning Profiles)

Step 1: Configuring iOS App Settings

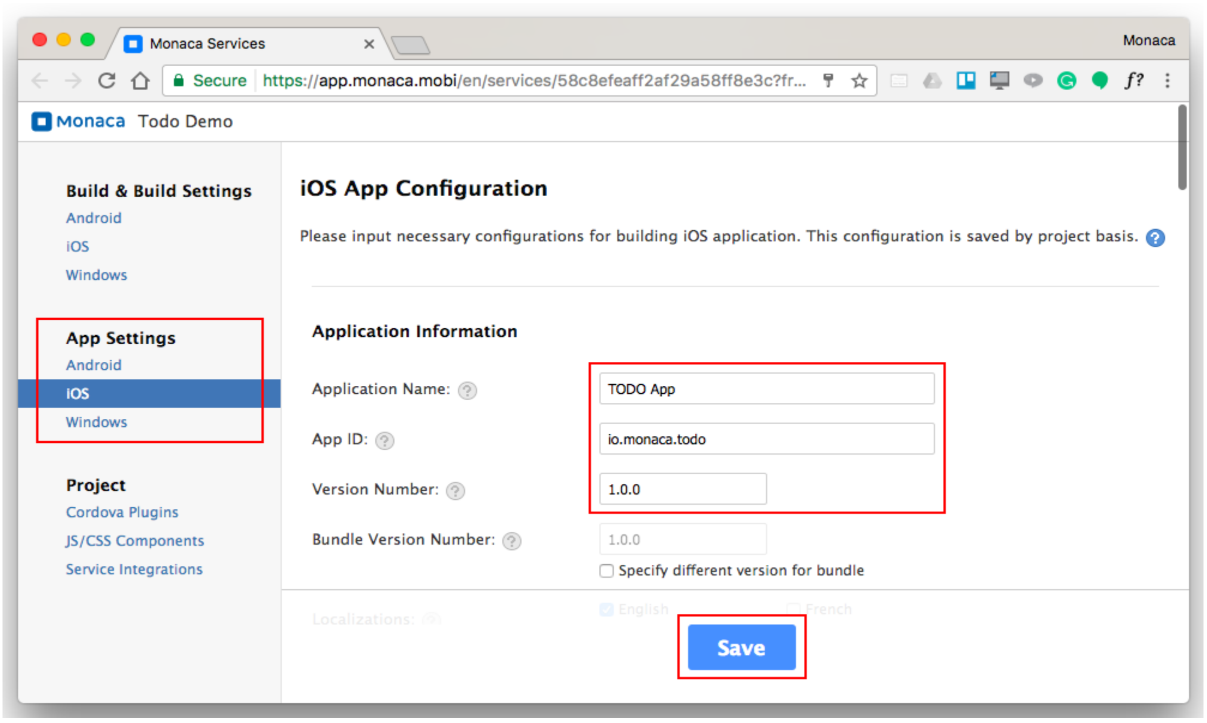
1. In the command window, navigate to your project folder and use monaca remote build command. This command will automatically upload your local project files to Monaca Cloud. Please type the following command to open an interactive build interface in your browser.

```
$ monaca remote build --browser
```

2. From the build interface under the `App Settings` on the left menu, select iOS.
3. Fill in the necessary information of your app:
    - `Application Name`: a name representing your app publicly such as in the Market.
    - `App ID`: a unique ID representing your app. It is recommended to use reverse-domain style (for example, mobi.monaca.appname) for App ID. Only alphanumeric

characters and periods (at least one period must be used) are allowed. Each segment separated by a period should begin with an alphabetic character.

- `Version Number`: a number representing the version of your app which will be required when uploading (publishing process) your application via App Store Connect later. It needs 3 numbers separated by dots (for example, 1.10.2). Each number should be in `[0-99]`.
- The remaining information is optional. In this page, you can also configure icon, splash screen and other configurations.



iOS App Configuration

Please input necessary configurations for building iOS application. This configuration is saved by project basis.

Application Information

Application Name: TODO App

App ID: io.monaca.todo

Version Number: 1.0.0

Bundle Version Number: 1.0.0

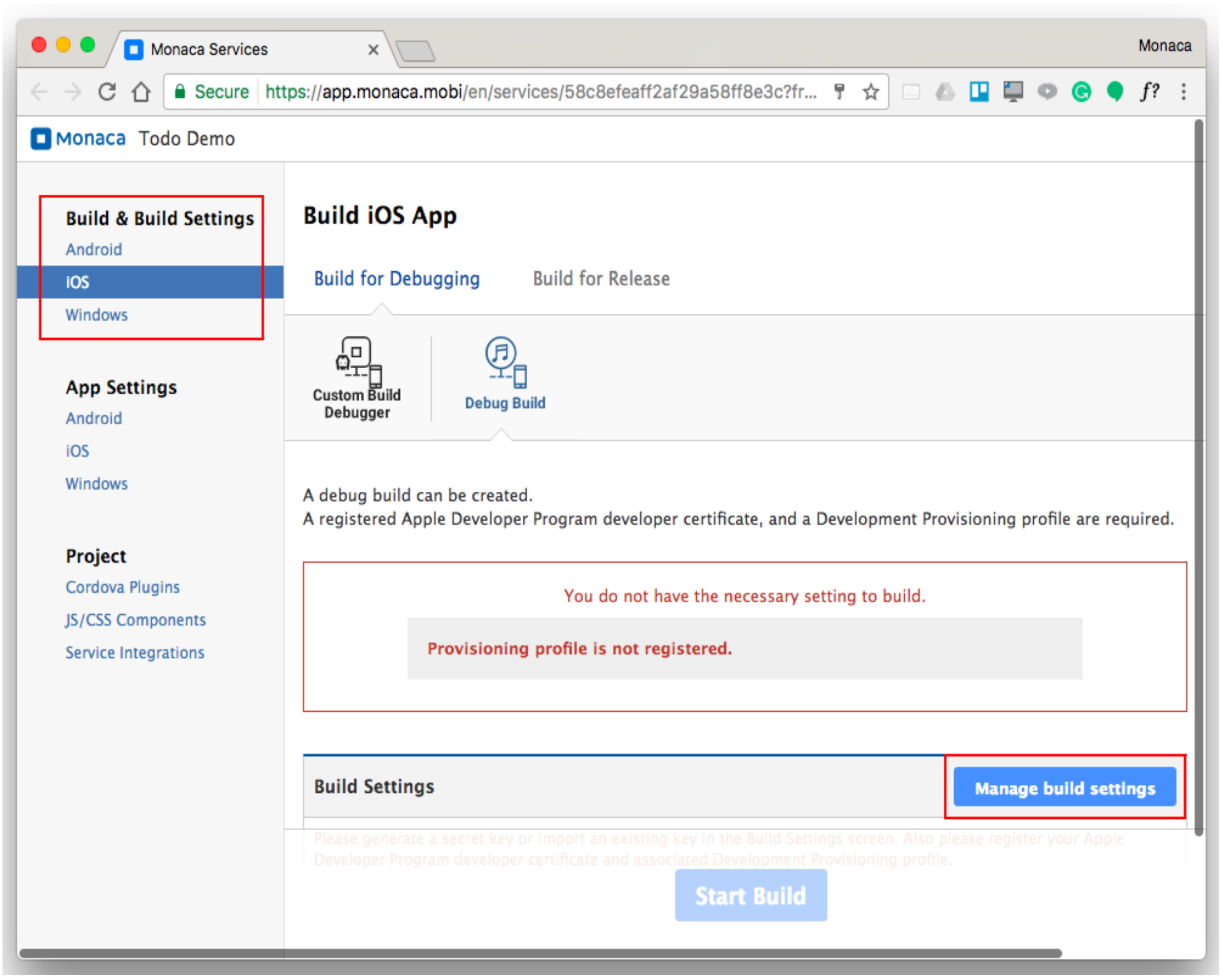
☐ Specify different version for bundle

The App ID in Monaca Build Settings must be the same as the App ID you have registered in iOS Provisioning Portal. This App ID (in Monaca Build Settings) cannot contain an asterisk (*); otherwise, the build will fail.

4. After finishing the configurations, click Save.

Step 2: Configuring iOS Build Settings

1. From the build interface under the `Build & Build Settings` on the left menu, select iOS. Then, select Manage build settings.



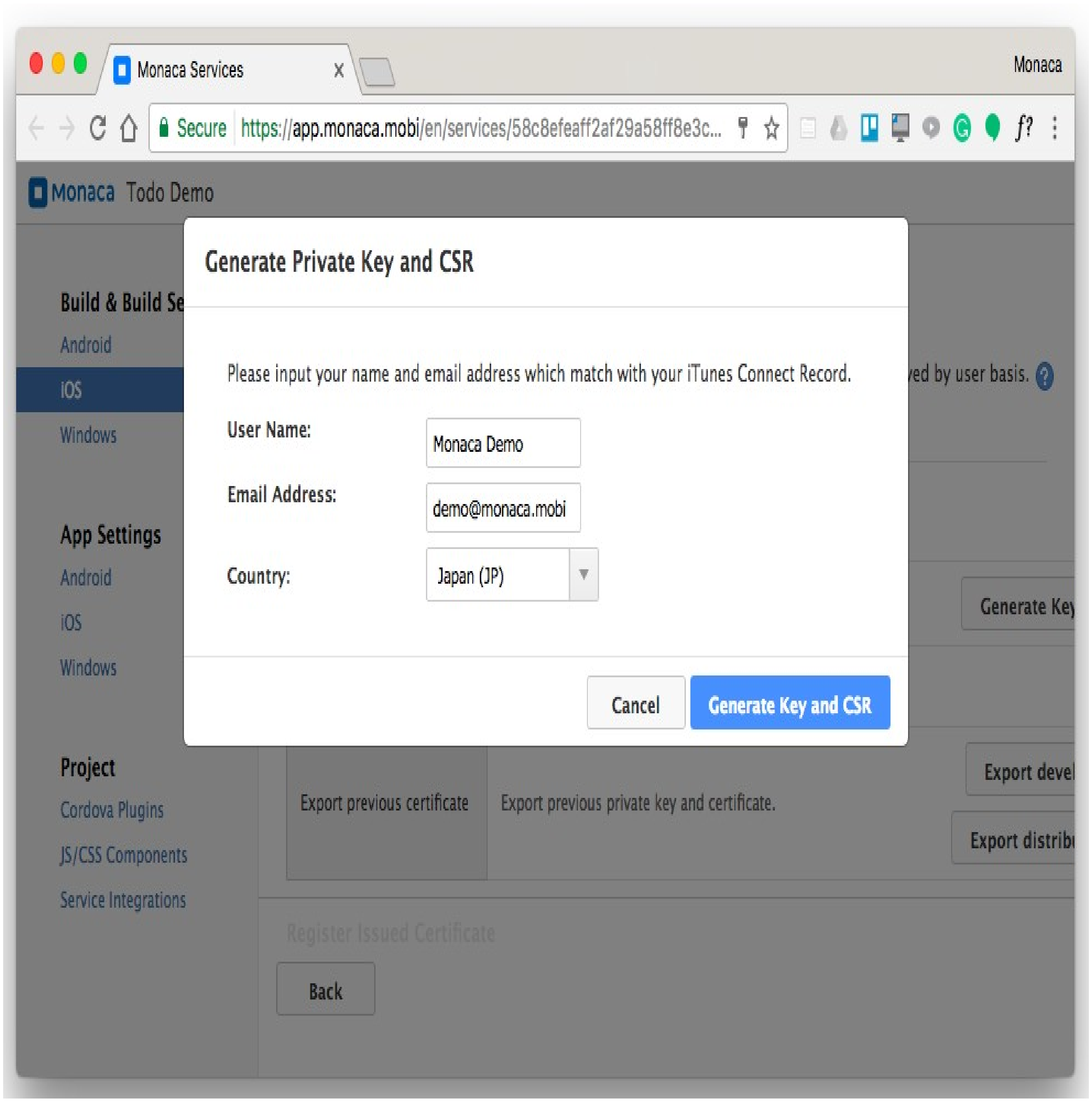
A debug build can be created.
A registered Apple Developer Program developer certificate, and a Development Provisioning profile are required.

You do not have the necessary setting to build.

Provisioning profile is not registered.

Please generate a secret key or import an existing key in the Build Settings screen. Also please register your Apple Developer Program developer certificate and associated Development Provisioning profile.

2. Click on Generate Key and CSR button and fill in your Apple ID information (user name and email address) and country. Then, click Generate Key and CSR button. After that, you will be asked to download the CSR file. You can also import an existing Private Key if you have one.



Generate Private Key and CSR

Please input your name and email address which match with your iTunes Connect Record.

User Name: Monaca Demo

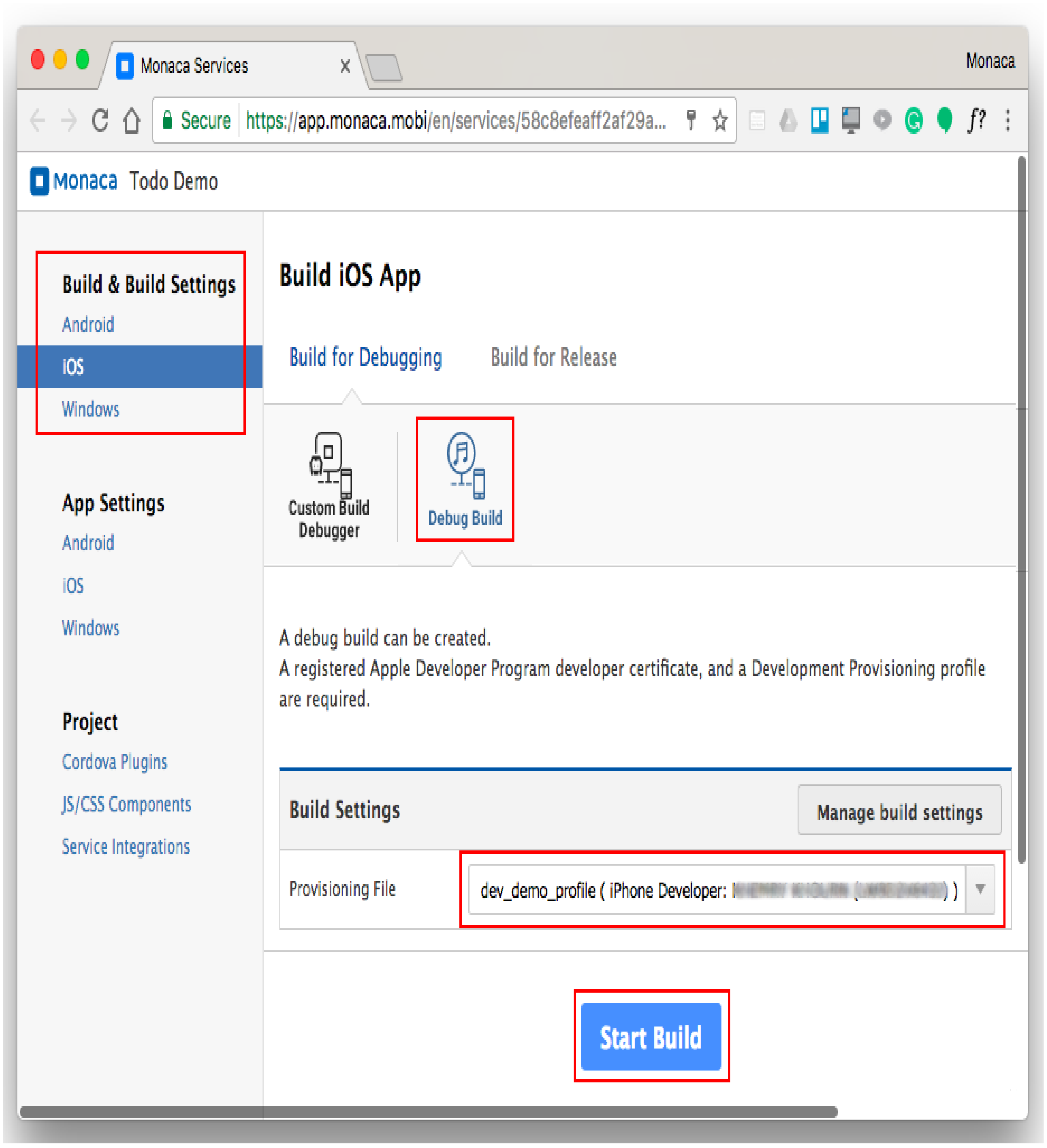
Email Address: demo@monaca.mobi

Country: Japan (JP)

If you import an existing private key, you need to use the certificates which are issued based on that imported private key. However, if you create a new private key and CRS file, you will need to use the new CRS file to issue new certificates.

Step 3: Building the App

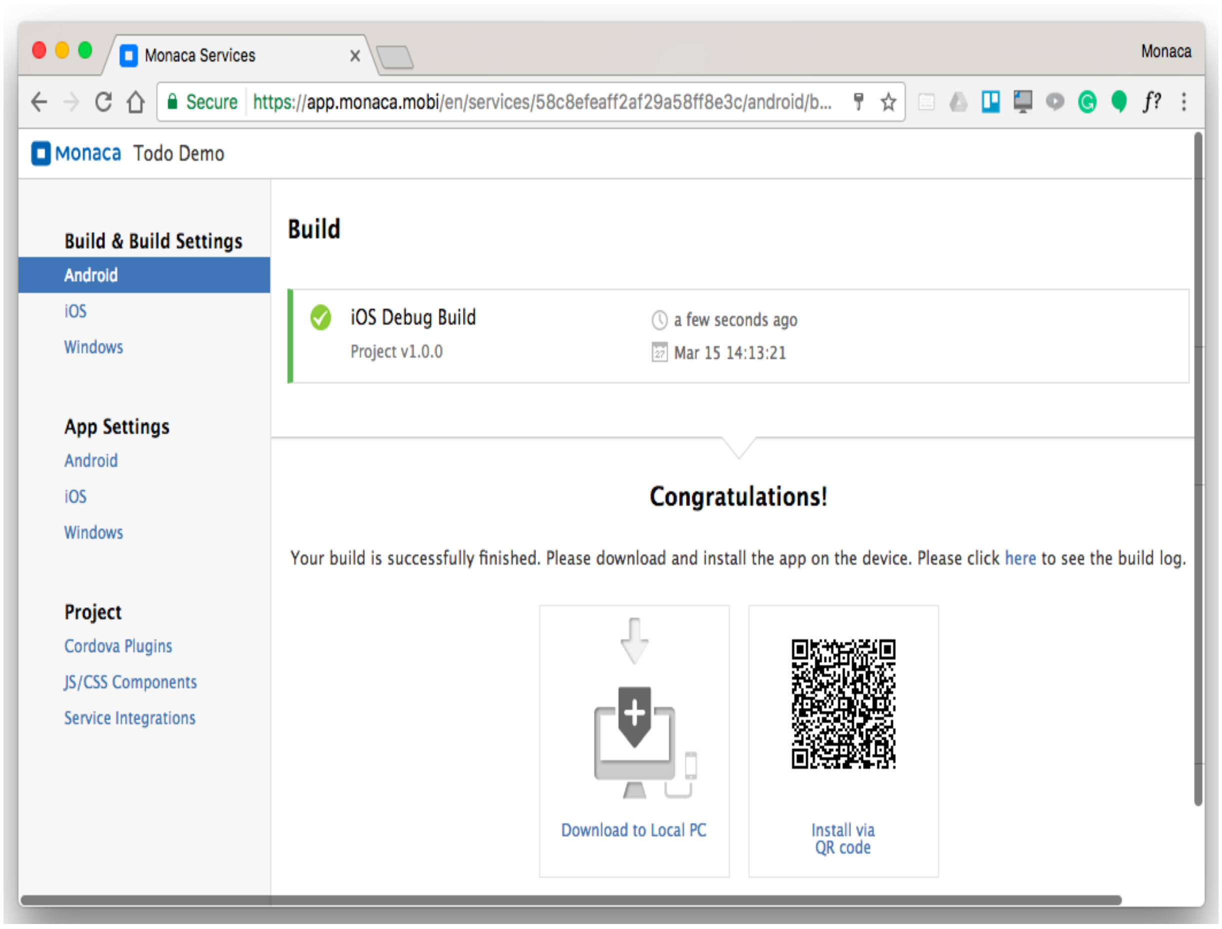
1. From the build interface under the `Build & Build Settings` on the left menu, select iOS. Then, select the `Debug Build` option and the corresponding provisioning profile. Then, click Start Build button.



A debug build can be created.
A registered Apple Developer Program developer certificate, and a Development Provisioning profile are required.

Provisioning File: dev_demo_profile (iPhone Developer: [redacted])

2. It may take several minutes for the build to complete. Please wait. The following screen will appear after the build is complete.



Congratulations!

Your build is successfully finished. Please download and install the app on the device. Please click here to see the build log.

Step 4: Installing the Built App

There are 3 ways to install the debug built app such as:

1. Download the built app and use Apple Configurator 2 to install the built app on your iOS device. (Mac Only)
2. Install via QR code.
3. Install via cofigured deployment services.

Building a Monaca App for Android

In this section, we will talk about how to create a Debug Build of your Monaca app for Android. For more information about other types of build, please refer to Types of Build.

Step 1: Configuring Android App Settings

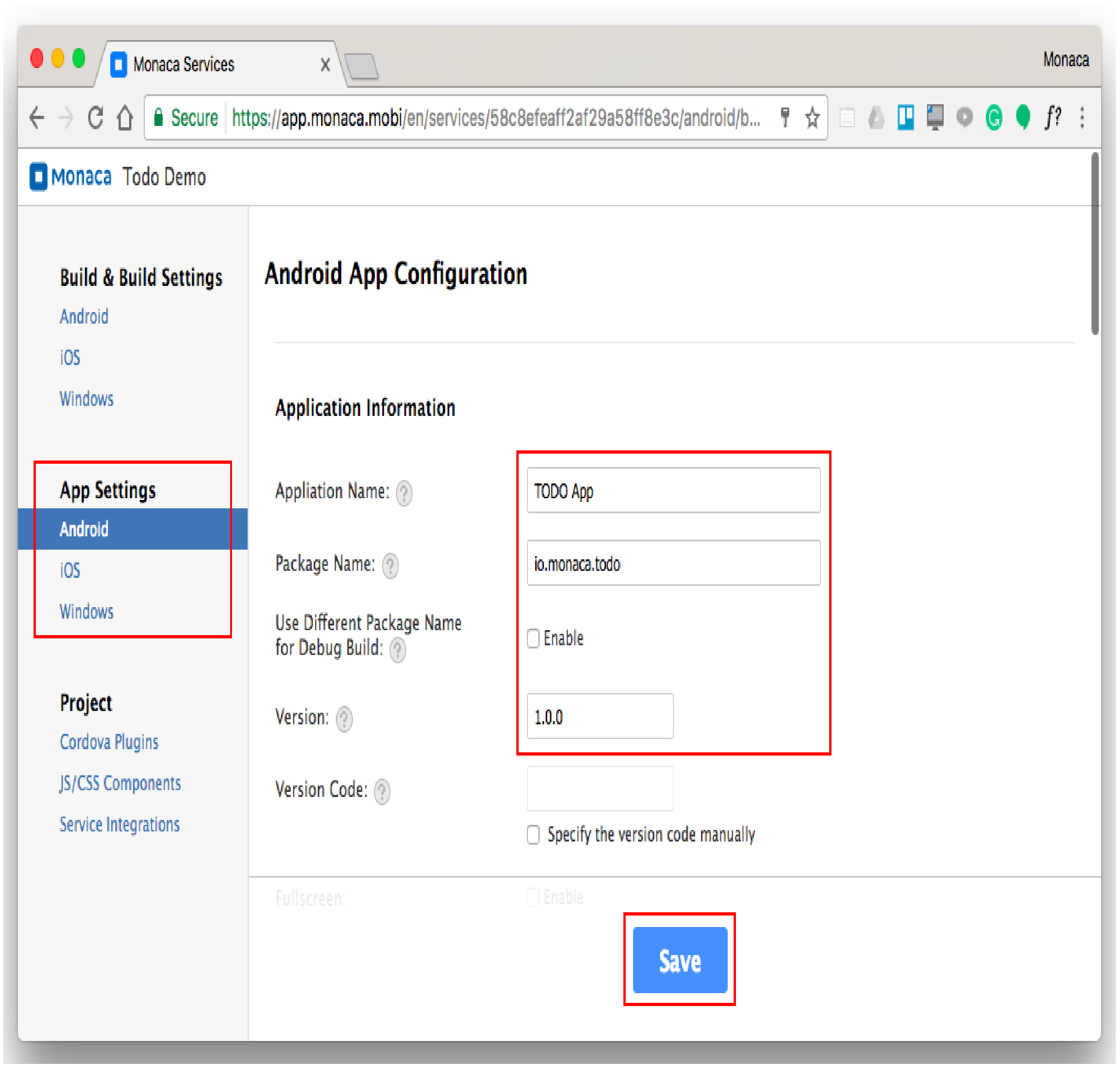
1. In the command window, navigate to your project folder and type monaca remote build command. This command will automatically upload your local project files to Monaca Cloud. Please type the following command to open an interactive build interface in your browser.

```
$ monaca remote build --browser
```

2. From the build interface under the `App Settings` on the left menu, select Android.

3. Fill in the necessary information of your app:

- `Application Name`: a name representing your app publicly such as in the Market.
- `Package Name`: a unique ID representing your app. It is recommended to use reverse-domain style (for example, mobi.monaca.appname) for App ID. Only alphanumeric characters and periods (at least one period must be used) are allowed. Each segment separated by a period should begin with an alphabetic character.
- `Version Number`: a number representing the version of your app. It needs 3 numbers separated by dots (for example, 1.10.2). Each number should be in `[0-99]`.
- `Use Different Package Name for Debug Build`: if checked, the package name of the debug-built app and custom-built debugger are different. In other words, the package name of debug-built app will have `.debug` extension, and the one for project debugger will have `.debugger` extension. However, this option is disable by default because it made some plugins impossible to be debugged due to the fact that they are tied to exact package names (eg. in-app purchase).
- The remaining information is optional. In this page, you can also configure icon, splash screen and other configurations.


Monaca Services
Monaca
https://app.monaca.mobi/en/services/58c8efeaff2af29a58ff8e3c/android/b...
Monaca Todo Demo

Build & Build Settings
Android
iOS
Windows

App Settings
Android
iOS
Windows

Project
Cordova Plugins
JS/CSS Components
Service Integrations

Android App Configuration

Application Information

Appliation Name:	TODO App
Package Name:	io.monaca.todo
Use Different Package Name for Debug Build:	☐ Enable
Version:	1.0.0
Version Code:	☐ Specify the version code manually
Fullscreen:	☐ Enable

Save

4. After finishing the configurations, click Save.

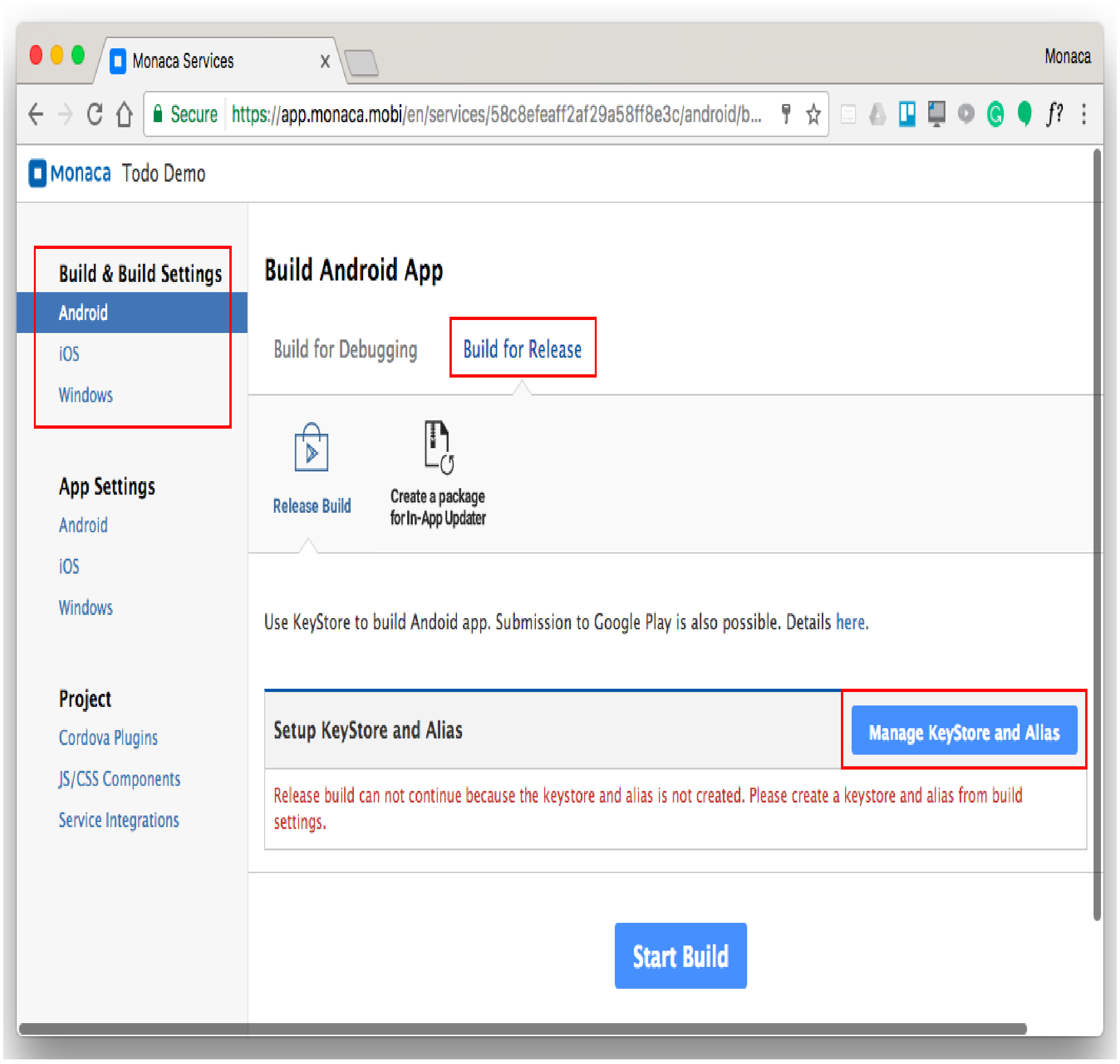
Step 2: Configuring Android KeyStore

Android KeyStore is used for storing the keys (Alias) needed to sign a package. When a KeyStore is lost or it is overwritten by another KeyStore, it is impossible to re-sign the signed package with the same key. One KeyStore can contain multiple aliases, but only one alias is used for code-sign an application.

Android KeyStore is required in order to create a Release build. However, it's not necessary in order to create a Debug build.

In order to configure a new Android KeyStore in Monaca, please do as follows:

1. From the build interface under the `Build & Build Settings` on the left menu, select Android. Then, choose `Build for Release` and select Manage KeyStore and Alias.


Monaca Services
Secure https://app.monaca.mobi/en/services/58c8efeaff2af29a58ff8e3c/android/b...
Monaca Todo Demo

Build & Build Settings
Android
iOS
Windows

App Settings
Android
iOS
Windows

Project
Cordova Plugins
JS/CSS Components
Service Integrations

Build Android App

Build for Debugging | Build for Release

Release Build | Create a package for In-App Updater

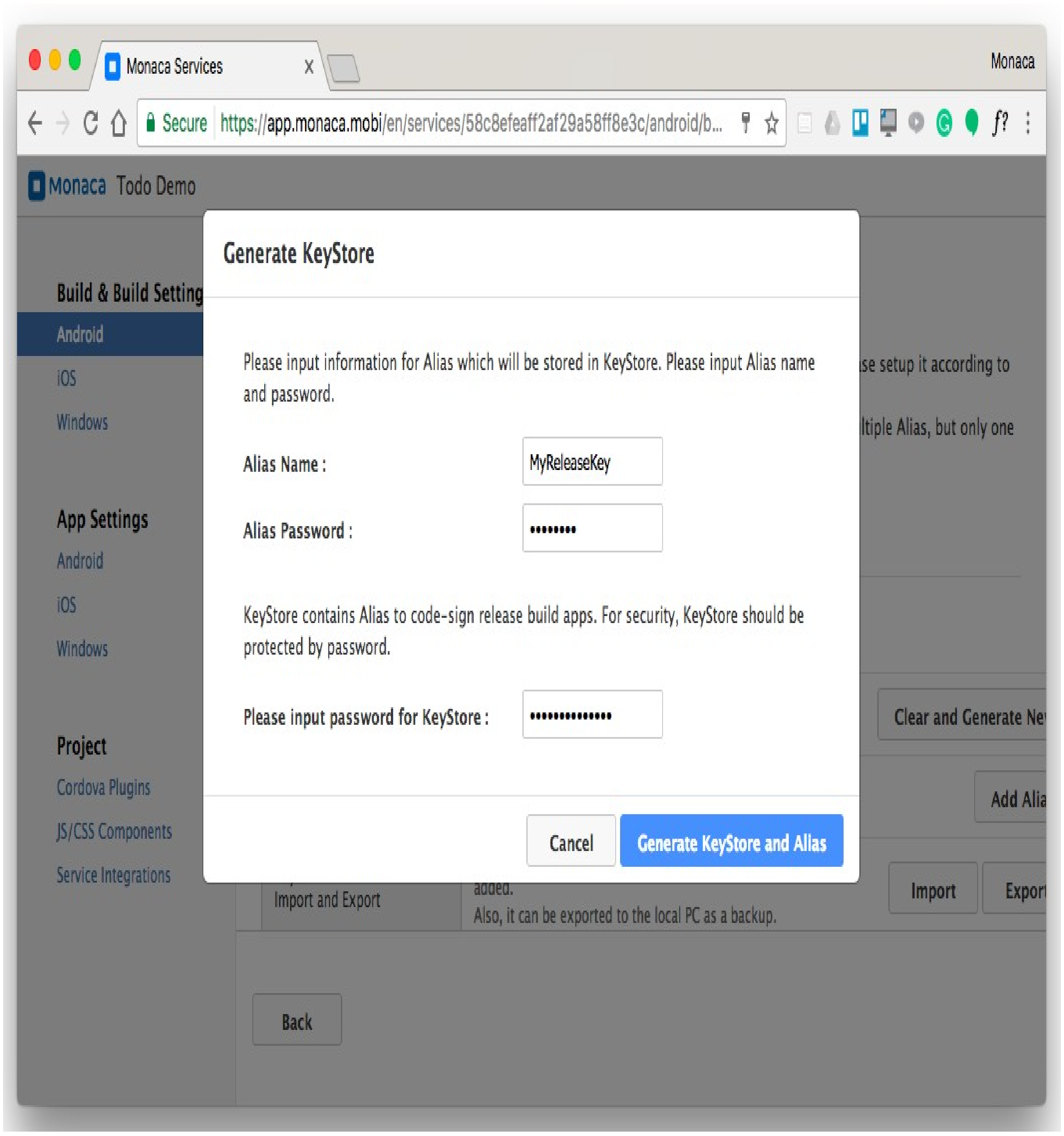
Use KeyStore to build Andoid app. Submission to Google Play is also possible. Details here.

Setup KeyStore and Alias | Manage KeyStore and Alias

Release build can not continue because the keystore and alias is not created. Please create a keystore and alias from build settings.

Start Build

2. The KeyStore can either be created or imported. In this tutorial, we assume that you need to create a new KeyStore. Therefore, click on Clear and Generate New button. Then, the following screen will appear:


Generate KeyStore

Please input information for Alias which will be stored in KeyStore. Please input Alias name and password.

Alias Name : MyReleaseKey

Alias Password : ••••••••

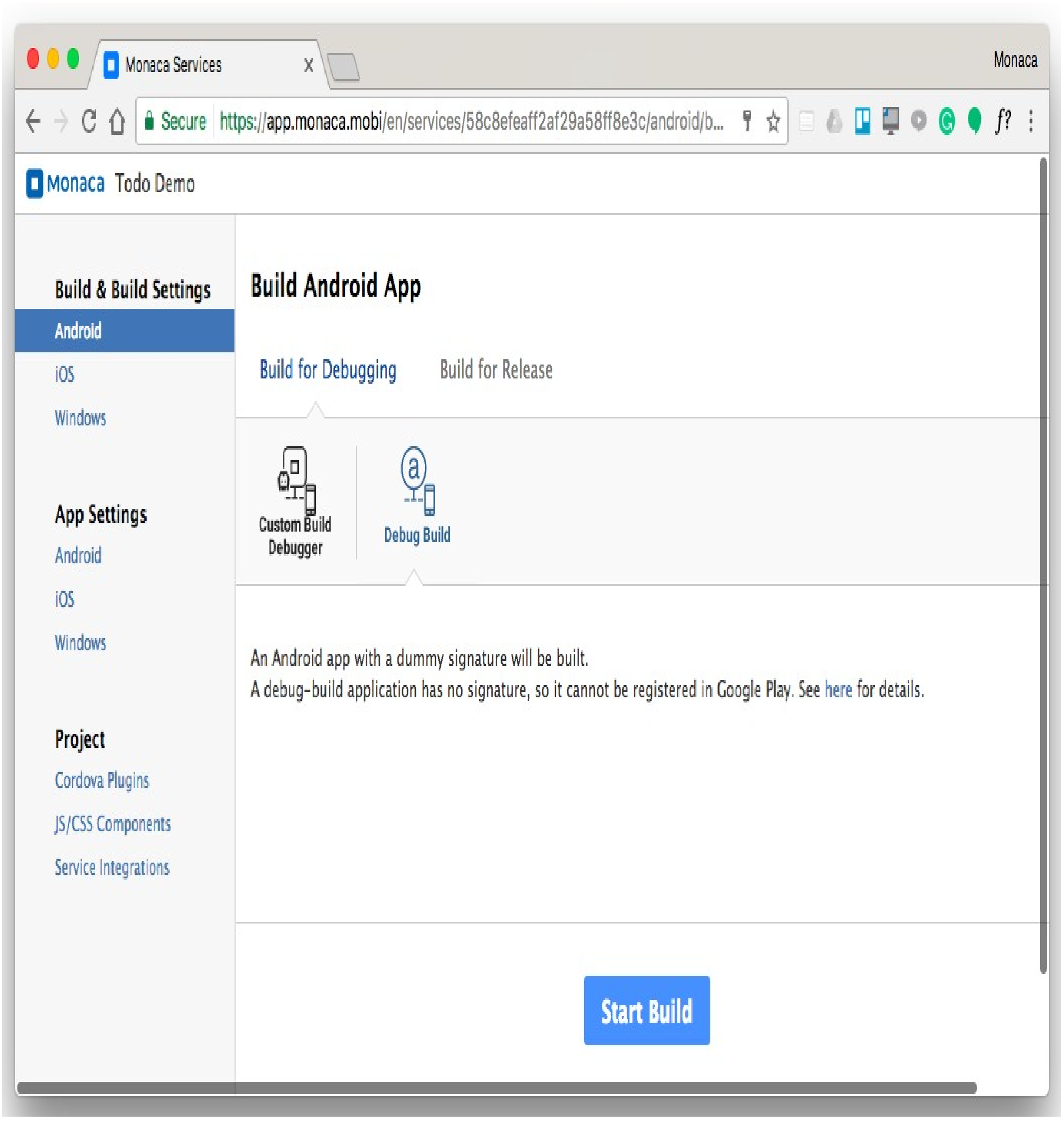
KeyStore contains Alias to code-sign release build apps. For security, KeyStore should be protected by password.

Please input password for KeyStore : ••••••••••••••

3. Fill in the necessary information related to the KeyStore such as:
   - `Alias`: key information stored in the KeyStore which is used to sign an app package.
   - `Password`: password for the Alias.
   - `KeyStore Password`: password for the new KeyStore.
4. Then, click Generate KeyStore and Alias button.

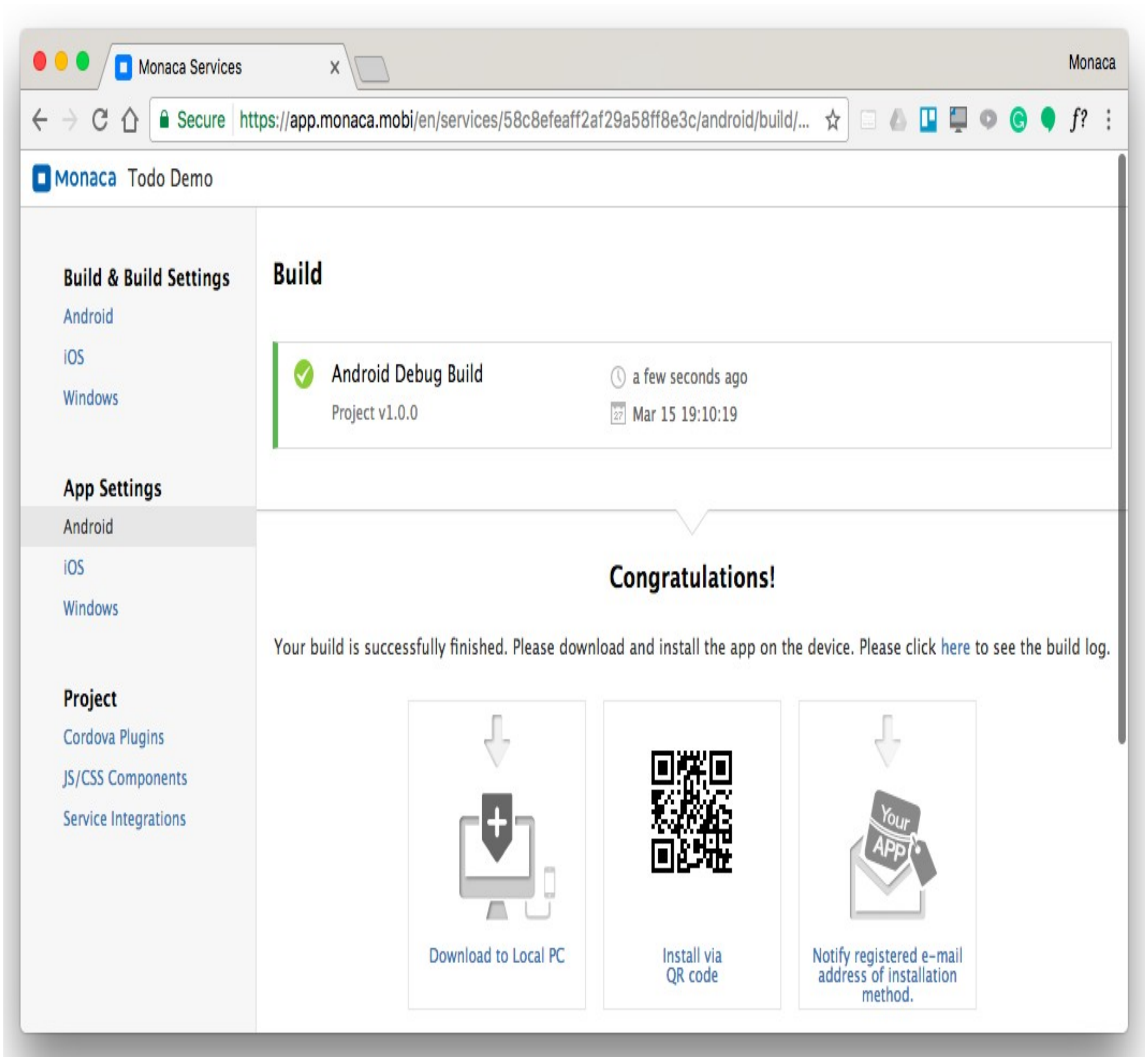
Step 3: Building the App

1. From the build interface under the `Build & Build Settings` on the left menu, select Android. Then, select the `Debug Build` option. Then, click Start Build button.



An Android app with a dummy signature will be built.
A debug-build application has no signature, so it cannot be registered in Google Play. See here for details.

2. It may take several minutes for the build to complete. Please wait. The following screen will appear after the build is successfully completed.


Monaca Todo Demo

Build & Build Settings
Android
iOS
Windows

App Settings
Android
iOS
Windows

Project
Cordova Plugins
JS/CSS Components
Service Integrations

Build

Android Debug Build
Project v1.0.0
a few seconds ago
Mar 15 19:10:19

Congratulations!

Your build is successfully finished. Please download and install the app on the device. Please click here to see the build log.

Monaca Services
https://app.monaca.mobi/en/services/58c8efeaff2af29a58ff8e3c/android/build/...
Download to Local PC
Install via QR code
Notify registered e-mail address of installation method.

Step 4: Installing the Built App

There are 5 ways you can install the built app such as:

1. using Network Install
2. install via QR Barcode.
3. downloading the built app directly to your computer and installing it via USB cable.
4. sending the URL to download the built app to your registered email address.
5. installing via configured deployment services.

Next:

- Part 4: Publishing Monaca App

Part 4: Publishing Monaca App

Publishing for App Store

Please refer to App Store Distribution.

Publishing for Google Play

Please refer to Google Play Distribution.

For more information regarding the distribution of Monaca Apps for other platforms, please refer to Distribution.

App Store Distribution

Before publishing iOS apps to the App Store, your apps need to be reviewed and accepted by Apple. App Store Connect is where you can upload your apps, edit their descriptions and metadata, and view finances earned. Once, your apps are submitted, you will need to wait for the review from Apple. Apple will check if your apps are eligible or qualified to be in the App Store. Usually, it takes two weeks for the review. Therefore, please wait patiently for it.

- App Store Connect Guide
- Monaca Upload Feature

App Store Connect Guide

This document describes how to use App Store Connect to submit an app for distribution through the App Store.

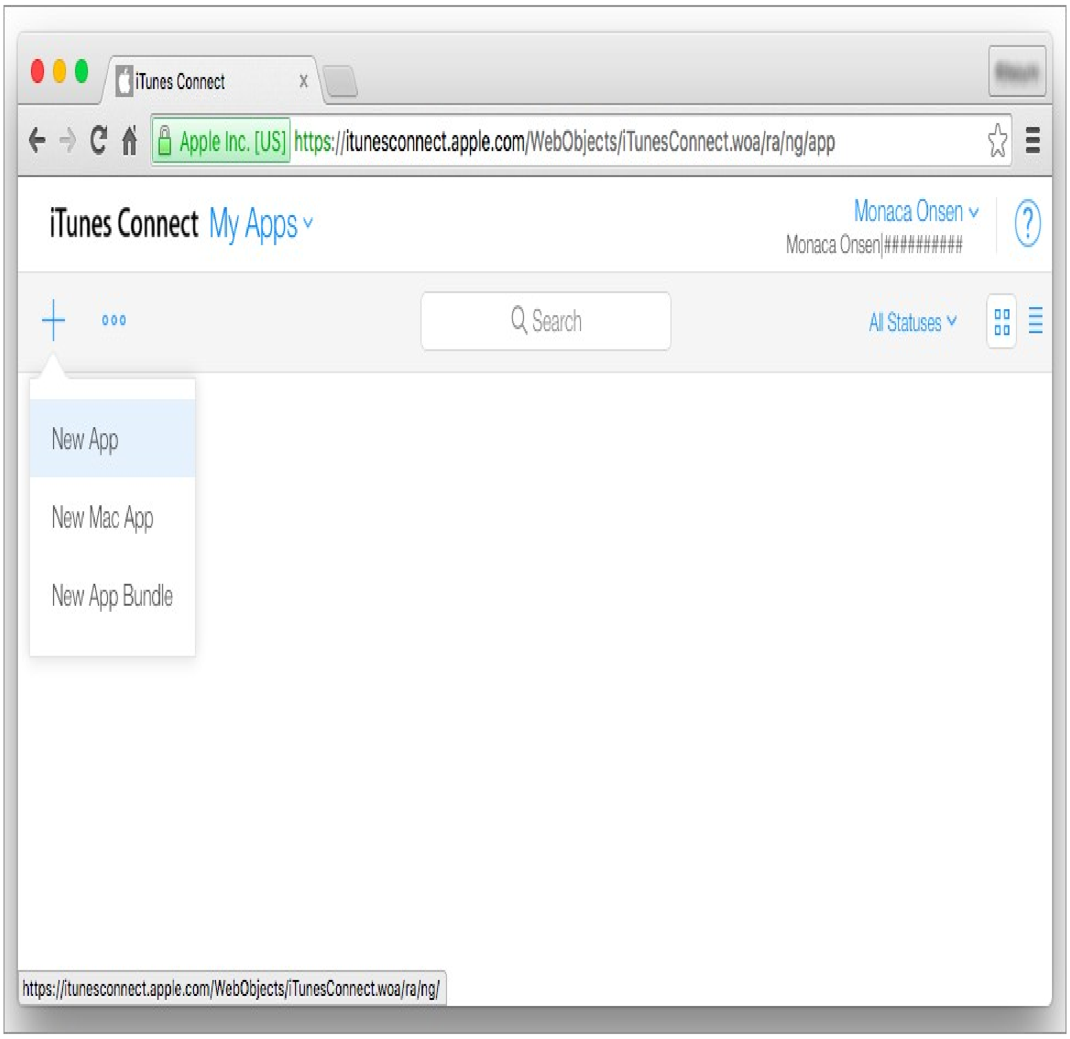
Prerequisite

You are required to have a Team Agent account under Apple Developer Program.

Registering the App

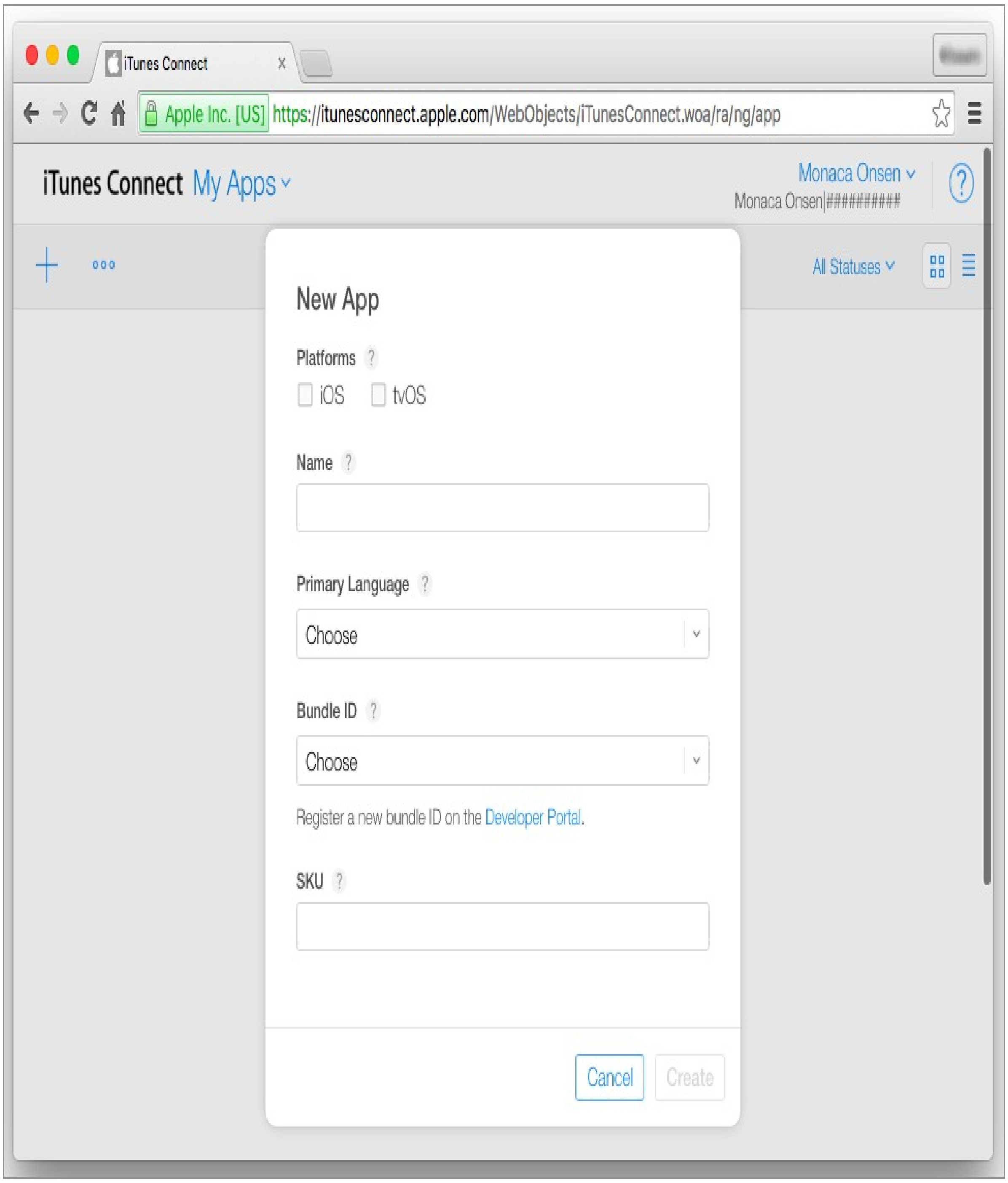
In order to register your app in App Store Connect, please do as follows:

1. Login to App Store Connect with your Team Agent account.
2. Go to `My Apps`.
3. Select **+** → **New App** .


iTunes Connect
https://itunesconnect.apple.com/WebObjects/iTunesConnect.woa/ra/ng/app
iTunes Connect My Apps
Monaca Onsen
Monaca Onsen|#########
Search
All Statuses
New App
New Mac App
New App Bundle
https://itunesconnect.apple.com/WebObjects/iTunesConnect.woa/ra/ng/

4. Enter the information about the app on the form that appears as seen below:

Data	Description
Platforms	Choose a platform for your app.
Name	Enter your app's name as it will appear on the App Store. It can't be longer than 255 characters.
Primary Language	Choose a default language for your app.
Bundle ID	Select a Bundle ID which can be an explicit App ID or a wildcard App ID. If it's a wildcard App ID, you also need to specify a bundle ID suffix. If it's an explicit App ID, it must exactly match the bundle identifier in your app.
Bundle ID Suffix	Once you select a Bundle ID, the input field for Bundle ID Suffix will be displayed. It's a string that is appended to the bundle ID property if the bundle ID is a wildcard App ID. The bundle ID and bundle ID suffix must form a bundle identifier that exactly matches the bundle identifier in your app.
SKU	Enter the code to identify the app. Although the code is up to your decision, it is necessary to create an ID that will identify the app in a unique way.


iTunes Connect

https://itunesconnect.apple.com/WebObjects/iTunesConnect.woa/ra/ng/app

iTunes Connect My Apps

Monaca Onsen

Monaca Onsen|#########

All Statuses

New App

Platforms

iOS tvOS

Name

Primary Language

Choose

Bundle ID

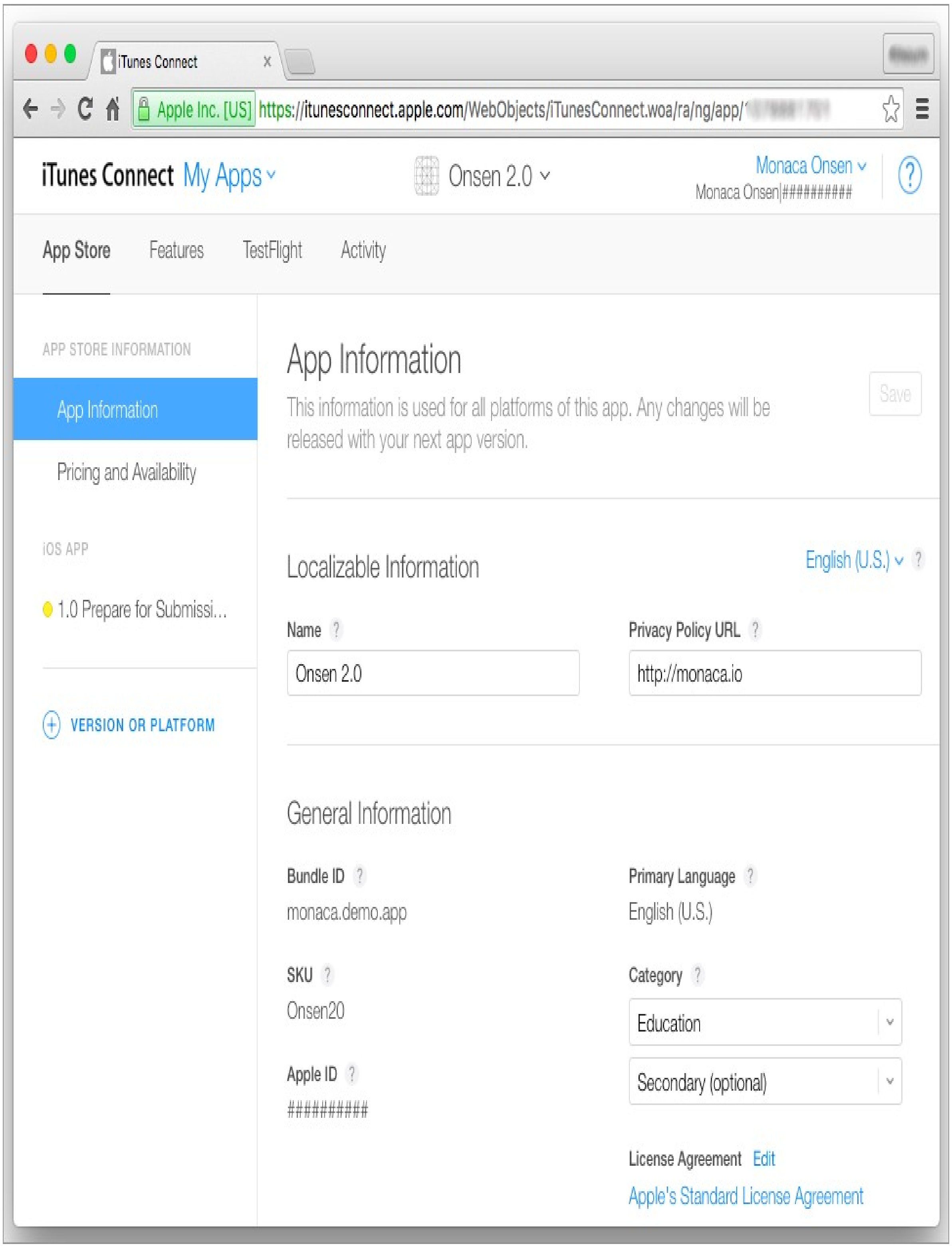
Choose

Register a new bundle ID on the Developer Portal.

SKU

Cancel Create

5. Click Create. Then, you will be redirected to the App Information page.
6. In the App Information page, choose a category for your app and click Save.



https://itunesconnect.apple.com/WebObjects/iTunesConnect.woa/ra/ng/app/[redacted]

Monaca Onsen|#########

App Information

This information is used for all platforms of this app. Any changes will be released with your next app version.

Localizable Information

English (U.S.)

Name: Onsen 2.0

Privacy Policy URL: http://monaca.io

General Information

Bundle ID: monaca.demo.app

Primary Language: English (U.S.)

SKU: Onsen20

Category: Education

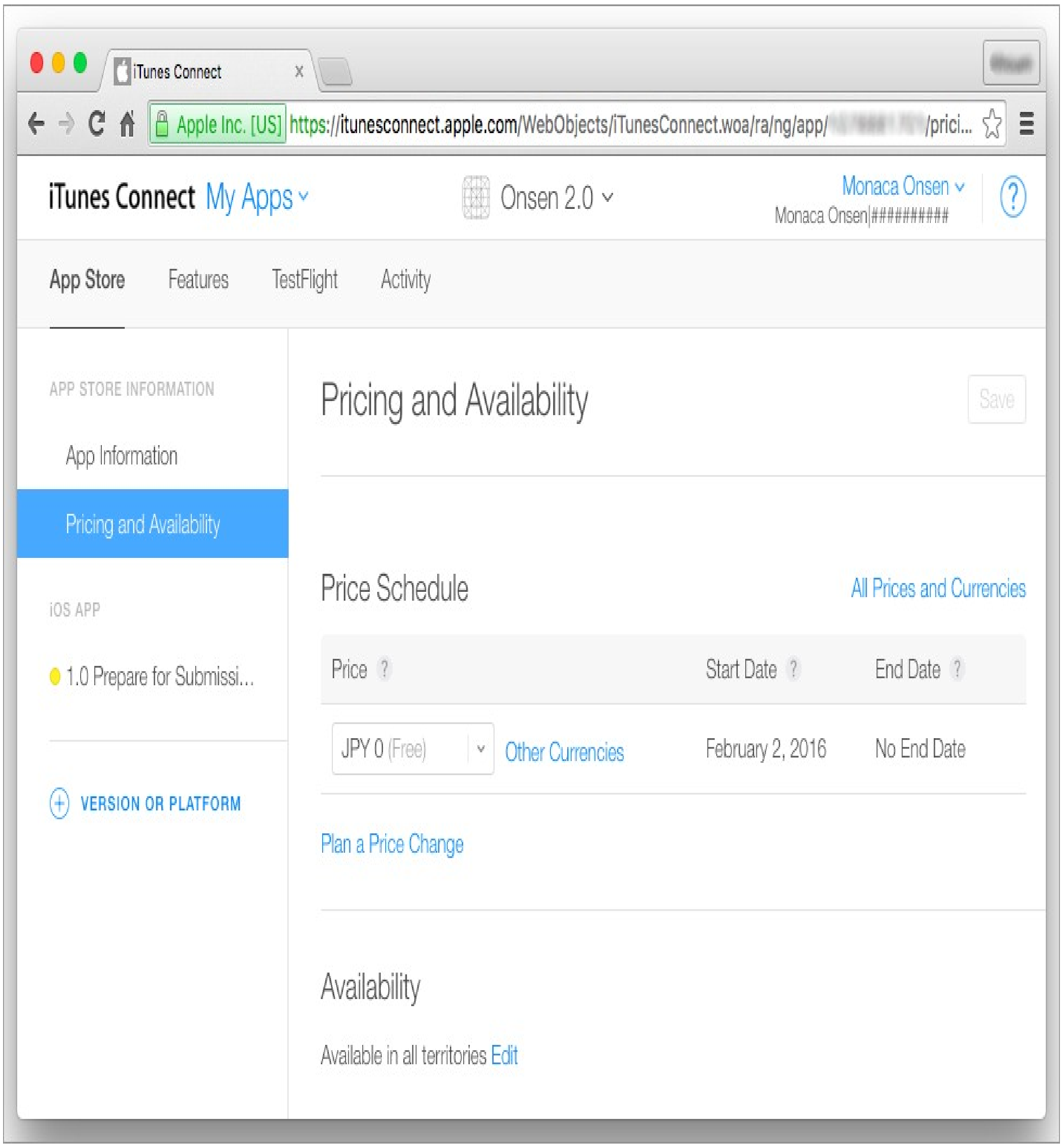
Secondary (optional)

Apple ID: #########

License Agreement Edit

Apple's Standard License Agreement

7. Go to `Pricing and Availability`. In this page, you will need to configure 3 main points such as price schedule, available areas for your app and volume purchase program. Then, click Save.



Pricing and Availability

Price Schedule

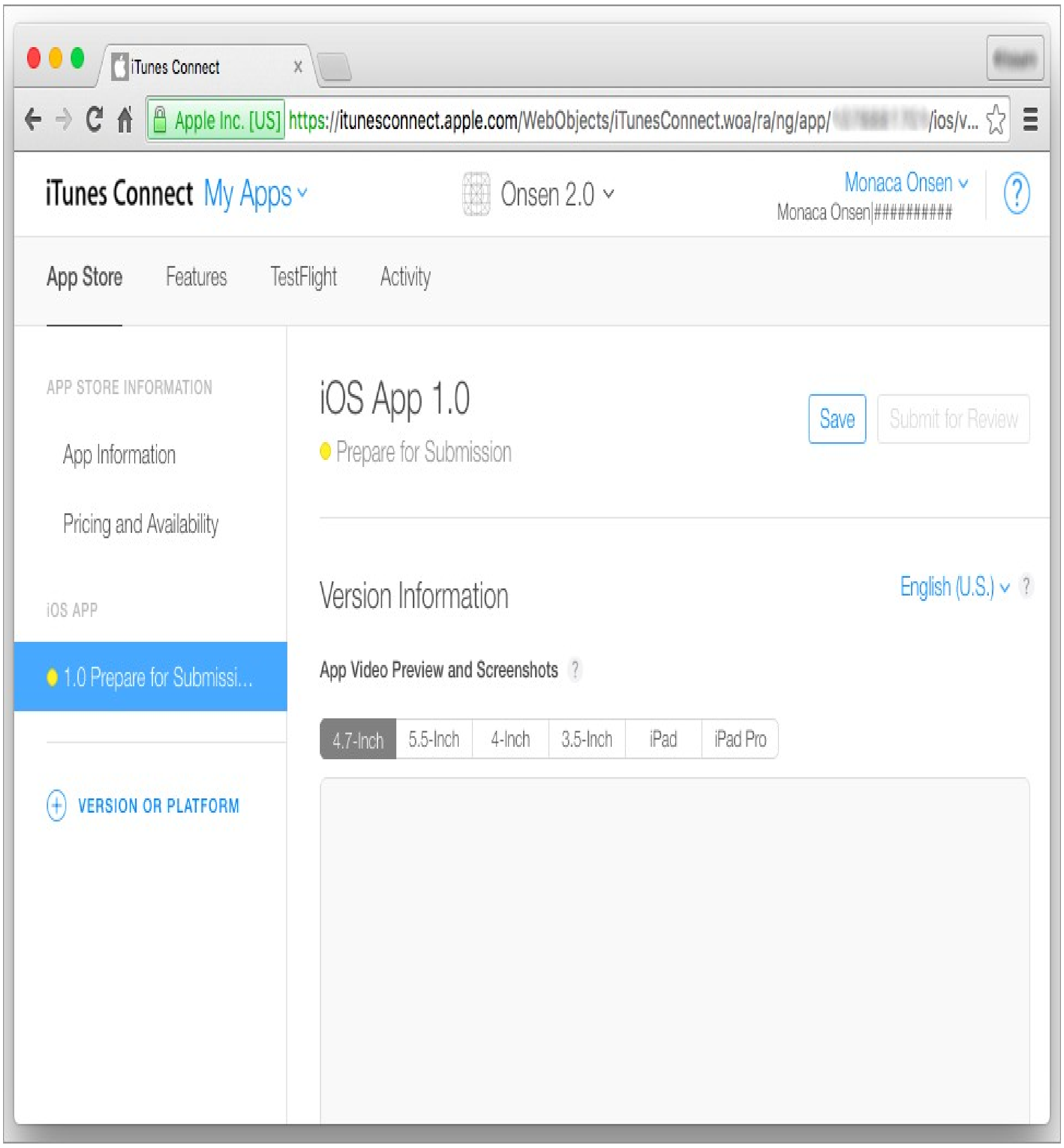
Price	Start Date	End Date
JPY 0 (Free)	February 2, 2016	No End Date

Availability

Available in all territories Edit

8. Go to `Prepare for Submission`. In this page, you will need to configure:

- App video preview and screenshots: at least one screenshot is required.
- Description
- Keywords
- Support URL
- App icon
- Copyright
- Contact information.



9. After the configuration, click Save.

Uploading the App

Monaca offers [iOS App Upload feature](#) which allows you to upload your app to App Store Connect right from Monaca IDE without using a Mac.

In addition to the upload function provided by Monaca, you can also upload apps to App Store Connect using [Transporter](#) or the `altool` included in Xcode.

altool usage example

```
xcrun altool --upload-app -f file -u username -p password
```

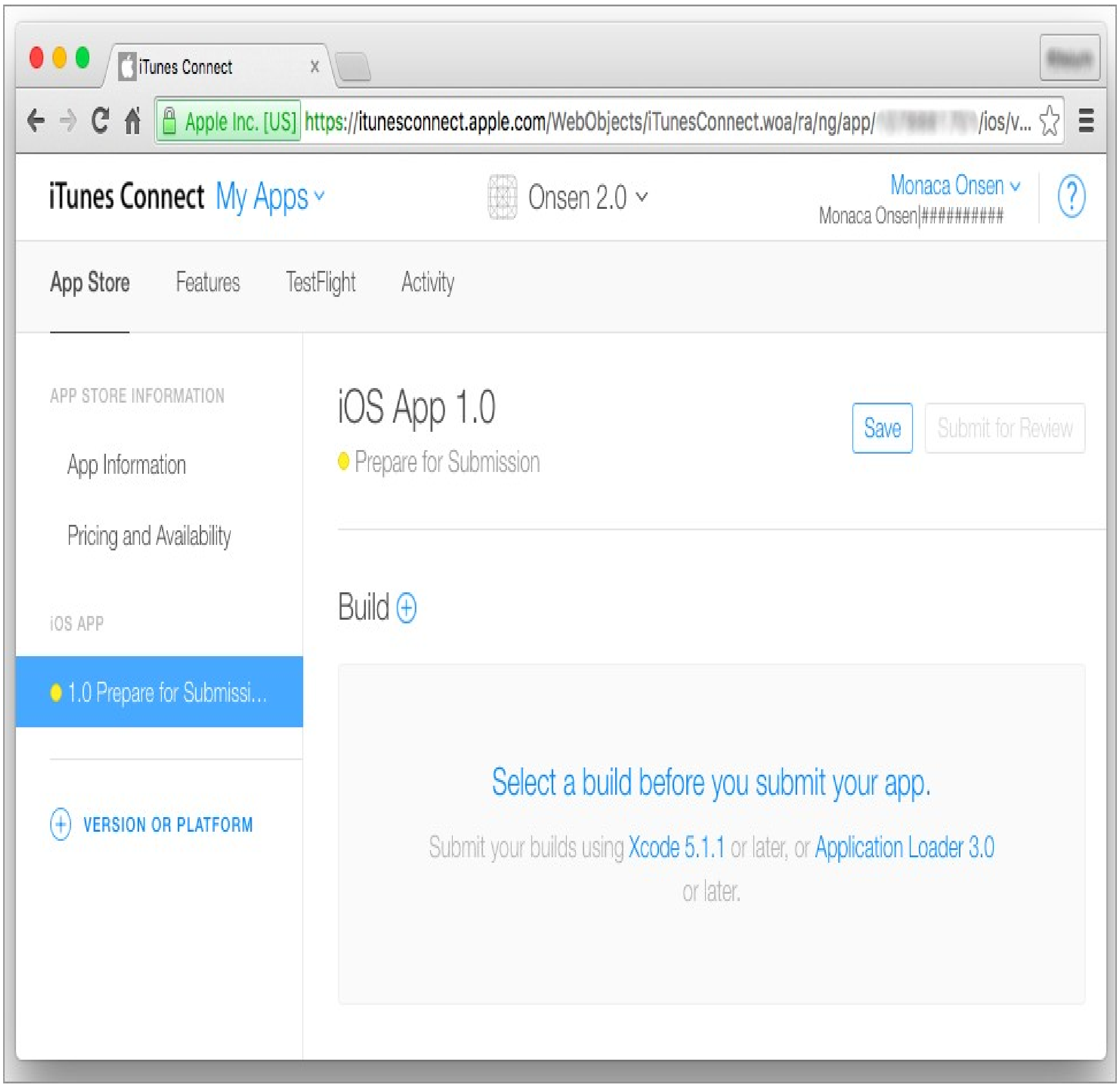
For information about altool, see [Using altool](#).

If you upload multiple versions of the app, please make sure that each app file has different version number. Otherwise, the upload will fail.

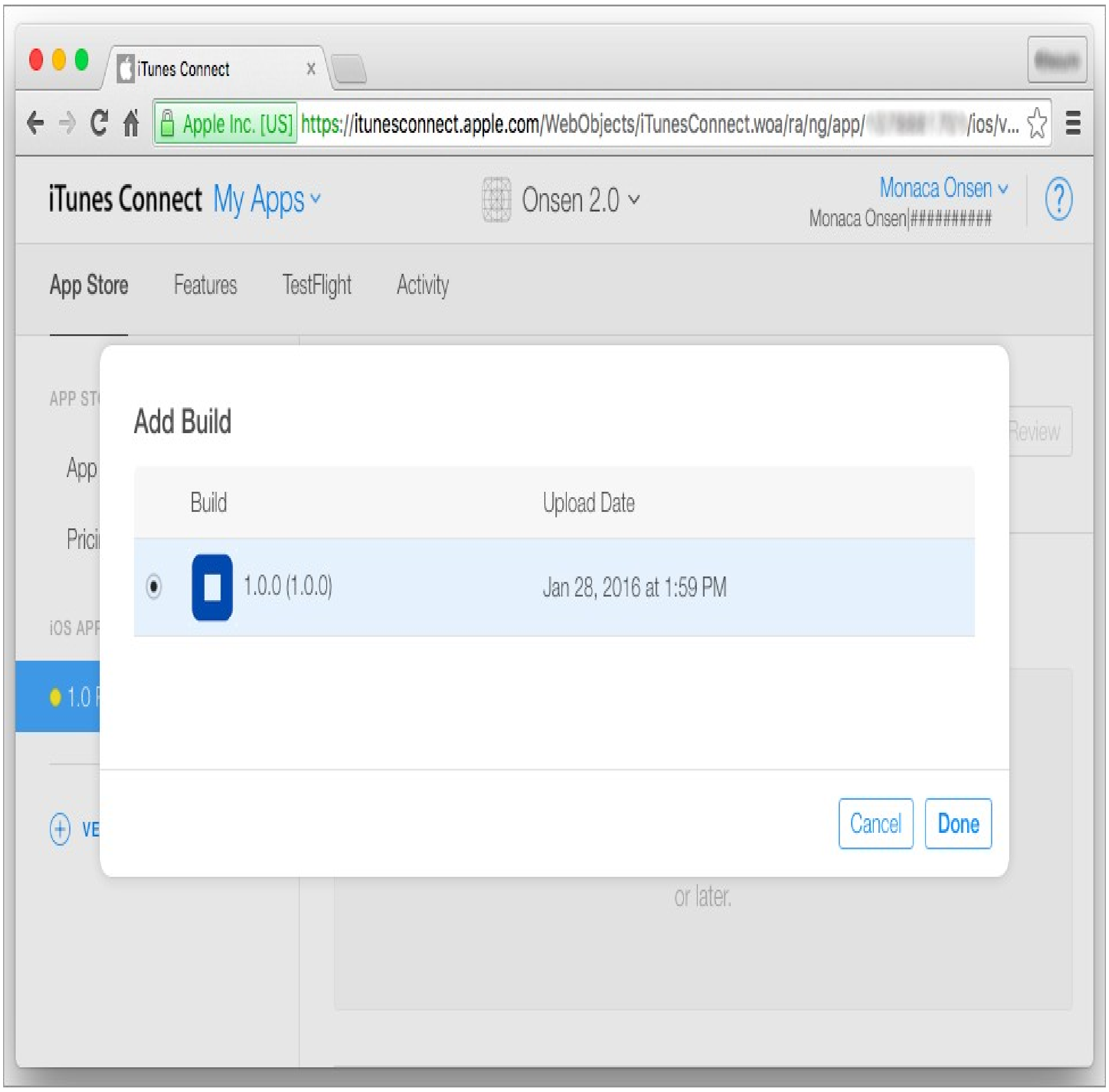
Selecting the Uploaded App

Once you successfully uploaded your app to App Store Connect, it can be selected to submit to the App Store. Please do as follows:

1. From App Store Connect, go to `Prepare for Submission`. Under `Build` section, click +.



2. Select your build and click Done.



Add Build

Build	Upload Date
● 1.0.0 (1.0.0)	Jan 28, 2016 at 1:59 PM

3. Click Save.

Submitting the App

Now that you have completed the necessary configurations, your app is ready to be submitted.

1. From App Store Connect, go to `Prepare for Submission`.
2. Click Submit for Review.

After you've successfully submitted the app, you will need to wait for the review from Apple. Apple will review your app and see if it is eligible or qualified to be in App Store. Usually, it takes two weeks for the review. Therefore, please wait patiently for it.

Publishing the App

When your app is accepted by Apple, it will be up in the App Store.

The large app icon, which was not necessary before, is now required for the submission. In fact, the required environment and information have changed in the submission procedure for the registration. If a registration/submission is not successful, please make sure you complete necessary configurations properly.

See Also:

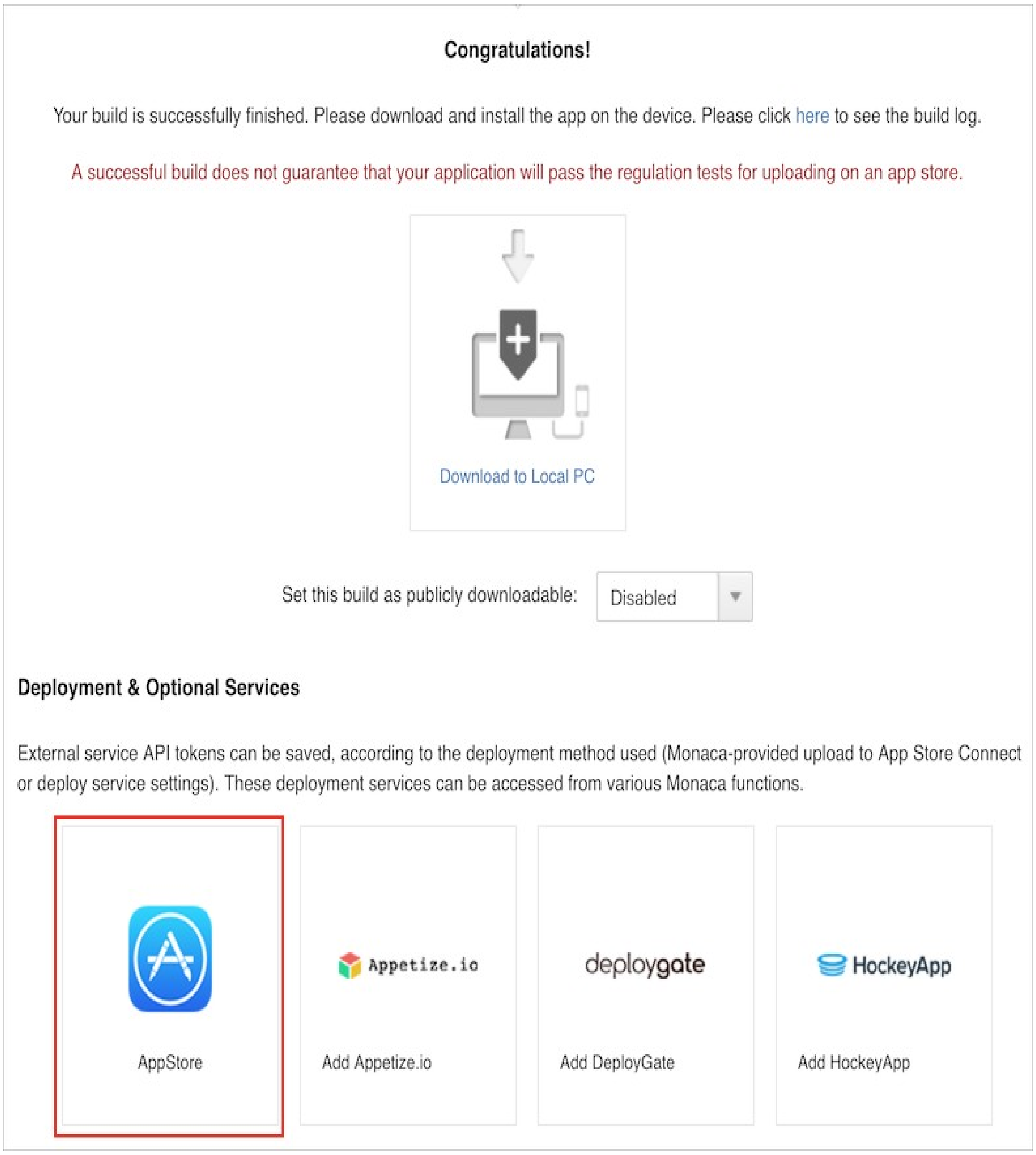
- Monaca Upload Feature
- Building an iOS App
- Google Play Distribution
- Non-market App Distribution
- Amazon Appstore Distribution

iOS App Upload Feature

In order to use iOS App Upload feature, you are required to subscribe to a valid plan. Please refer to Monaca Subscription Plans .
For shared projects, only project owner can use iOS App Upload feature.

Monaca allows you to upload your app to App Store Connect right from Monaca IDE. In order to do this, please do as follows:

1. Create a release build version of your app through Monaca. Please refer to Building an iOS App.
2. When you successfully build a release version of your app, the following screen will appear. Click on the `AppStore` icon as shown in the screen below.


Congratulations!

Your build is successfully finished. Please download and install the app on the device. Please click here to see the build log.

A successful build does not guarantee that your application will pass the regulation tests for uploading on an app store.

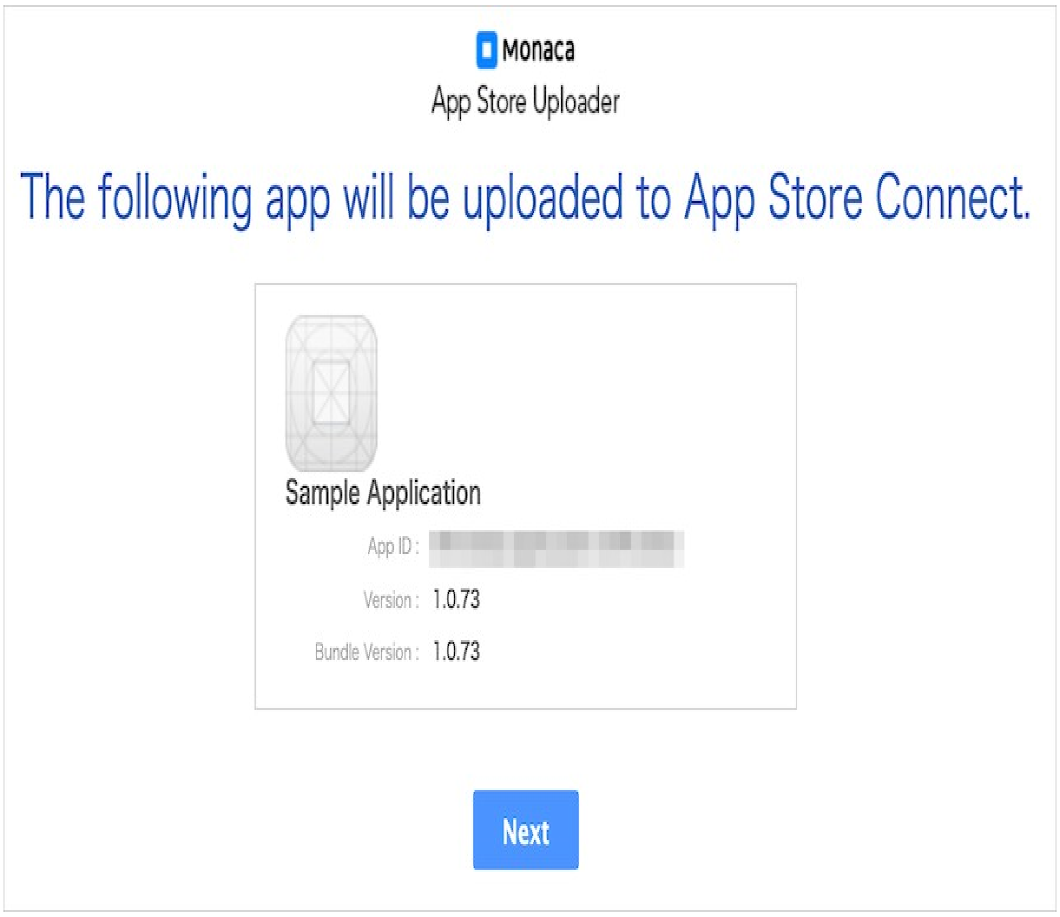
Download to Local PC

Set this build as publicly downloadable: Disabled

Deployment & Optional Services

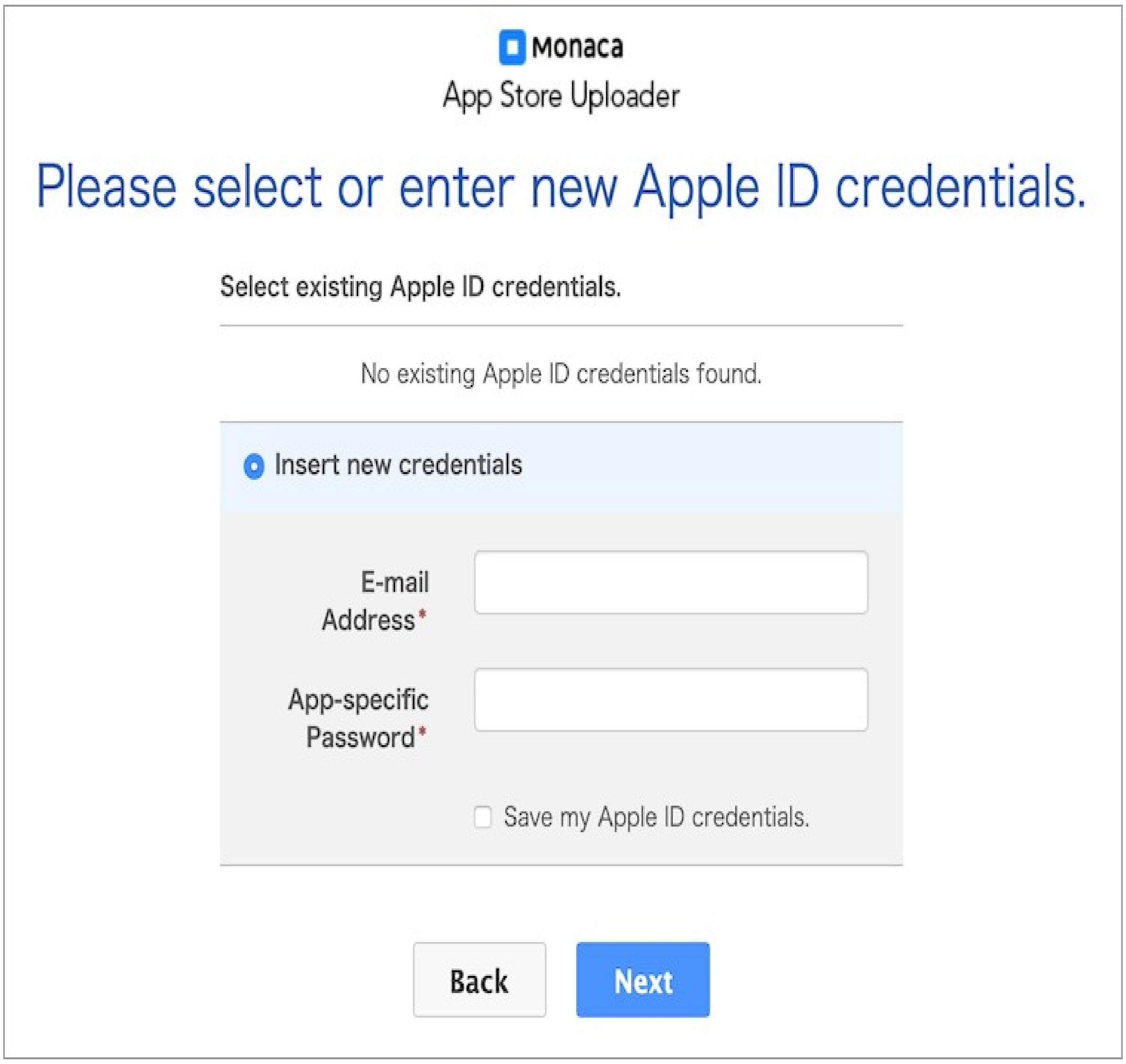
External service API tokens can be saved, according to the deployment method used (Monaca-provided upload to App Store Connect or deploy service settings). These deployment services can be accessed from various Monaca functions.

3. Then, the App Upload window will appear. Click Next.



4. Fill in a valid Apple account information. Click Next.

   Use `app-specific passwords` to the password field.
   For details on how to obtain the password for App, please refer here .


Monaca
App Store Uploader

Please select or enter new Apple ID credentials.

Select existing Apple ID credentials.

No existing Apple ID credentials found.

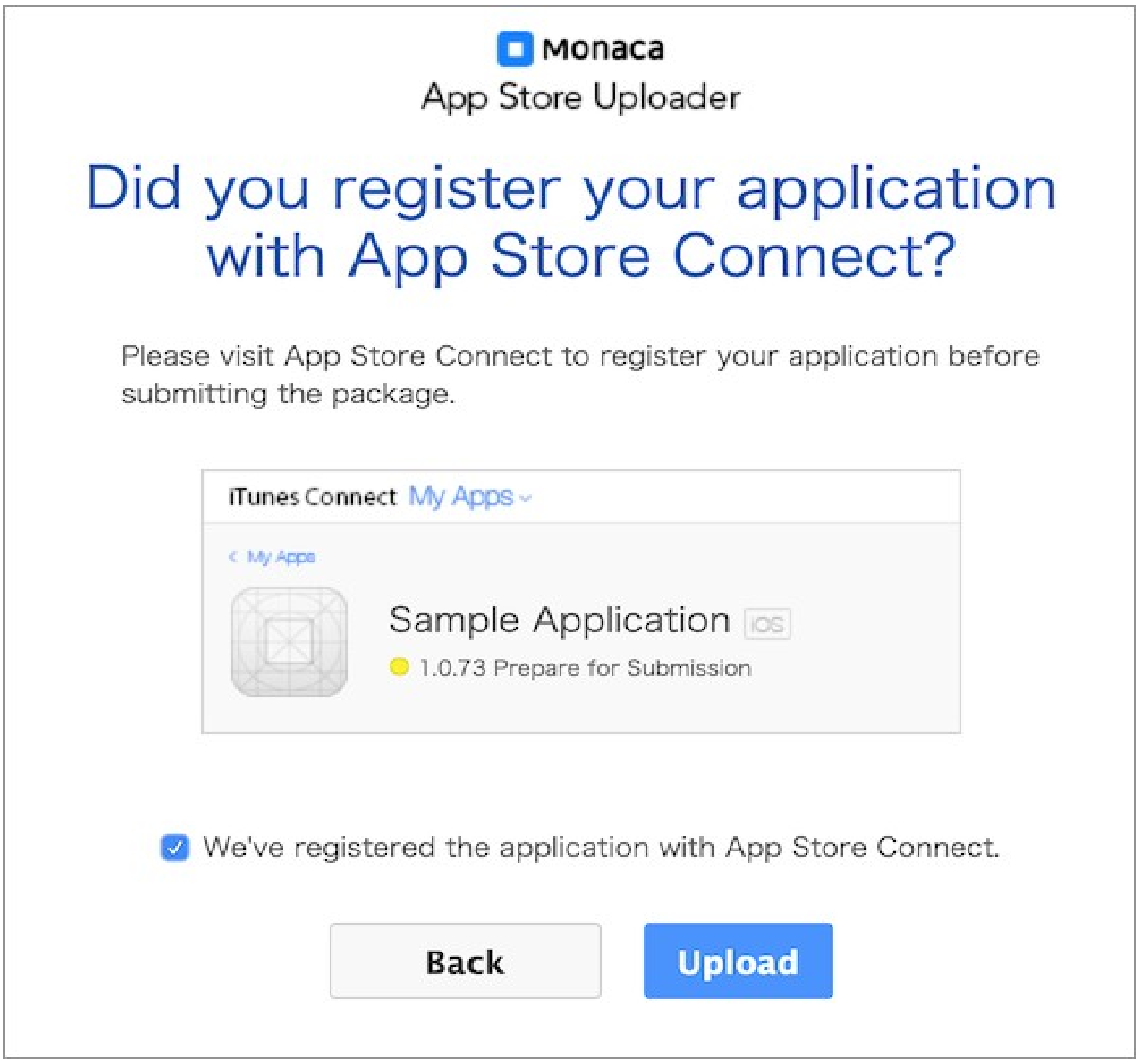
● Insert new credentials

E-mail Address*

App-specific Password*

☐ Save my Apple ID credentials.

5. Make sure you’ve registered this app with App Store Connect before uploading your app. Then, tick `We've registered the application with App Store Connect.`. Click Upload.



Monaca

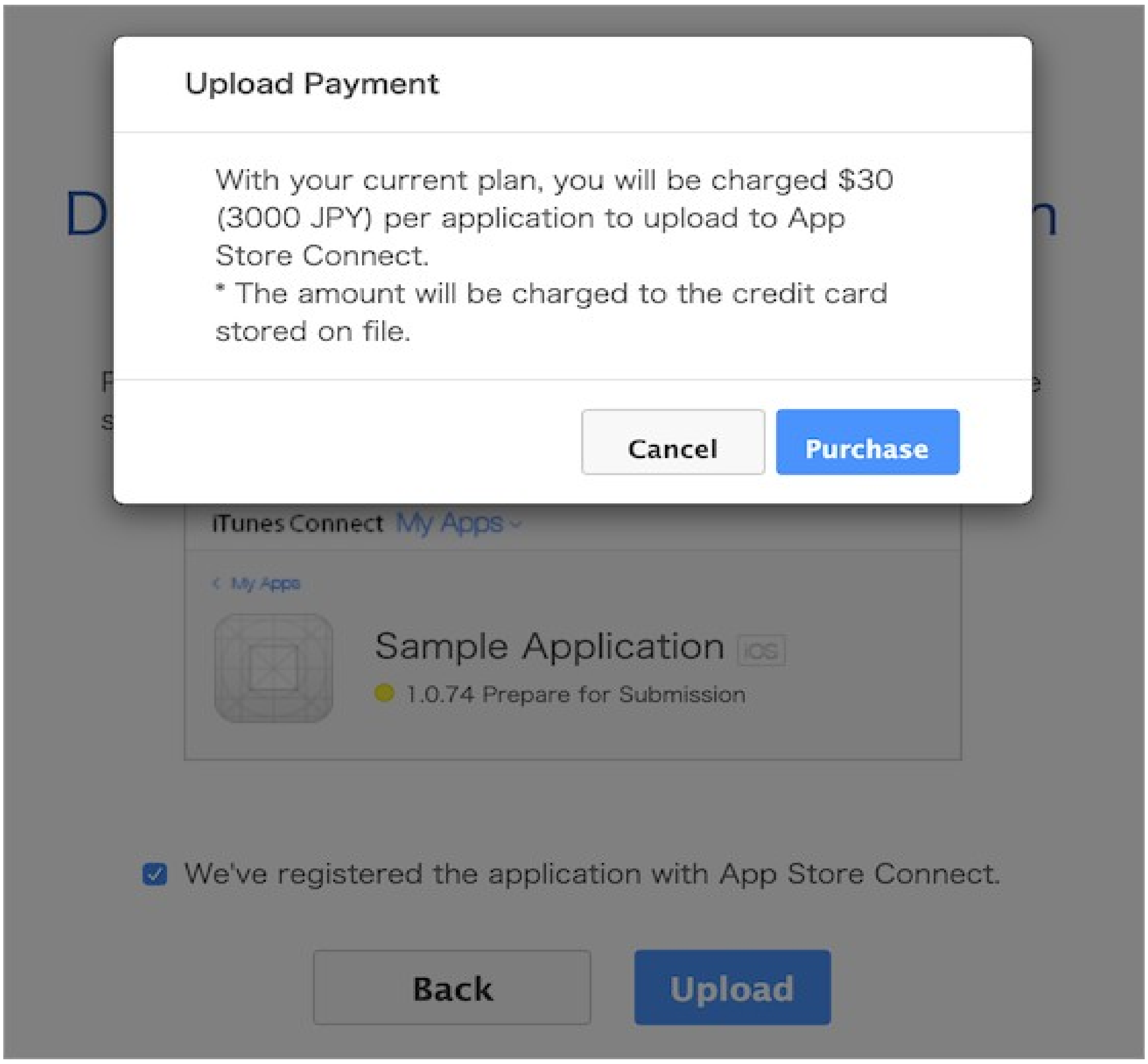
App Store Uploader

Did you register your application with App Store Connect?

Please visit App Store Connect to register your application before submitting the package.

✓ We've registered the application with App Store Connect.

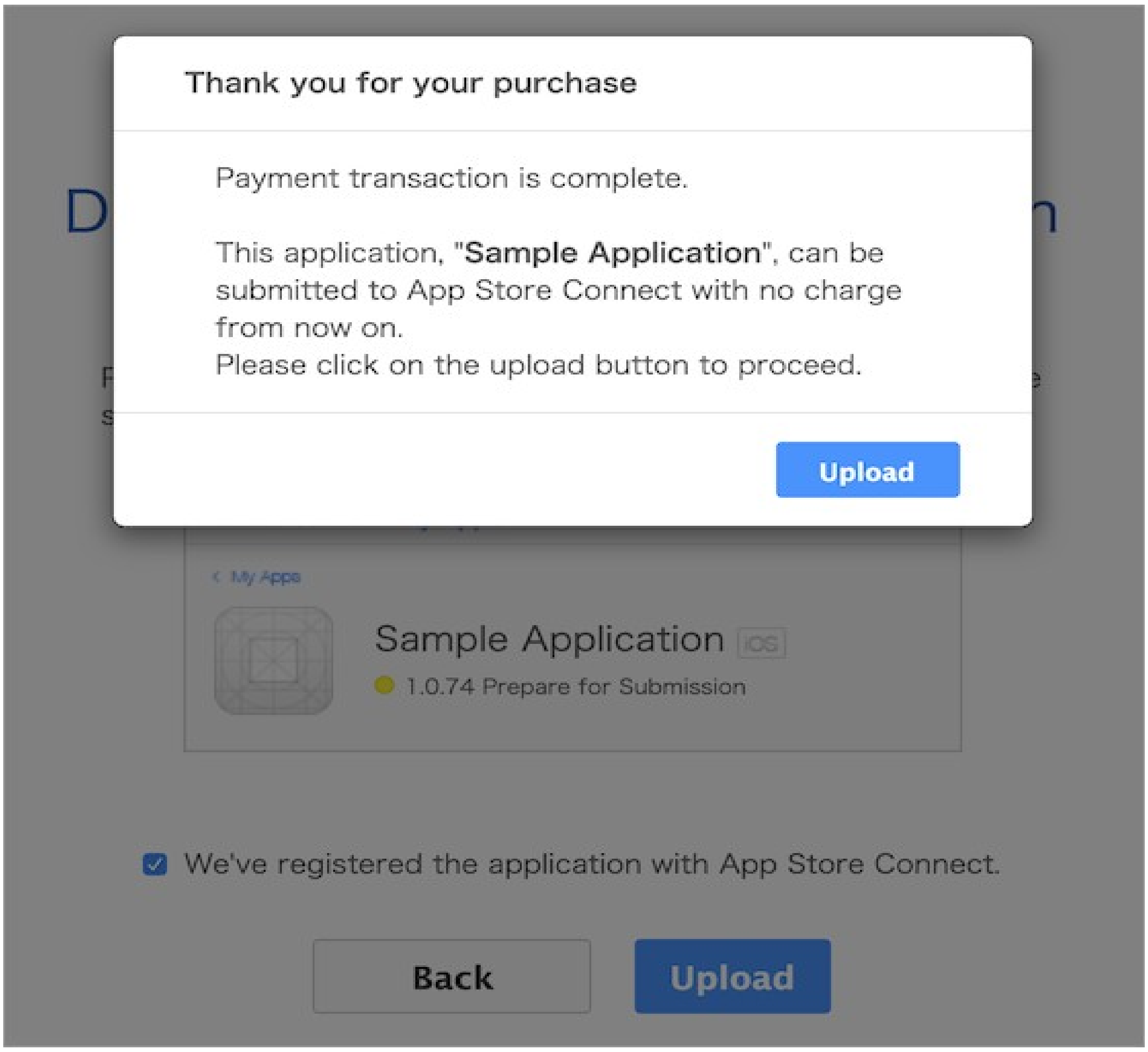
6. The following screen is displayed for a plan that requires billing. Click Purchase.


Upload Payment

With your current plan, you will be charged $30 (3000 JPY) per application to upload to App Store Connect.
* The amount will be charged to the credit card stored on file.

Cancel
Purchase
iTunes Connect My Apps
< My Apps
Sample Application iOS
1.0.74 Prepare for Submission
We've registered the application with App Store Connect.
Back
Upload

7. The following screen will be displayed when the billing process of [6] is completed. Click Upload.


Thank you for your purchase

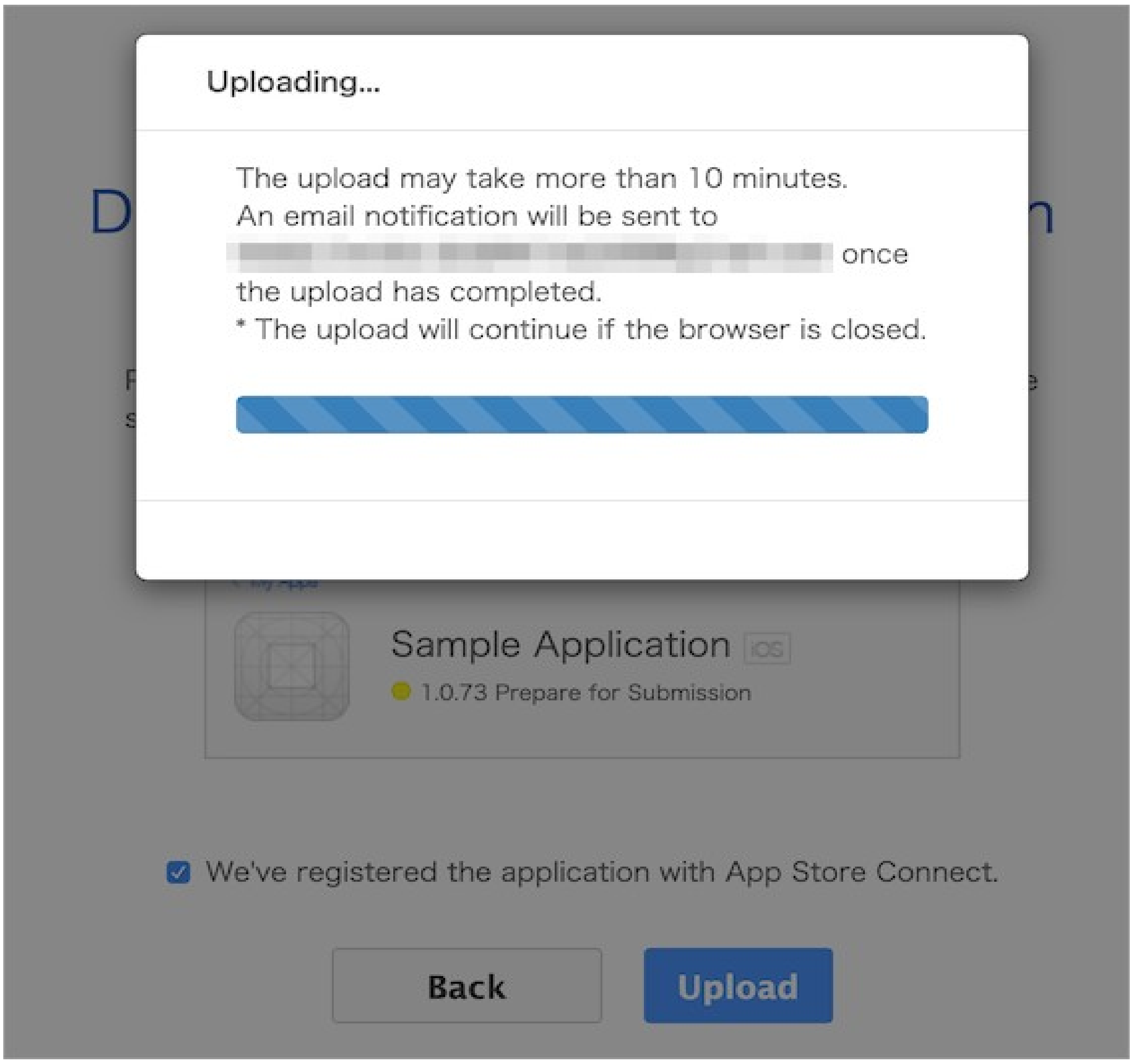
Payment transaction is complete.

This application, "**Sample Application**", can be submitted to App Store Connect with no charge from now on.
Please click on the upload button to proceed.

Upload
< My Apps
Sample Application iOS
1.0.74 Prepare for Submission
We've registered the application with App Store Connect.
Back
Upload

8. The uploading will start. Please wait.

   If you upload multiple versions of the app, please make sure that each app file has different version number. Otherwise, the upload will fail.

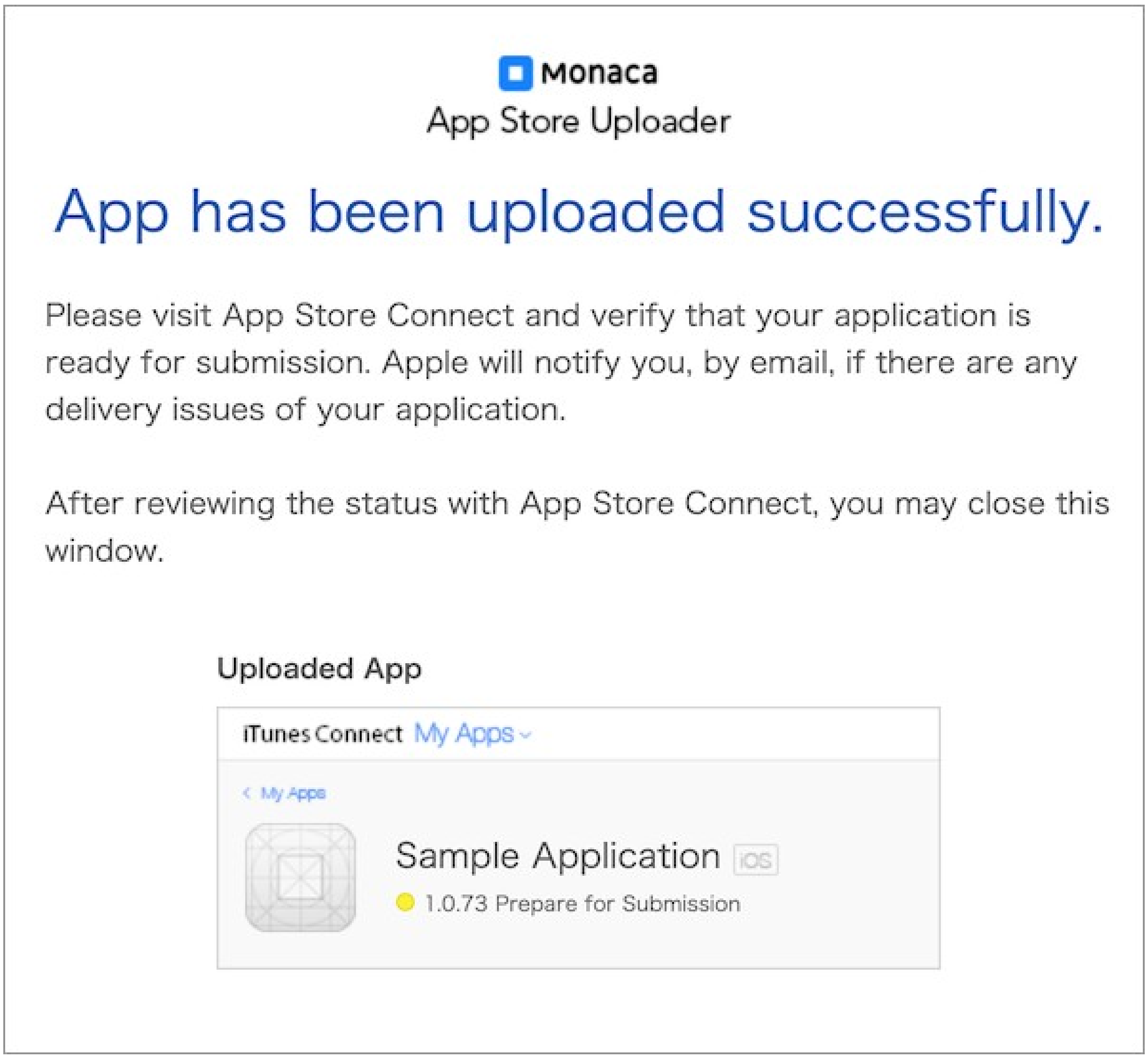

Uploading...

The upload may take more than 10 minutes.
An email notification will be sent to [redacted] once the upload has completed.
* The upload will continue if the browser is closed.

Sample Application iOS
1.0.73 Prepare for Submission

✓ We've registered the application with App Store Connect.

9. If your upload is successful, the following screen will appear. It may take sometimes until the app shows up in the App Store Connect.


Monaca
App Store Uploader

App has been uploaded successfully.

Please visit App Store Connect and verify that your application is ready for submission. Apple will notify you, by email, if there are any delivery issues of your application.

After reviewing the status with App Store Connect, you may close this window.

Uploaded App

Sometimes Monaca is successfully uploaded your app to App Store Connect but Apple may find error(s) on their end and report to you via email. If this happens, please see the report and fix the error(s) appropriately. Then, re-upload your app.

10. Now that you successfully uploaded your app, you can start selecting it in App Store Connect. Please refer to Selecting the Uploaded App.

If you use activation code to upgrade your account, you can use Monaca upload feature in case your plan supports this feature. Please contact us for more details.

See Also:

- App Store Distribution
- App Store Connect Guide

Google Play Distribution

Prerequisite

In order to distribute your apps through Google Play, you must register as a *Google Play Developer Console* account. The registration fee is `$25` (one-time only). It will take `48` hours for your *Google Play Developer Console* registration to be fully processed. To register, please go here.

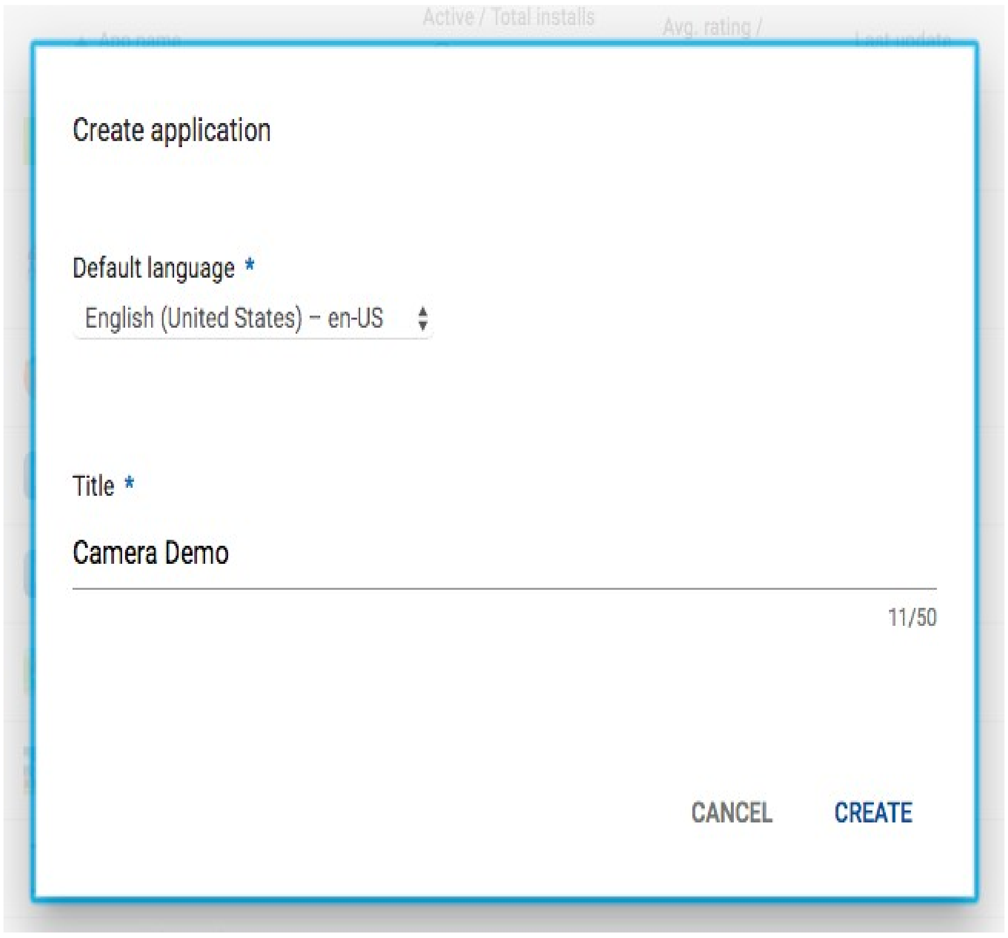
If you want to sell your apps, you will also need to register as a Google Checkout merchant as well. To register, please visit Link a Google Play Developer account to your payments profile .

Create a Release Build of the App

Create and download a release build version of your app through Monaca. Please refer to Building for Android to build a release build of the app. Then, download the built app (APK file).

Register the Apps in Google Play

1. Go to Google Play Developer Console and login with a valid Google Developer account.
2. Click on CREATE APPLICATION button.
3. Choose a default language and enter a title for your app. Then, click CREATE. Then, you will be forwarded to `Store listing` page.


Create application

Default language *
English (United States) – en-US

Title *
Camera Demo
11/50

4. In this page, you will need to fill in the following necessary information and click SAVE DRAFT.

Data	**Description**
Short description	Description of your app shown in Google Play. It can be up to `80` characters.
Full description	Description of your app shown in Google Play. It can be up to `4000` characters.
Screenshots	At least 2 screenshots are required but you can upload up to 8 screenshots per type.
Hi-res icon	You are required to upload a high-resolution icon (`512x512` PNG file) for your app.
Feature Graphic	You are required to upload a feature graphic (`1024x500` PNG file) for your app.
Application type	It can be `Applications` or `Games`. They are the major application types in Google Play.
Category	Select a category for your app.
Content rating	Select a content rating of your app as appropriate.
Contact details	You must have at least one support channel for your app. The support channels can be website, email and phone. This information can be viewed by users from Google Play.
Privacy Policy	Input your privacy policy URL if you have one. Otherwise, please tick `Not submitting a privacy policy URL at this time.`.

5. Go to `Content rating` section and click CONTINUE.
6. Fill in your email address and select your app category. Then, you will be asked to answer some questions related to the selected app category.
7. Click on SAVE QUESTIONNAIRE and CALCULATE RATING.
8. Confirm your rating information and click APPLY RATING.
9. Go to `Pricing & distribution` section. In this page, you will be asked to complete various questions related the price and availability of your app. Until this point, you are successfully complete the necessary information for your app to be ready for a release.

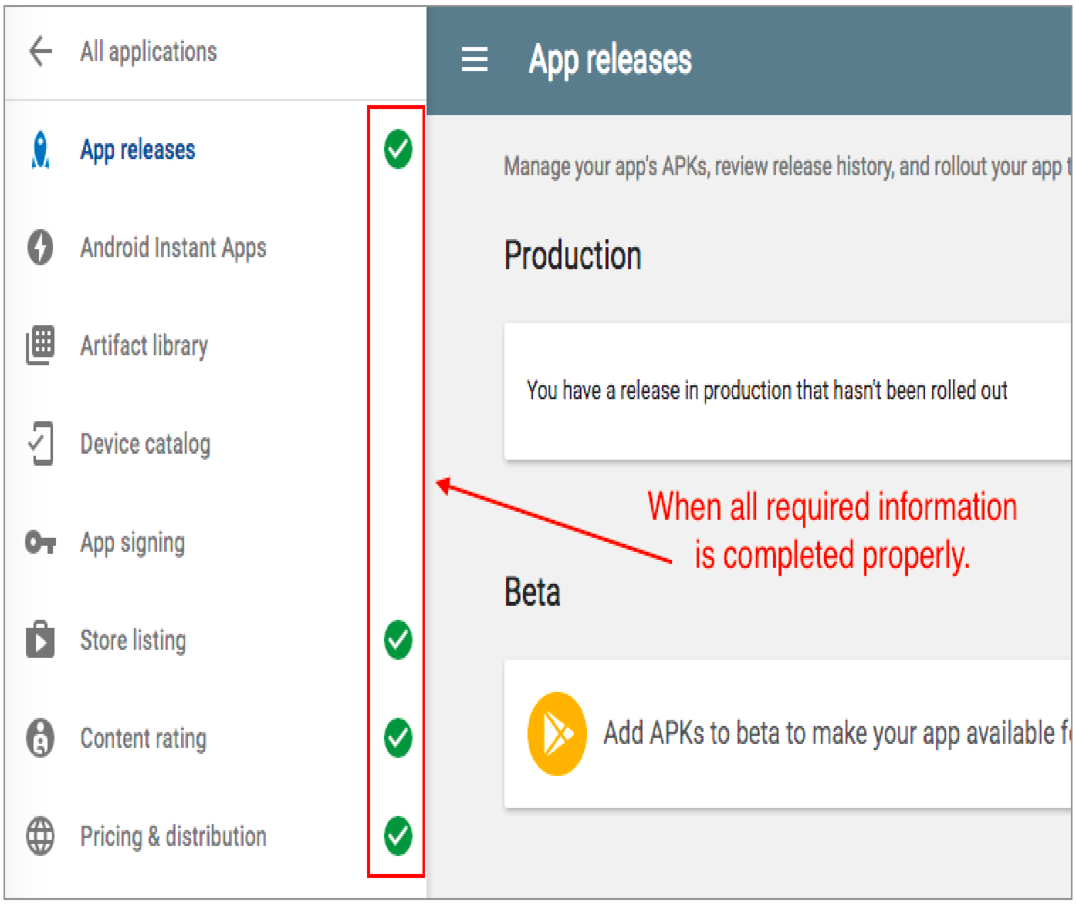
Release the App

In this section, we will start uploading the APK file and finalize the release information of the app before submitting to the Play Store.

1. Go to `App releases` section. In this page, you can upload your APK files for testing (Beta and Alpha) and production.
2. Under the `Production section`, click on MANAGE PRODUCTION. Then, click on CREATE RELEASE.
3. Then, you will need to:
    - complete the google Play App Signing option
    - upload the APK file

- configure the Release name (release version)
- write about what's new about this release

4. Click SAVE and REVIEW.
5. Review your app's release information one more time and click on START ROLLOUT TO PRODUCTION button to submit your app to the Play Store. Please note that you can't submit your app if you don't configure your app properly. In other words, the START ROLLOUT TO PRODUCTION button is disable unless the required information is completed properly.



Manage your app's APKs, review release history, and rollout your app t

You have a release in production that hasn't been rolled out

Add APKs to beta to make your app available f

See Also:

- Building for Android
- App Store Distribution
- Non-market App Distribution

Building for Android

Types of Build

In Monaca, Android app has two types of build: debug version and release version. The differences between these types of build are as follows:

Types of Build	Description	Installation
Debug Build	An unsigned package which cannot be distributed in the market	• QR Code • Network Install • Sideloading
Release Build	A signed package with the developer's code sign which can be distributed in the market	• Sideloading • Google Play Store and other eligible markets

Sideloading typically refers to media file transfer to a mobile device via USB, Bluetooth, WiFi or by writing to a memory card for insertion into the mobile device. When referring to Android apps, "sideloading" typically means installing an application package in APK format onto an Android device without going through the market.

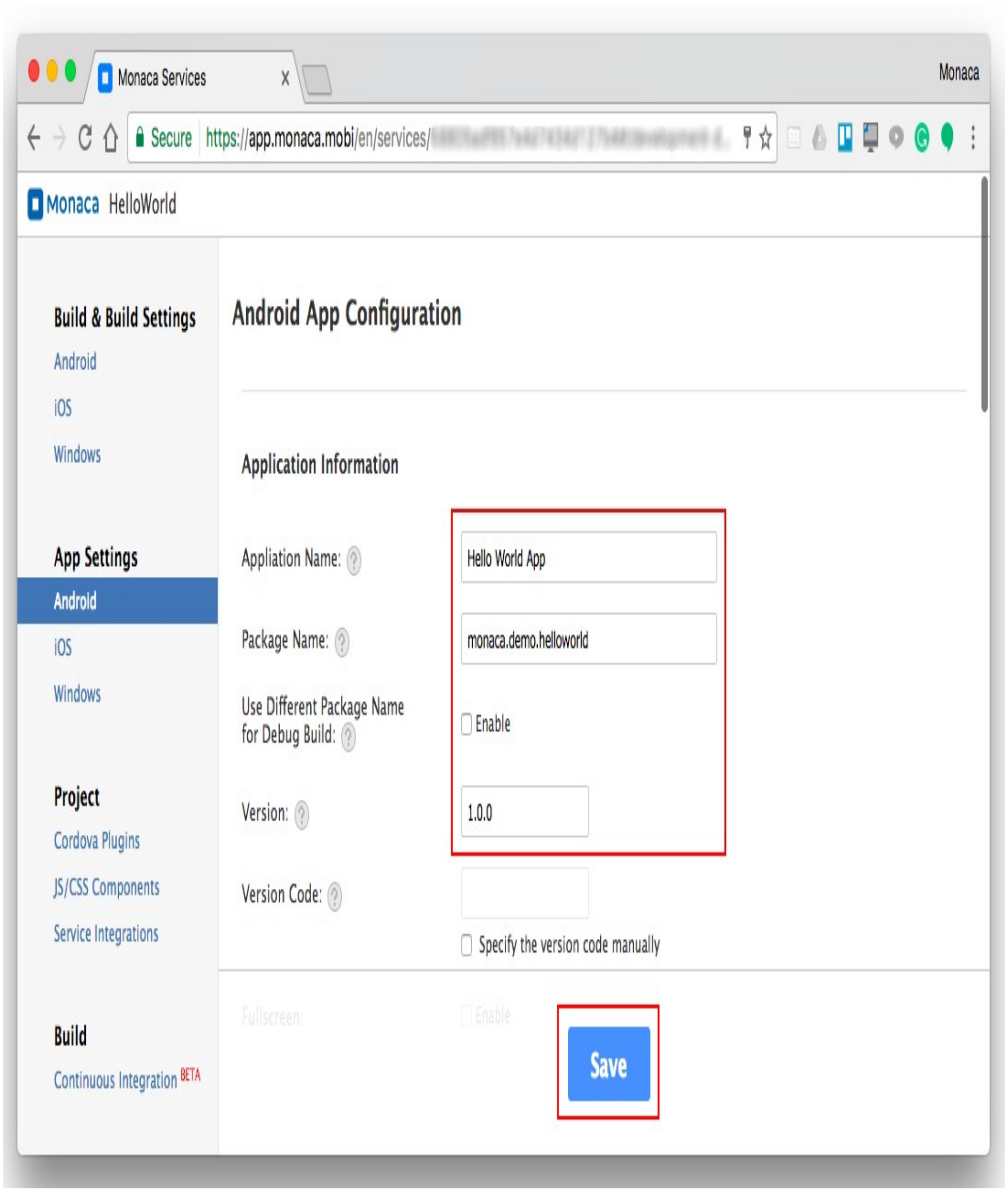
Step 1: Configure Android App

1. From the Monaca Cloud IDE menu, go to **Configure → App Settings for Android** .
2. Fill in the necessary information of your app:

    - General settings:

Application Name	A name representing your app publicly such as in the Market
Package Name	A unique name which will be used when uploading to the Android Market. It is recommended to use reverse-domain style (for example, io.monaca.app_name) for App ID. Only alphanumeric characters, periods (at least one period must be used) and underscore are allowed. Each segment should be separated by a period and started with an alphabetic character.
Use Different Package Name for Debug Build	If enable, the package name of the release-built and debug-built apps are different. In other words, the package name of debug-built app will have `.debug` extension, and the one for project debugger will have `.debugger` extension. However, this option is disable by default because it made some plugins impossible to be debugged due to the fact that they are tied to exact package names (eg. in-app purchase).
Version	The version number of your app. A version number consist of only number seperated by dots (for example, 1.0.0).
Version Code	An internal version number of your app, relative to other versions. The value must be integer, so that the applications can programmatically evaluate it for an upgrade.
Fullscreen	This option is only available with the Cordova 3.5 and later.

If enable, your app will be run in a fullscreen mode which hide the status bar.



Android App Configuration

Application Information

Appliation Name:	Hello World App
Package Name:	monaca.demo.helloworld
Use Different Package Name for Debug Build:	☐ Enable
Version:	1.0.0
Version Code:	☐ Specify the version code manually
Fullscreen:	☐ Enable

Save

- Misc: various settings regarding your Android app such as:

Allowed URL	*	Specify URL(s) which can be accessed from your app. If set to *, you can access all domains from your app.
Keep Running	Enable	Enable this if you want Cordova to keep running in the background.
Disallow Overscroll	Enable	Enable this if you want to disable the glow when a user scrolls beyond the edge of the webview.
Screen Orientation	Default	You can also set the device's screen orientation when running your app as Landscape or Portrait.

After finishing the configurations, click Save.

Currently, when you update either iOS's App ID or Android's Package Name, both of them will change. In other words, they are configured to be the same. However, it is possible to make them different. Please refer to How to make iOS’s App ID and Android’s Package Name differently .

3. Set of adaptive icons

   If you are using a monaca project with cordova 9.0 or later, you can set an adaptive icon for android 8.0 or later.

   To set an adaptive icon, upload an image file under the `res` folder and add a `background` attribute and a `foreground` attribute to the `icon` tag of config.xml. If the previous src attribute is set, the value of the src attribute is not used.

   If the os version does not support adaptive icons, the image set in the `foreground` attribute will be displayed. If the previous `src` attribute is set, the value of the src attribute takes precedence.

   The following is an example of placing an image under `/res/android/icon/`.

```
<platform name="android">
    <icon density="ldpi" background="/res/android/icon/ldpi-
background.png" foreground="/res/android/icon/ldpi-foreground.png"/>
    <icon density="mdpi" background="/res/android/icon/mdpi-
background.png" foreground="/res/android/icon/mdpi-foreground.png"/>
    <icon density="hdpi" background="/res/android/icon/hdpi-
background.png" foreground="/res/android/icon/hdpi-foreground.png"/>
    <icon density="xhdpi" background="/res/android/icon/xhdpi-
background.png" foreground="/res/android/icon/xhdpi-foreground.png"/>
    <icon density="xxhdpi" background="/res/android/icon/xxhdpi-
background.png" foreground="/res/android/icon/xxhdpi-foreground.png"/>
    <icon density="xxxhdpi" background="/res/android/icon/xxxhdpi-
background.png" foreground="/res/android/icon/xxxhdpi-foreground.png"/>
</platform>
```

   The following is an example of setting the src attribute.

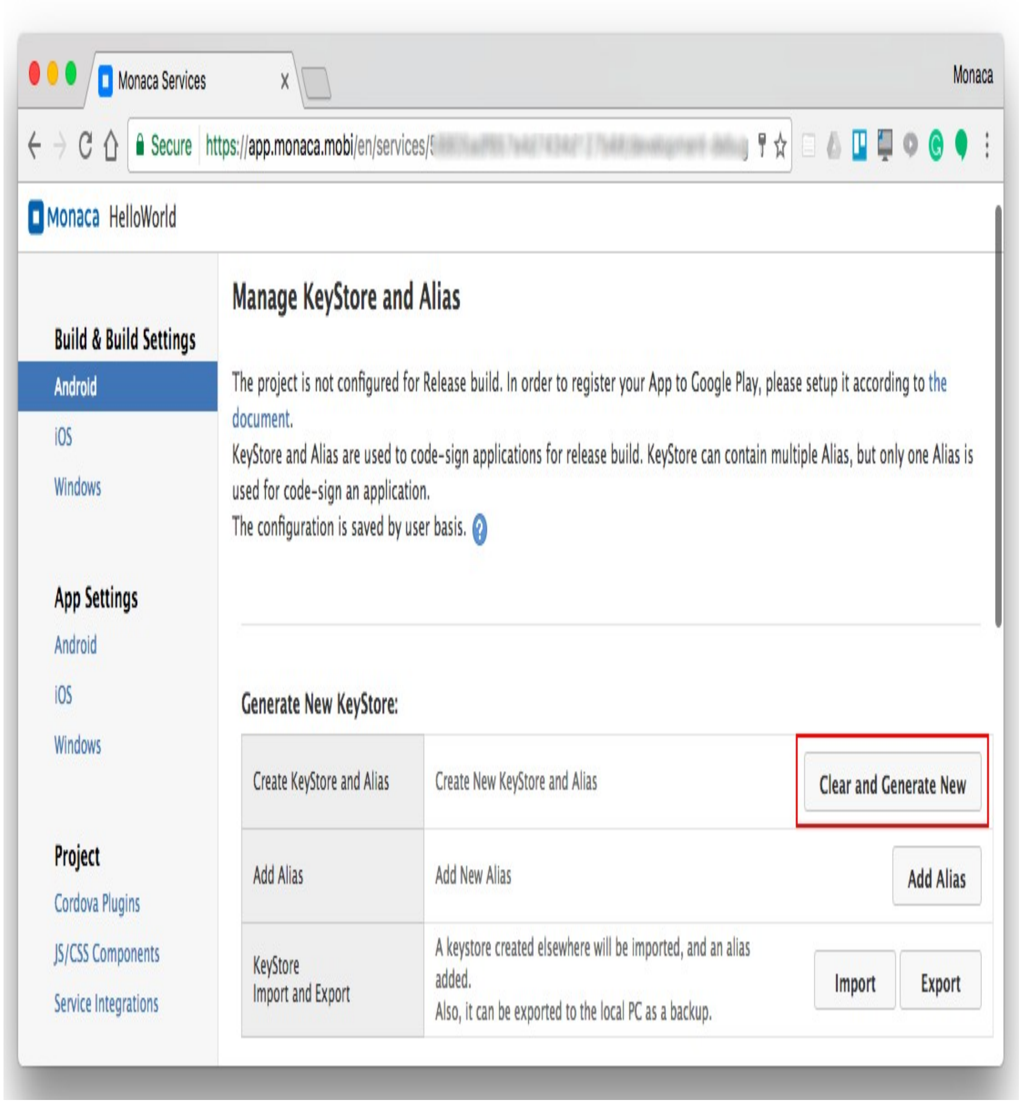
```
<platform name="android">
    <icon src="/res/android/icon/ldpi.png" density="ldpi"
background="/res/android/icon/ldpi-background.png"
foreground="/res/android/icon/ldpi-foreground.png"/>
    <icon src="/res/android/icon/mdpi.png" density="mdpi"
background="/res/android/icon/mdpi-background.png"
foreground="/res/android/icon/mdpi-foreground.png"/>
    <icon src="/res/android/icon/hdpi.png" density="hdpi"
background="/res/android/icon/hdpi-background.png"
foreground="/res/android/icon/hdpi-foreground.png"/>
    <icon src="/res/android/icon/xhdpi.png" density="xhdpi"
background="/res/android/icon/xhdpi-background.png"
foreground="/res/android/icon/xhdpi-foreground.png"/>
    <icon src="/res/android/icon/xxhdpi.png" density="xxhdpi"
background="/res/android/icon/xxhdpi-background.png"
foreground="/res/android/icon/xxhdpi-foreground.png"/>
    <icon src="/res/android/icon/xxxhdpi.png" density="xxxhdpi"
background="/res/android/icon/xxxhdpi-background.png"
foreground="/res/android/icon/xxxhdpi-foreground.png"/>
</platform>
```

Step 2: Configure Android Keystore

A keystore is a binary file that contains a set of private keys. A private key represents the entity to be identified with the app, such as a person or a company. A keystore is encrypted with a password and it cannot be restored if the password is lost. When a keystore is lost or it overwrites another keystore, it is impossible to use the same key to re-sign the signed package.

A keystore is required for the building of a release version for your Android app. In Monaca, you can either create a new keystore or import an existing one. In order to create a new keystore, please do as follows:

1. From the Monaca Cloud IDE menu, go to **Configure** → **Android KeyStore Settings** .
2. Then, Manage KeyStore and Alias page will appear.



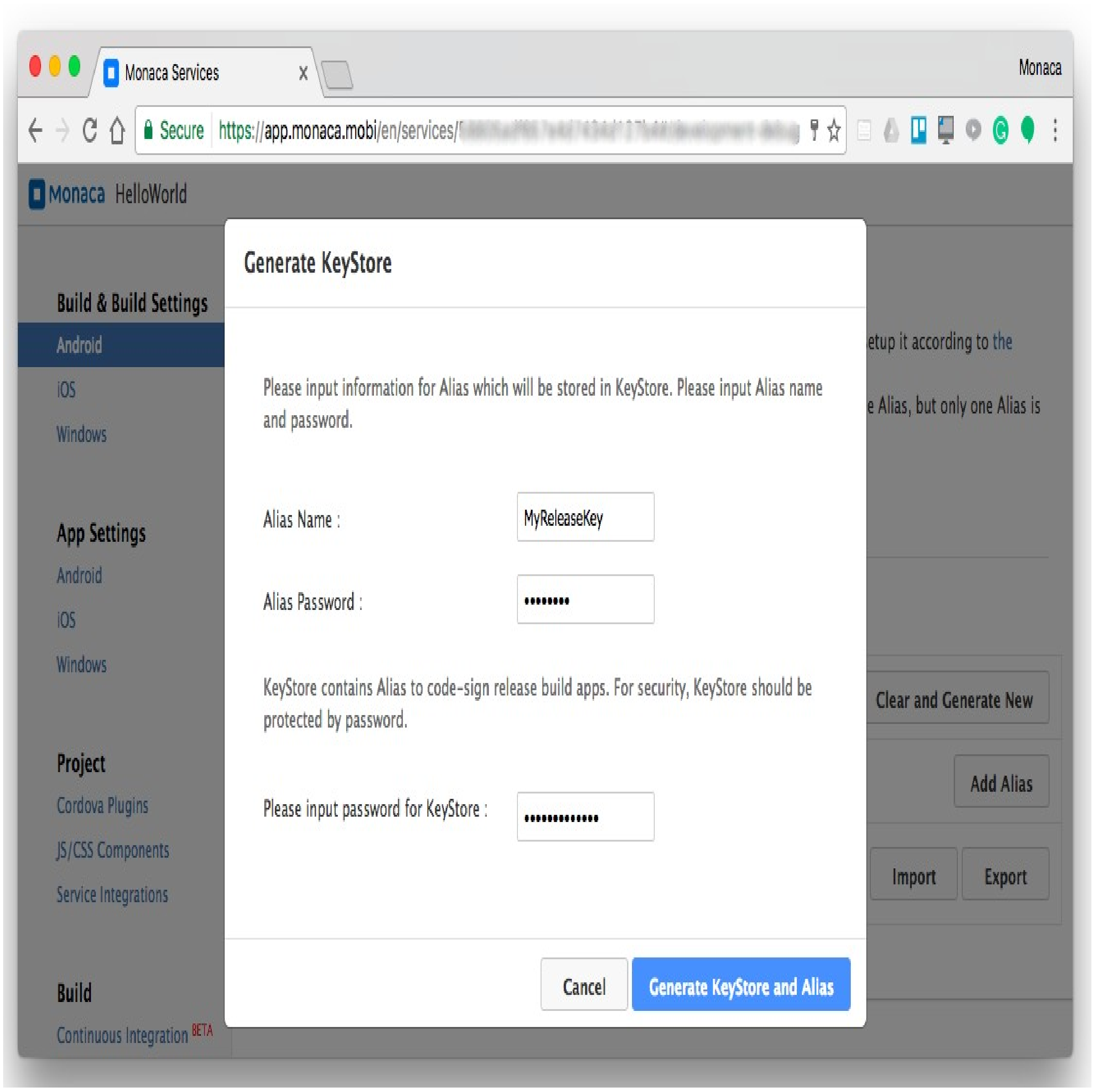
Manage KeyStore and Alias

The project is not configured for Release build. In order to register your App to Google Play, please setup it according to the document.
KeyStore and Alias are used to code-sign applications for release build. KeyStore can contain multiple Alias, but only one Alias is used for code-sign an application.
The configuration is saved by user basis.

Generate New KeyStore:

Create KeyStore and Alias	Create New KeyStore and Alias	Clear and Generate New
Add Alias	Add New Alias	Add Alias
KeyStore Import and Export	A keystore created elsewhere will be imported, and an alias added. Also, it can be exported to the local PC as a backup.	Import Export

3. Click on Clear and Generate New button. Then, the following screen will appear:



Generate KeyStore

Please input information for Alias which will be stored in KeyStore. Please input Alias name and password.

Alias Name : MyReleaseKey

Alias Password : ••••••••

KeyStore contains Alias to code-sign release build apps. For security, KeyStore should be protected by password.

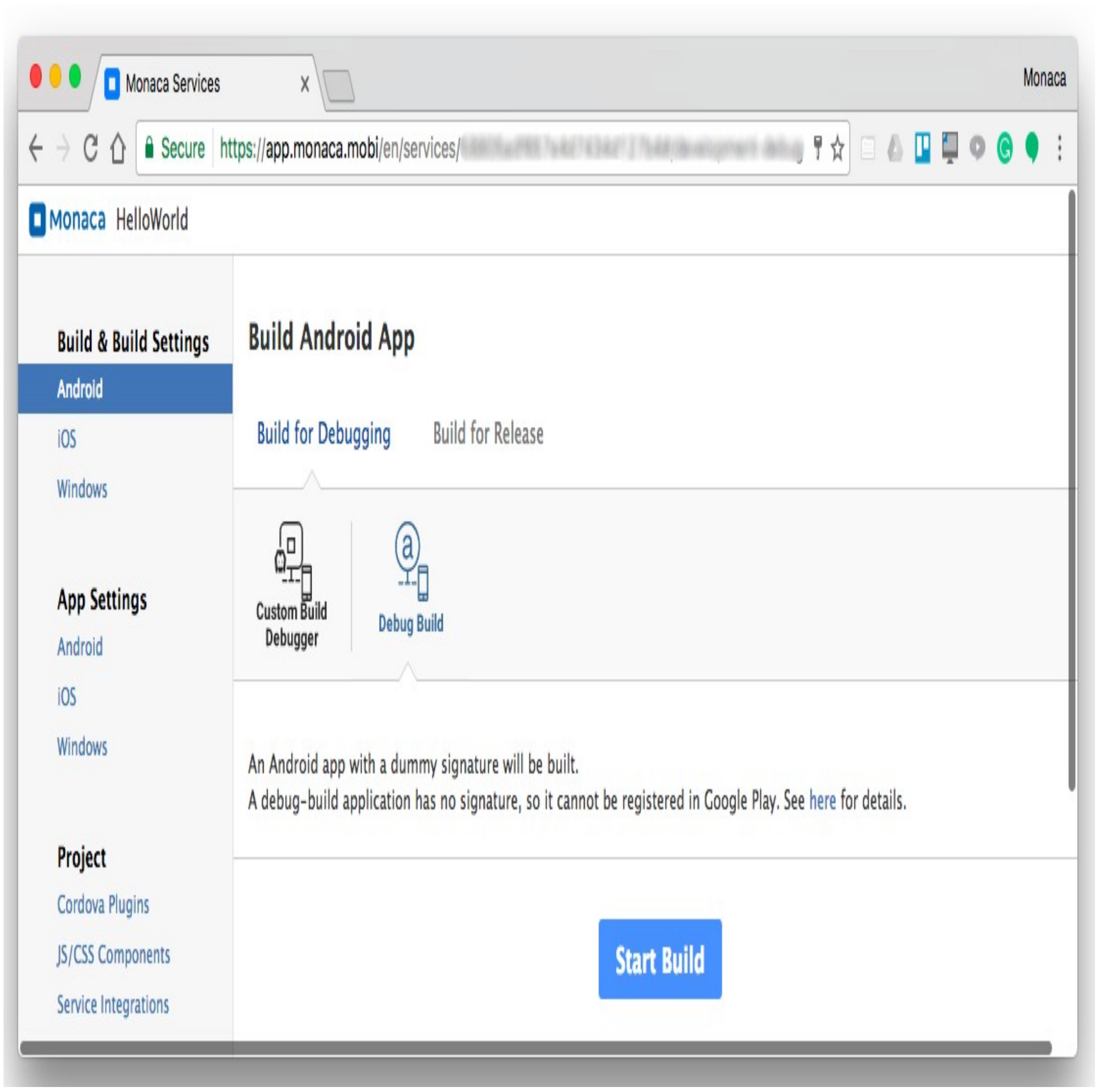
Please input password for KeyStore : •••••••••••••

4. Fill in the necessary information as shown in the above screen such as:
   - `Alias`: a name representing a private key that you will use later when signing your app. Multiple aliases can be stored within one keystore.
   - `Password`: a password for the private key (alias).
   - `Password of the keystore`: a password for the keystore. You will need this password when importing this keystore.
5. Then, click on Generate Keystore and Alias button to Generate the keystore.

When a keystore is lost, it is impossible to use the same key to re-sign the signed package. Therefore, always back up and keep the keystore which is used to sign application(s). Use the Export button to download your keystore.

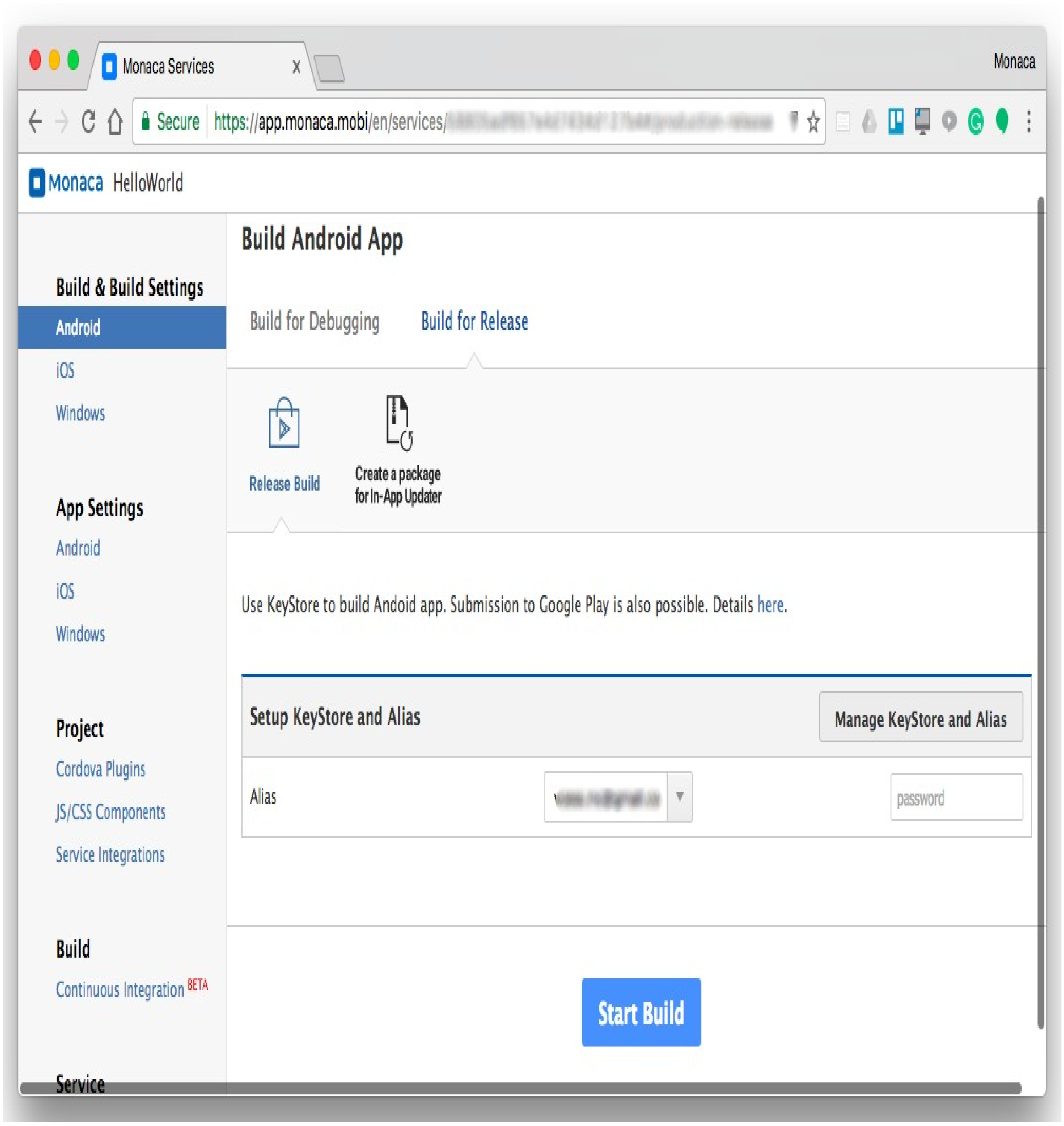
Step 3: Start Building

1. From the Monaca Cloud IDE menu, go to `Build` → `Build App for Android`.
2. Select appropriate type of build you want and click Start Build.



An Android app with a dummy signature will be built.
A debug-build application has no signature, so it cannot be registered in Google Play. See here for details.

3. If you choose `Release Build`, you will also need to select an alias to sign your package before start building.



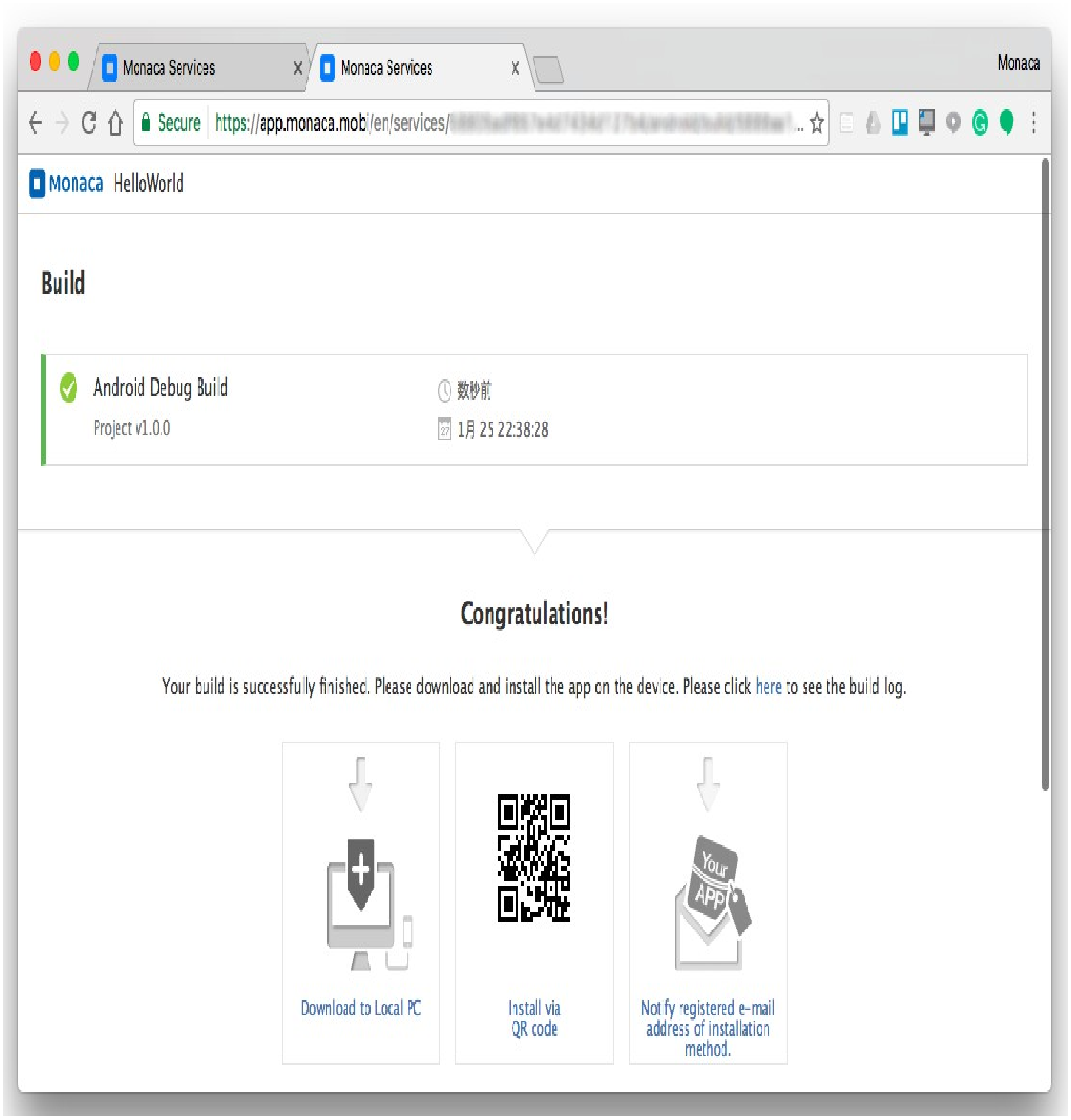
Use KeyStore to build Andoid app. Submission to Google Play is also possible. Details here.

Setup KeyStore and Alias

Manage KeyStore and Alias

Alias	[redacted]	password

4. It may take several minutes for the build to complete. Please wait. Once the build is completed, your built app is ready to be installed/downloaded. See below screenshot as an example:



Build

Android Debug Build
Project v1.0.0
数秒前
1月 25 22:38:28

Congratulations!

Your build is successfully finished. Please download and install the app on the device. Please click here to see the build log.

See Also:

- Building for iOS
- Building for Windows
- Google Play Distribution
- Build History

Non-market App Distribution

For iOS Apps

There are two cases which you want to distribute your applications outside the App Store:

1. Testing Purpose: Before releasing your apps, you would want to test them as much as you can. Thus, you would want to distribute your apps to users (testers) by various ways besides the App Store.
2. In-house Applications: The applications are made for internal uses (in a company or organization) only.

For in-house distribution, you will need to have an Apple Developer Enterprise Program account.

The differences between Apple Developer and Apple Developer Enterprise programs regarding apps distribution:

	Apple Developer Program	**Apple Developer Enterprise Program**
Beta OS Releases	Yes	Yes
Ad Hoc Distribution	Yes	Yes
App Store Distribution	Yes	No
In-house Distribution	No	Yes
TestFlight Beta Testing	Yes	No
Team Management	No	Yes
App Analytics	Yes	No

For more information about the differences between the two programs, please refer to Choosing a Membership.

There are two ways to distribute your pre-release apps for testing:

1. using App Store Connect: it's required iOS Developer Account and takes time since it needs approval from Apple Review prior to the distribution.
2. using Ad Hoc distribution: it can be done with either iOS Developer and iOS Developer Enterprise accounts. Plus, it doesn't require the approval from Apple Review.

In-house distribution is to securely distribute your iOS apps to your employees. In other words, you can distribute your app to any company devices. However, if you want someone outside of your company to test your app or restrict distribution to specific devices, you can use Ad Hoc distribution.

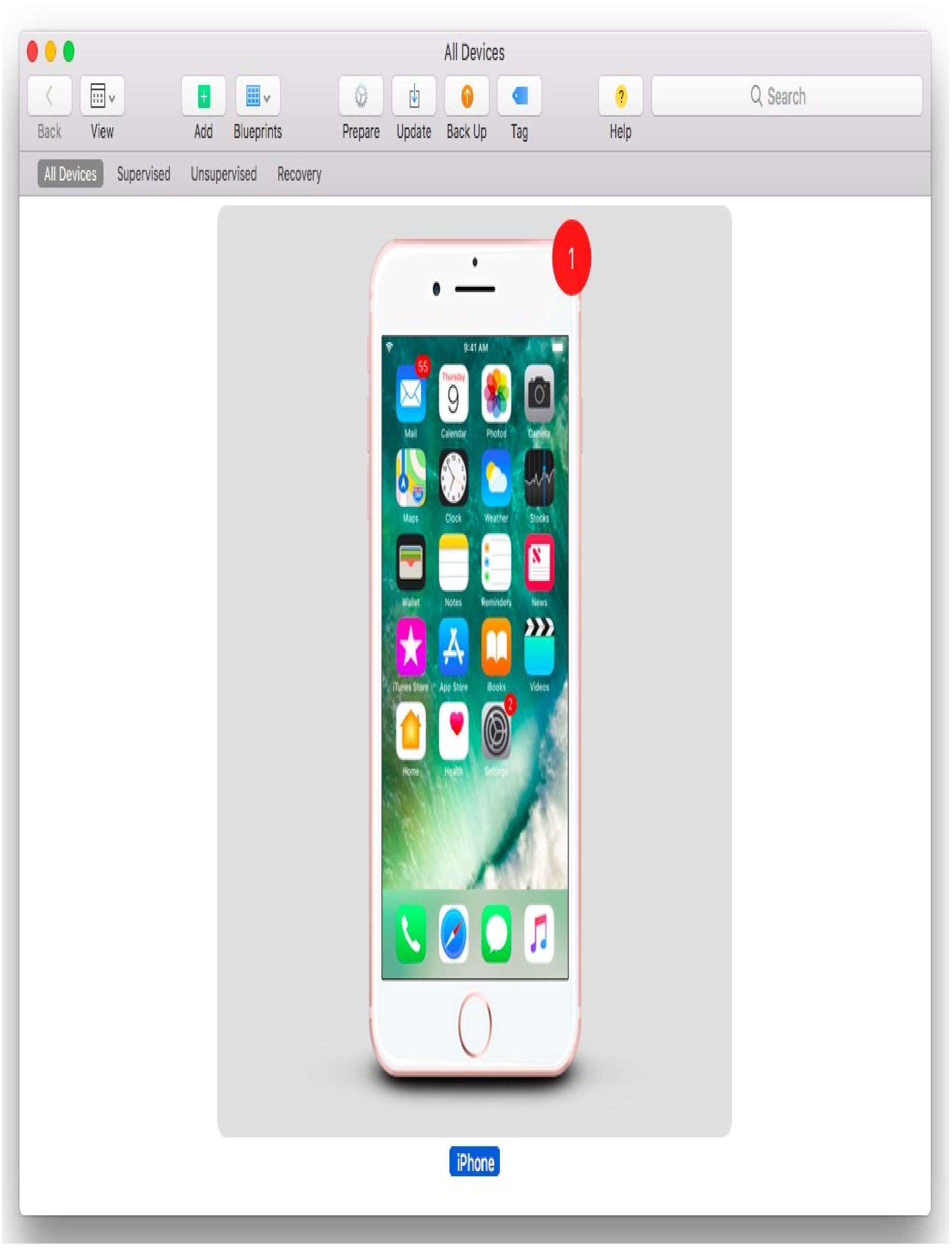
Ad Hoc distribution is to distribute your apps using Ad Hoc provisioning profile to registered devices up to `100`.

Both types of distribution have the same ways for app installation. The difference is the provisioning profile.

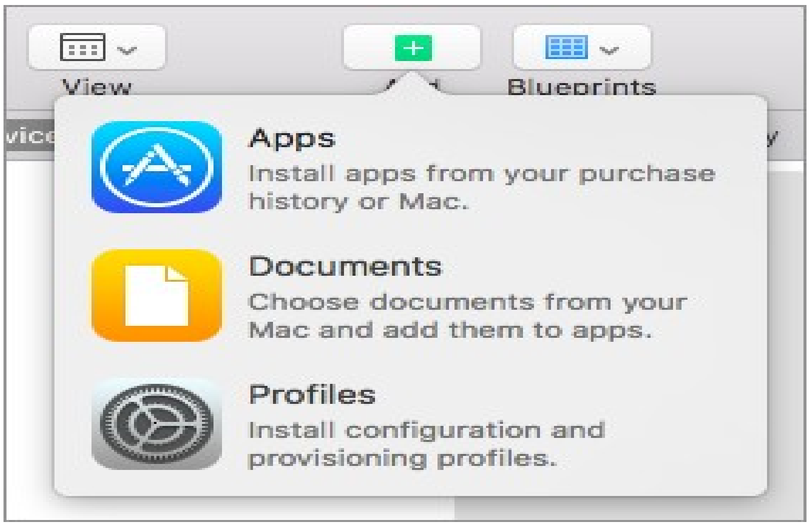
There are several ways you can install your applications outside the App Store as shown in the following sections.

Install using Apple Configurator 2 (Mac Only)

1. Install `Apple Configurator 2` on your Mac from the App Store.
2. Connect your device to your PC.
3. Open `Apple Configurator 2`, select your device. If you device doesn't appear here, please make sure that your device is successfully connected to your Mac.



4. Click on Add button and select `App` option.


View
Add
Blueprints
Apps
Install apps from your purchase history or Mac.
Documents
Choose documents from your Mac and add them to apps.
Profiles
Install configuration and provisioning profiles.

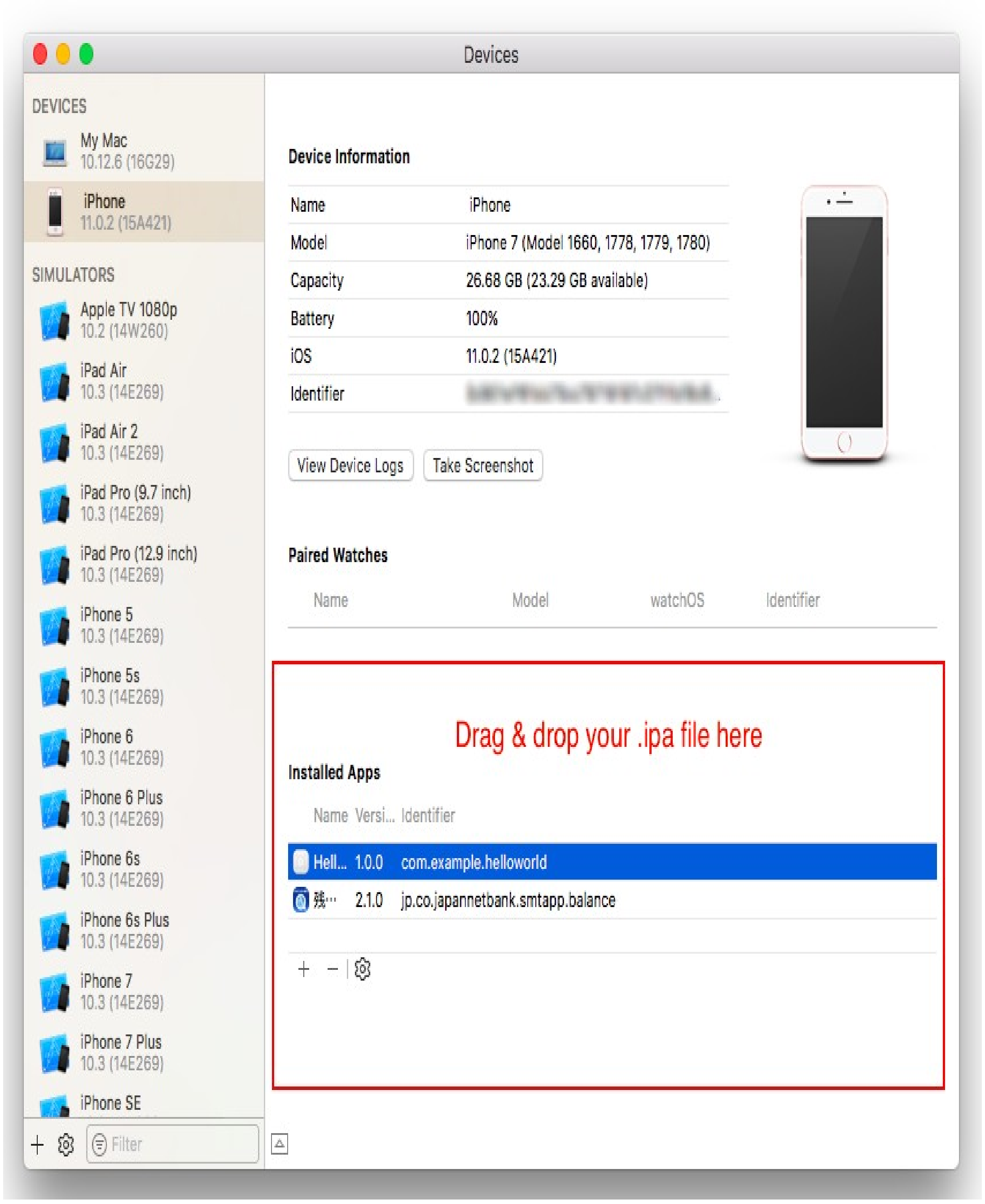
5. Select Choose from my Mac button and browse the `.ipa` file. Then, the app will be installed on your device.

Install using Xcode

You can install your iOS app (`.ipa` file) via Xcode as follows:

1. Connect your device to your PC.
2. Open Xcode, go to **Window** → **Devices** .
3. Then, the Devices screen will appear. Choose the device you want to install the app on.

4. Drag and drop your `.ipa` file into the Installed Apps as shown below:


Devices

DEVICES
My Mac 10.12.6 (16G29)
iPhone 11.0.2 (15A421)

SIMULATORS
Apple TV 1080p 10.2 (14W260)
iPad Air 10.3 (14E269)
iPad Air 2 10.3 (14E269)
iPad Pro (9.7 inch) 10.3 (14E269)
iPad Pro (12.9 inch) 10.3 (14E269)
iPhone 5 10.3 (14E269)
iPhone 5s 10.3 (14E269)
iPhone 6 10.3 (14E269)
iPhone 6 Plus 10.3 (14E269)
iPhone 6s 10.3 (14E269)
iPhone 6s Plus 10.3 (14E269)
iPhone 7 10.3 (14E269)
iPhone 7 Plus 10.3 (14E269)
iPhone SE

Device Information

Name	iPhone
Model	iPhone 7 (Model 1660, 1778, 1779, 1780)
Capacity	26.68 GB (23.29 GB available)
Battery	100%
iOS	11.0.2 (15A421)
Identifier	[redacted]

Paired Watches

Name	Model	watchOS	Identifier

Drag & drop your .ipa file here

Installed Apps

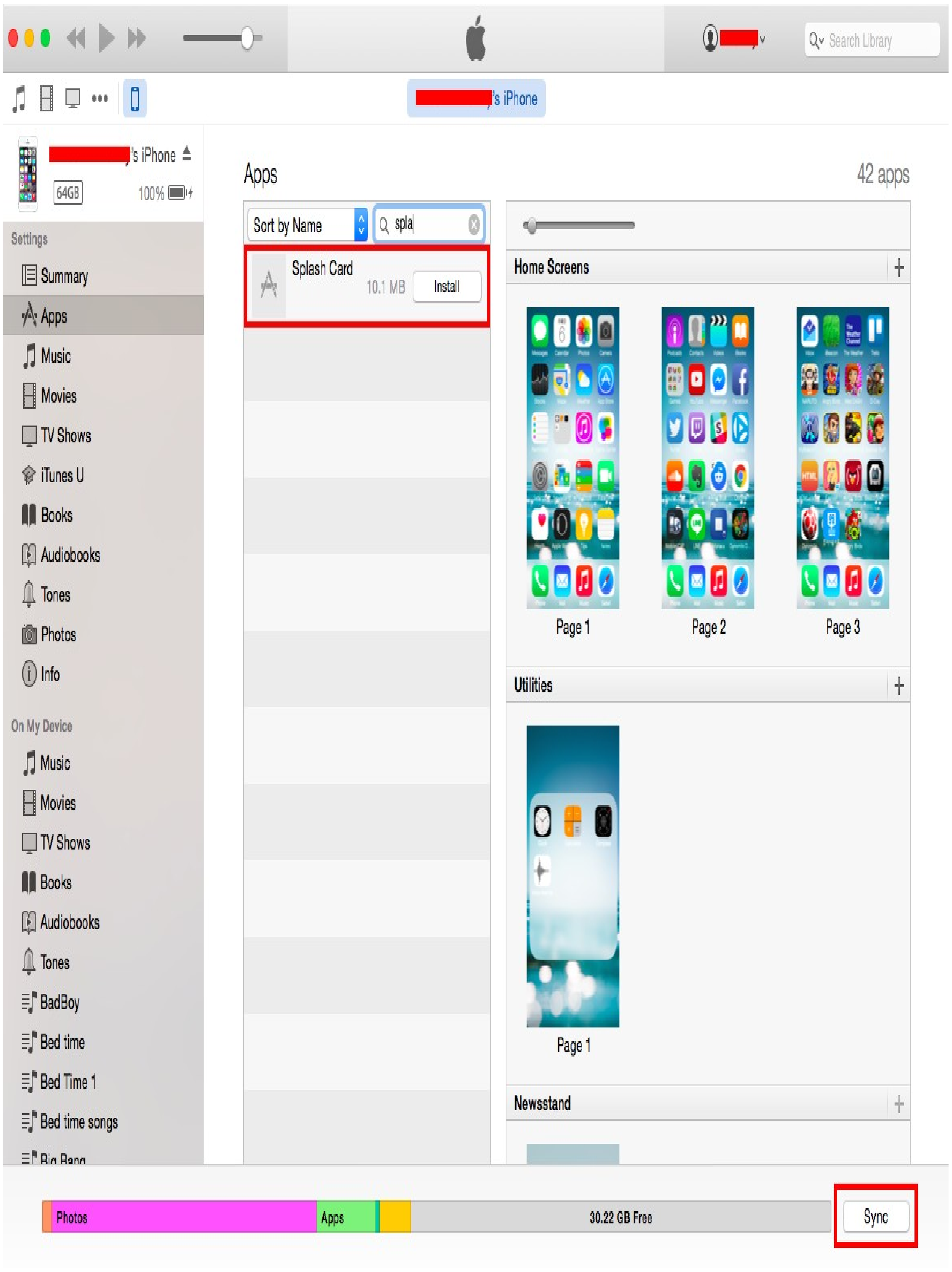
Name	Versi...	Identifier
Hell...	1.0.0	com.example.helloworld
残...	2.1.0	jp.co.japannetbank.smtapp.balance

Install using iTunes

iTunes 12.7 for Mac was released on Tuesday with a major change in the app. Apple has redesign iTunes so that it focuses on sales of music, movies, TV shows, audiobooks, and podcasts. It no longer has an App Store for buying apps for your iPhone or iPad. Therefore, you can no long install your iOS App (.ipa file) through iTunes any longer.

1. Build your application with with debug or ad-hoc build. For more details about iOS build process, please refer to Building an iOS App.
2. Download the `.ipa` file after the build completes.
3. Open iTunes, go to `App library`.
4. Drag and drop the downloaded `.ipa` file into the `App library`.
5. Connect your device to iTunes and go to your device apps.

6. Click Install button of the app and click Sync button. See the example below:



Install using OTA Deployment

OTA (Over-The-Air) Deployment enables you to install your built apps via HTTPS.

1. Build your application with either debug, ad-hoc or in-house build. For more details about iOS build process, please refer to Building an iOS App.
2. Download the `.ipa` file after the build completes.
3. Upload the `.ipa` file to the site you want.
4. Create a `.plist` file for this built application. The `.plist` file should look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>items</key>
    <array>
        <dict>
            <key>assets</key>
            <array>
                <dict>
                    <key>kind</key>
                    <string>software-package</string>
                    <key>url</key>

<string>https://www.anysite.com/application/your_app.ipa</string>
                </dict>
            </array>
            <key>metadata</key>
            <dict>
                <key>bundle-identifier</key>
                <string>com.example.helloworld</string>
                <key>bundle-version</key>
                <string>1.0.0</string>
                <key>kind</key>
                <string>software</string>
                <key>title</key>
                <string>HELLO</string>
            </dict>
        </dict>
    </array>
</dict>
</plist>
```

   While creating `.plist` file, please pay attention to these points:

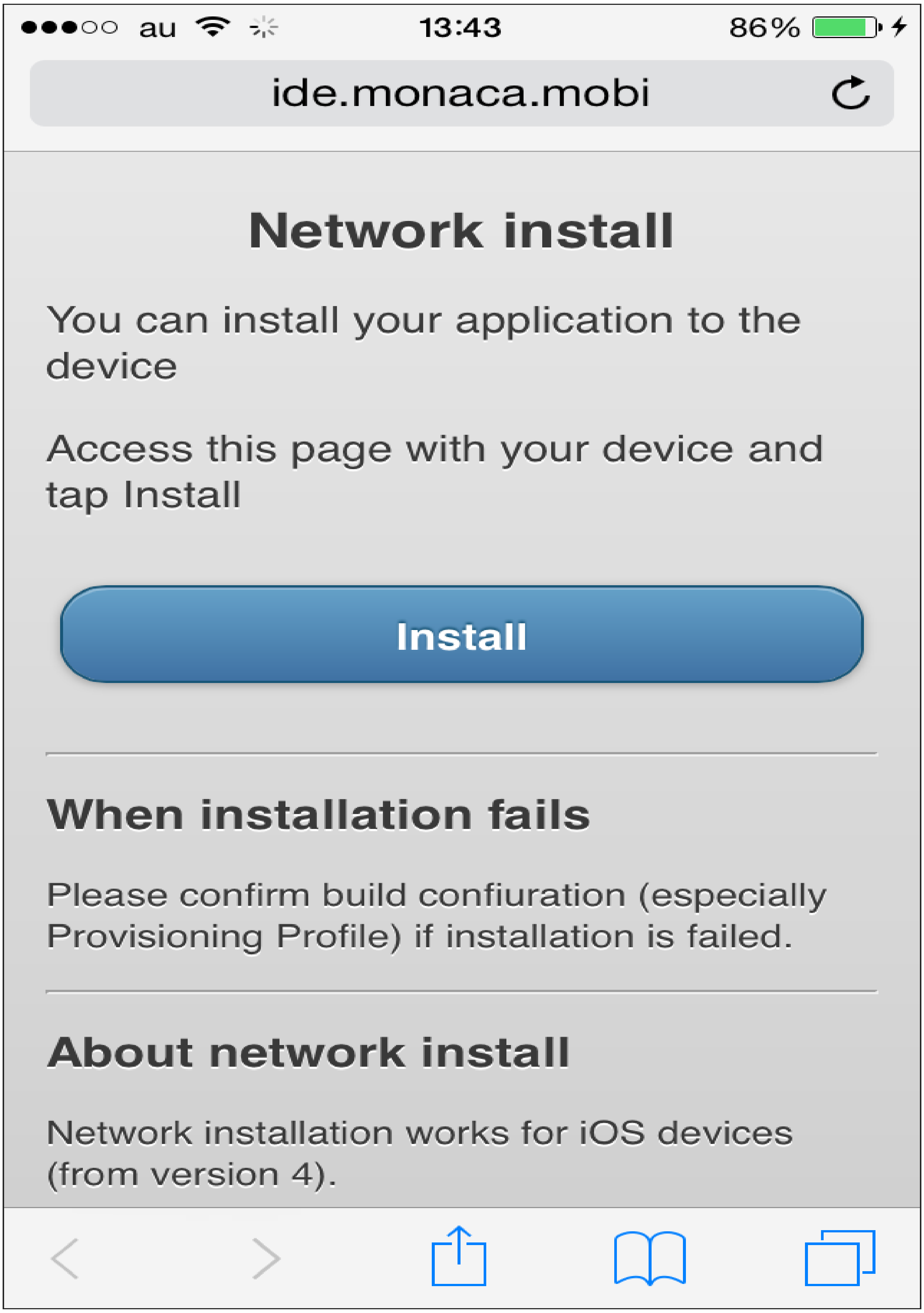
   - `.plist` file must be accessed via https protocol.
   - update bundle-identifier with the App ID.
   - specify correct path to the `.ipa` file.

5. Upload the `.plist` file to the site you want. Make sure this file must be accessed via HTTPS protocol.

6. Create a webpage embedded the link to the uploaded `.plist` file using special `itms-services://` protocol. See blow example:

```
<a href="itms-services://?action=download-manifest&amp;
    url=https://www.anysite.com/application/your_app.plist">
    Download
</a>
```

7. After you get the link, use your device to access the link. Then, you will be prompted to install the application. See below example:



Network install

You can install your application to the device

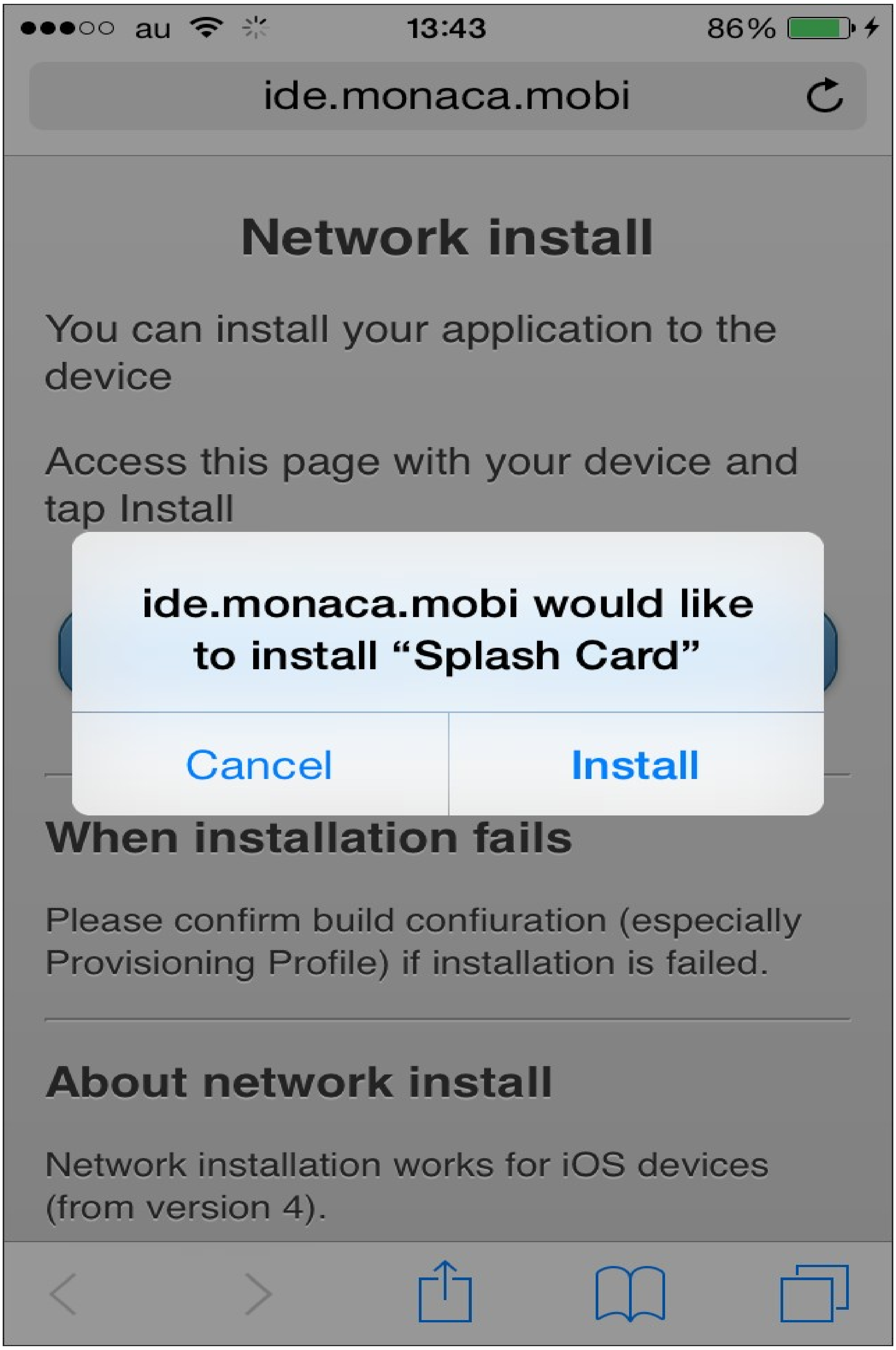
Access this page with your device and tap Install

When installation fails

Please confirm build confiuration (especially Provisioning Profile) if installation is failed.

About network install

Network installation works for iOS devices (from version 4).



Network install

You can install your application to the device

Access this page with your device and tap Install

ide.monaca.mobi would like to install "Splash Card"

When installation fails

Please confirm build confiuration (especially Provisioning Profile) if installation is failed.

About network install

Network installation works for iOS devices (from version 4).

For Android Apps

There are two cases which you want to distribute your applications outside the offical markets such as Google Play Store, Amazon AppStore and so on:

1. Testing Purpose: Before releasing your apps, you would want to test them as much as you can. Thus, you would want to distribute your apps to users (testers) by various ways besides the official markets.
2. Personal/Internal Purpose: The applications are made for your own personal uses or just for internal uses.

Install using ADB command

ADB (Android Debug Bridge) is a tool enabling you to use various terminal commands to your phone.

Prerequisite:

- install Android SDK on your computer
- locate the ADB path after Android SDK installation
- enable USB Debugging and allow installation of apps from sources other than the Play Store on your device.

In order to install your built app via ADB command:

1. Build your application with debug build. For more details about Android build process, please refer to Building for Android.
2. Download the `.apk` file after the build completes.
3. Plug your device via USB to your computer.
4. Run below command on your computer in command window. Make sure to use the correct path to your `.apk` file.

```
adb install foo.apk
```

Install using Direct Link

This is simply an installation through direct link to your `.apk` file:

1. Build your application with debug build. For more details about Android build process, please refer to Building for Android.
2. Download the `.apk` file after the build completes.
3. Upload the downloaded file to any sites you want.
4. Go the link of the uploaded file from your device. Then, you will be prompted to install the applicaiton.

   Please make sure make these settings on your device beforehand:
   - enable USB Debugging.
   - allow installation of apps from sources other than the Play Store.

See Also:

- Building for iOS

- Building for Android
- Google Play Distribution
- App Store Distribution

Building for Android

Types of Build

In Monaca, Android app has two types of build: debug version and release version. The differences between these types of build are as follows:

Types of Build	Description	Installation
Debug Build	An unsigned package which cannot be distributed in the market	• QR Code • Network Install • Sideloading
Release Build	A signed package with the developer’s code sign which can be distributed in the market	• Sideloading • Google Play Store and other eligible markets

Sideloading typically refers to media file transfer to a mobile device via USB, Bluetooth, WiFi or by writing to a memory card for insertion into the mobile device. When referring to Android apps, "sideloading" typically means installing an application package in APK format onto an Android device without going through the market.

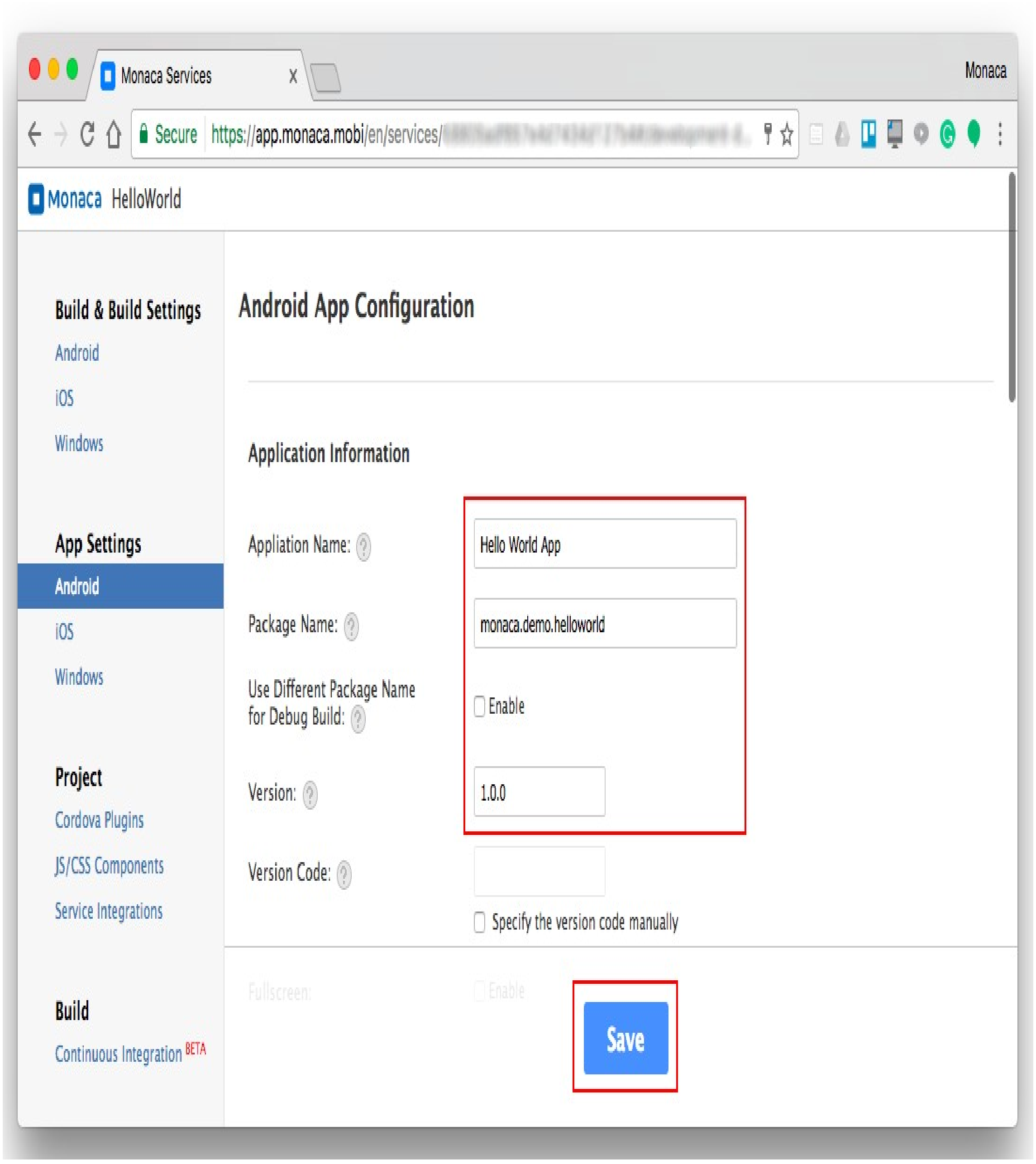
Step 1: Configure Android App

1. From the Monaca Cloud IDE menu, go to **Configure** → **App Settings for Android** .
2. Fill in the necessary information of your app:

   - General settings:

Application Name	A name representing your app publicly such as in the Market
Package Name	A unique name which will be used when uploading to the Android Market. It is recommended to use reverse-domain style (for example, io.monaca.app_name) for App ID. Only alphanumeric characters, periods (at least one period must be used) and underscore are allowed. Each segment should be separated by a period and started with an alphabetic character.
Use Different Package Name for Debug Build	If enable, the package name of the release-built and debug-built apps are different. In other words, the package name of debug-built app will have `.debug` extension, and the one for project debugger will have `.debugger` extension. However, this option is disable by default because it made some plugins impossible to be debugged due to the fact that they are tied to exact package names (eg. in-app purchase).
Version	The version number of your app. A version number consist of only number seperated by dots (for example, 1.0.0).

Version Code	An internal version number of your app, relative to other versions. The value must be integer, so that the applications can programmatically evaluate it for an upgrade.
Fullscreen	This option is only available with the Cordova 3.5 and later. If enable, your app will be run in a fullscreen mode which hide the status bar.



https://app.monaca.mobi/en/services/[redacted]

Android App Configuration

Application Information

Appliation Name:	Hello World App
Package Name:	monaca.demo.helloworld
Use Different Package Name for Debug Build:	☐ Enable
Version:	1.0.0
Version Code:	
	☐ Specify the version code manually
Fullscreen:	☐ Enable

- Misc: various settings regarding your Android app such as:

Allowed URL	*	Specify URL(s) which can be accessed from your app. If set to *, you can access all domains from your app.
Keep Running	Enable	Enable this if you want Cordova to keep running in the background.
Disallow Overscroll	Enable	Enable this if you want to disable the glow when a user scrolls beyond the edge of the webview.
Screen	Default	You can also set the device’s screen orientation

Orientation when running your app as Landscape or Portrait. After finishing the configurations, click Save.

Currently, when you update either iOS's App ID or Android's Package Name, both of them will change. In other words, they are configured to be the same. However, it is possible to make them different. Please refer to How to make iOS’s App ID and Android’s Package Name differently .

3. Set of adaptive icons

   If you are using a monaca project with cordova 9.0 or later, you can set an adaptive icon for android 8.0 or later.

   To set an adaptive icon, upload an image file under the `res` folder and add a `background` attribute and a `foreground` attribute to the `icon` tag of config.xml. If the previous src attribute is set, the value of the src attribute is not used.

   If the os version does not support adaptive icons, the image set in the `foreground` attribute will be displayed. If the previous `src` attribute is set, the value of the src attribute takes precedence.

   The following is an example of placing an image under `/res/android/icon/`.

```
<platform name="android">
    <icon density="ldpi" background="/res/android/icon/ldpi-
background.png" foreground="/res/android/icon/ldpi-foreground.png"/>
    <icon density="mdpi" background="/res/android/icon/mdpi-
background.png" foreground="/res/android/icon/mdpi-foreground.png"/>
    <icon density="hdpi" background="/res/android/icon/hdpi-
background.png" foreground="/res/android/icon/hdpi-foreground.png"/>
    <icon density="xhdpi" background="/res/android/icon/xhdpi-
background.png" foreground="/res/android/icon/xhdpi-foreground.png"/>
    <icon density="xxhdpi" background="/res/android/icon/xxhdpi-
background.png" foreground="/res/android/icon/xxhdpi-foreground.png"/>
    <icon density="xxxhdpi" background="/res/android/icon/xxxhdpi-
background.png" foreground="/res/android/icon/xxxhdpi-foreground.png"/>
</platform>
```

   The following is an example of setting the src attribute.

```
<platform name="android">
    <icon src="/res/android/icon/ldpi.png" density="ldpi"
background="/res/android/icon/ldpi-background.png"
foreground="/res/android/icon/ldpi-foreground.png"/>
    <icon src="/res/android/icon/mdpi.png" density="mdpi"
background="/res/android/icon/mdpi-background.png"
foreground="/res/android/icon/mdpi-foreground.png"/>
    <icon src="/res/android/icon/hdpi.png" density="hdpi"
background="/res/android/icon/hdpi-background.png"
foreground="/res/android/icon/hdpi-foreground.png"/>
    <icon src="/res/android/icon/xhdpi.png" density="xhdpi"
background="/res/android/icon/xhdpi-background.png"
foreground="/res/android/icon/xhdpi-foreground.png"/>
    <icon src="/res/android/icon/xxhdpi.png" density="xxhdpi"
background="/res/android/icon/xxhdpi-background.png"
foreground="/res/android/icon/xxhdpi-foreground.png"/>
    <icon src="/res/android/icon/xxxhdpi.png" density="xxxhdpi"
background="/res/android/icon/xxxhdpi-background.png"
foreground="/res/android/icon/xxxhdpi-foreground.png"/>
```

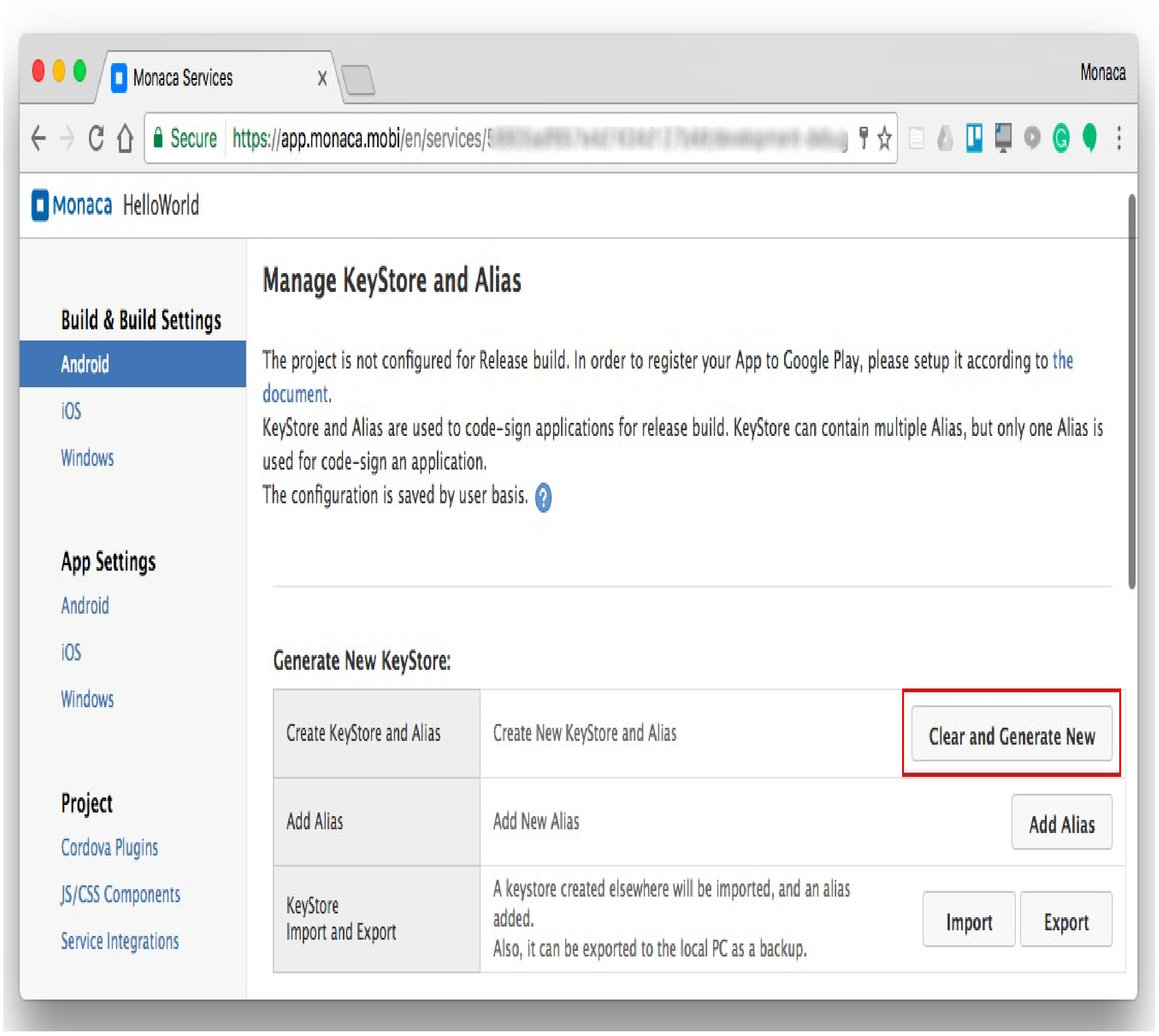
```
</platform>
```

Step 2: Configure Android Keystore

A keystore is a binary file that contains a set of private keys. A private key represents the entity to be identified with the app, such as a person or a company. A keystore is encrypted with a password and it cannot be restored if the password is lost. When a keystore is lost or it overwrites another keystore, it is impossible to use the same key to re-sign the signed package.

A keystore is required for the building of a release version for your Android app. In Monaca, you can either create a new keystore or import an existing one. In order to create a new keystore, please do as follows:

1. From the Monaca Cloud IDE menu, go to **Configure** → **Android KeyStore Settings** .
2. Then, Manage KeyStore and Alias page will appear.



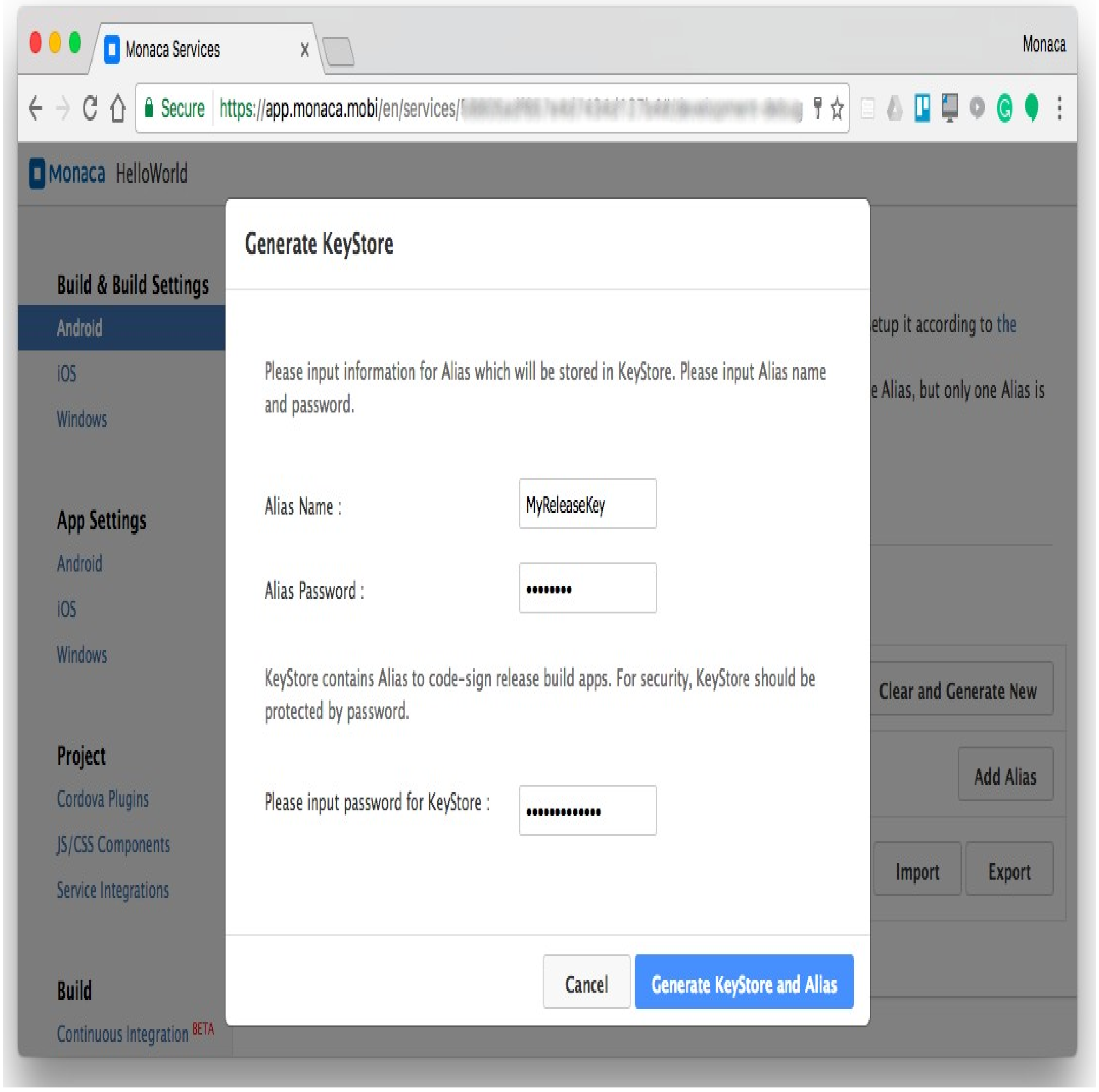
Manage KeyStore and Alias

The project is not configured for Release build. In order to register your App to Google Play, please setup it according to the document.
KeyStore and Alias are used to code-sign applications for release build. KeyStore can contain multiple Alias, but only one Alias is used for code-sign an application.
The configuration is saved by user basis.

Generate New KeyStore:

Create KeyStore and Alias	Create New KeyStore and Alias	Clear and Generate New
Add Alias	Add New Alias	Add Alias
KeyStore Import and Export	A keystore created elsewhere will be imported, and an alias added. Also, it can be exported to the local PC as a backup.	Import Export

3. Click on Clear and Generate New button. Then, the following screen will appear:


Generate KeyStore

Please input information for Alias which will be stored in KeyStore. Please input Alias name and password.

Alias Name : MyReleaseKey

Alias Password : ••••••••

KeyStore contains Alias to code-sign release build apps. For security, KeyStore should be protected by password.

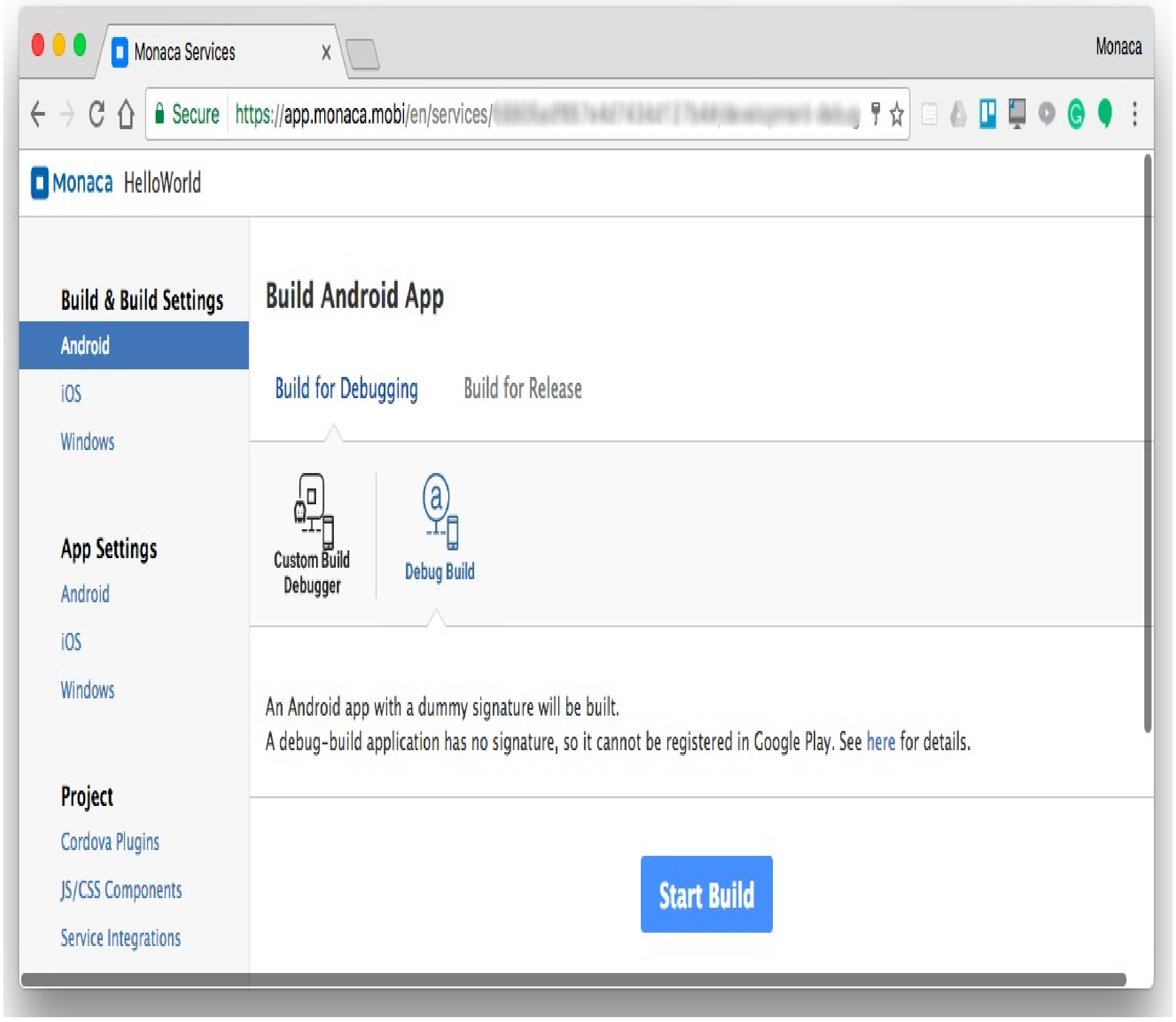
Please input password for KeyStore : •••••••••••••

4. Fill in the necessary information as shown in the above screen such as:
   - `Alias`: a name representing a private key that you will use later when signing your app. Multiple aliases can be stored within one keystore.
   - `Password`: a password for the private key (alias).
   - `Password of the keystore`: a password for the keystore. You will need this password when importing this keystore.
5. Then, click on Generate Keystore and Alias button to Generate the keystore.

When a keystore is lost, it is impossible to use the same key to re-sign the signed package. Therefore, always back up and keep the keystore which is used to sign application(s). Use the Export button to download your keystore.

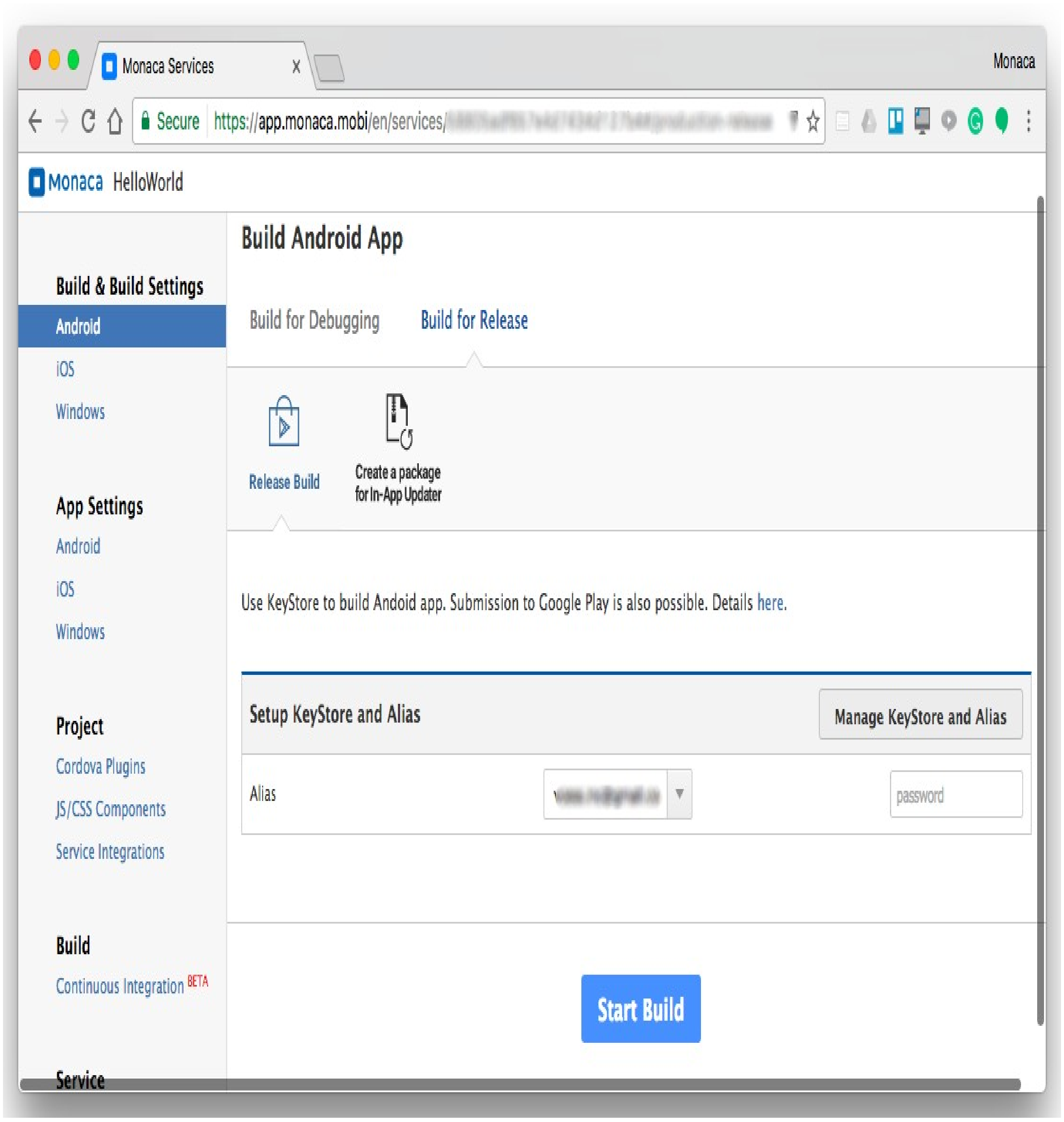
Step 3: Start Building

1. From the Monaca Cloud IDE menu, go to `Build` → `Build App for Android`.
2. Select appropriate type of build you want and click Start Build.



An Android app with a dummy signature will be built.
A debug-build application has no signature, so it cannot be registered in Google Play. See here for details.

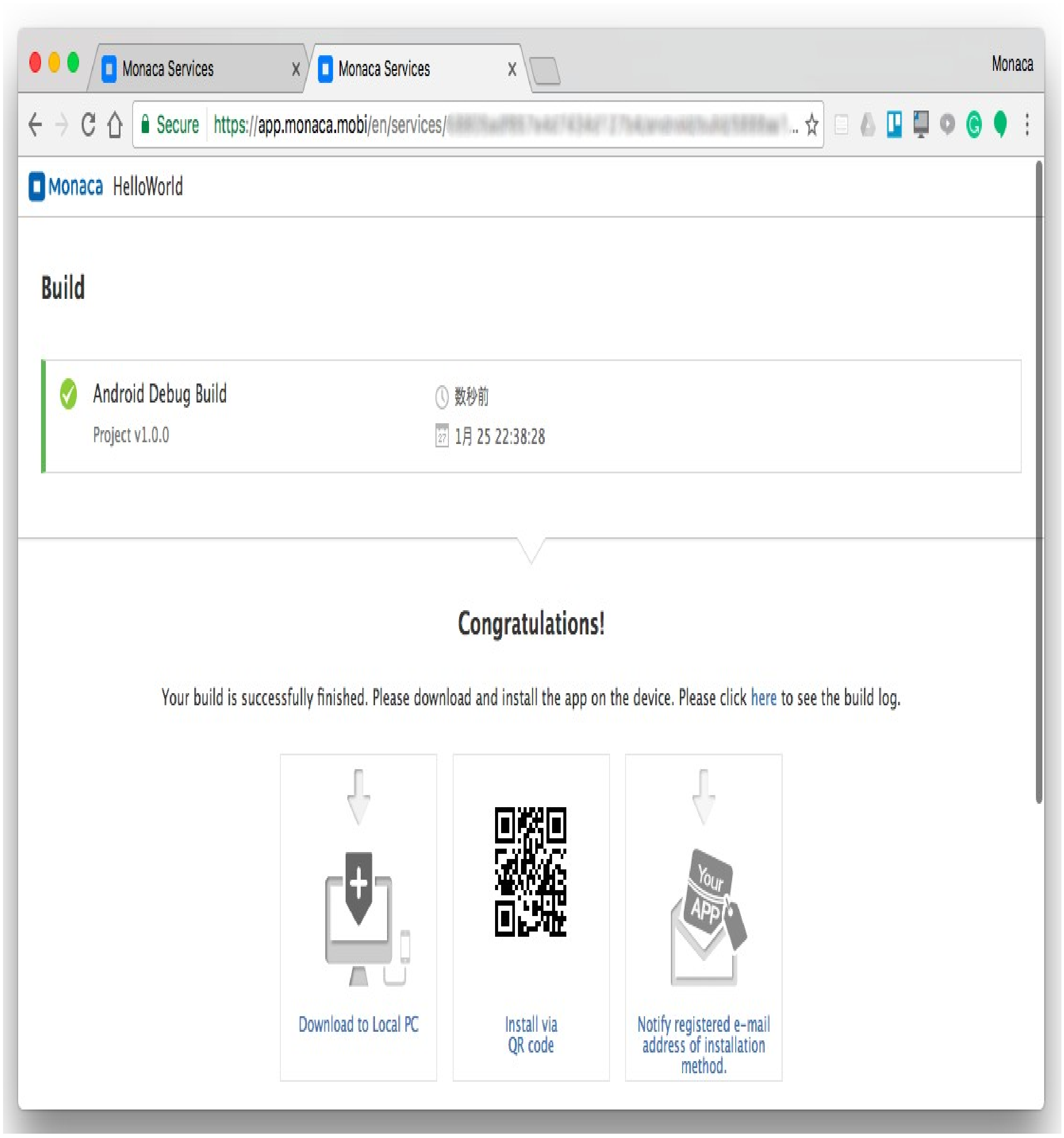
3. If you choose `Release Build`, you will also need to select an alias to sign your package before start building.



Use KeyStore to build Andoid app. Submission to Google Play is also possible. Details here.

Alias [redacted] password

4. It may take several minutes for the build to complete. Please wait. Once the build is completed, your built app is ready to be installed/downloaded. See below screenshot as an example:



Build

Android Debug Build
Project v1.0.0
数秒前
1月 25 22:38:28

Congratulations!

Your build is successfully finished. Please download and install the app on the device. Please click here to see the build log.

See Also:

- [Building for iOS]()
- [Building for Windows]()
- [Google Play Distribution]()
- [Build History]()