Rock.MobileApp

The purpose of this document is to guide you in the setup of Rock.MobileApp to work for your organization. When you are finished, you’ll have an app ready for upload to the iOS App Store and Google Play Store!

Table of Contents

[Getting Started 1](#_Toc419964596)

[Basic Requirements 1](#_Toc419964597)

[Detailed Reading 1](#_Toc419964598)

[Making Rock.MobileApp Your Own 1](#_Toc419964599)

[App Identification 2](#_Toc419964600)

[App Customization 2](#_Toc419964601)

[Artwork Customization 3](#_Toc419964602)

## Getting Started

### Basic Requirements

If you haven’t already, begin by ensuring you have the basic requirements to setup the mobile app.

[Xamarin Studio](http://xamarin.com/download) – This is the development environment you will use to compile and run the mobile app.

[iOS Developer License](https://developer.apple.com/programs/ios/) – This is required to publish and keep the app available in the iOS App Store.

[Google Developer](https://developers.google.com/) – While there is no required licensing fee to publish on the Google Play Store, you will want an account setup so you can utilize APIs such as Google Maps.

Once these requirements are setup, you should begin by ensuring you can compile and run the included sample app.

This can be done by opening the Xamarin Project File “RSCApp.sln” located in the RSCApp folder.

Once open, choose compile, and the project should build successfully.

### Detailed Reading

The hope is that this document will be sufficient for customizing Rock.MobileApp. If you wish to gain a deeper understanding of the design and usage of the app, the following document is recommended reading.

* [MobileApp Code Overview](MobileApp%20Code%20Overview.docx) – Read this to familiarize yourself with the specifics of the code project layout, and the code architecture of the Mobile App.

## Making Rock.MobileApp Your Own

You may notice when running the app that it’s named “RSC”. This stands for “Rock Solid Church”, and simply serves as an example to illustrate how the app can be customized.

Once you’ve built and run the sample, it’s time to begin converting for your organization. There are three primary steps involved:

### App Identification

In this step, you will be “rebranding” the app so that when compiled and deployed, the OS recognizes it as your organization’s app. This involves changing a handful of fields in various files. Let’s walk through that now.

* Update the Bundle/Package name.
  + *The Bundle / Package name is what iOS and Android use to identify unique applications.*
  + Open “Droid/Properties/AndroidManifest.xml”
    - Find package=”com.rsconline.RSCMobileApp”. Change this to fit your organization.
    - Find android:label="RSC Mobile". Change this to fit your organization.
  + Open “iOS/Info.plist”
    - Find CFBundleIdentifier. Change this to fit your organization.
    - Find CFBundleDisplayName. Change this to fit your organization.
* Update the Localytics app ID. **(If you don’t need Localytics, you can skip this step.)**
  + You will need a Localytics account to use their analytics system.
  + Open “Droid/Properties/AndroidManifest.xml”
    - Find <meta-data android:name="LOCALYTICS\_APP\_KEY" android:value="0" />. Change this to fit your organization.
  + Open “App.Shared/Config.cs”
    - Find public const bool Use\_Localytics = false; and set it to true.
    - Find public const string Localyitics\_iOS\_Key = "0";. Change this to fit your organization.
* Update the Google Play Maps API Key.
  + You will need to register your application’s Package / Bundle name with Google. They will then give you a unique key that will grant your app access to Google Play Maps.
  + Open “Droid/Properties/AndroidManifest.xml”
    - Find <meta-data android:name="com.google.android.maps.v2.API\_KEY" android:value="0" />. Change this to fit your organization.
* Update the Server URLs.
  + Open “App.Shared.Config.cs”
    - Find public const string RockBaseUrl = "http://rock.ccvonline.com/";. Change this to your organization’s Rock host domain.
    - Find public const string NoteBaseURL = "http://ccv.church/ccvmobile/";. Change this to where you will store the Note Database.

### App Customization

You’re finished with the dry technical part. It’s time to start customizing the app! 90% of the work you will be doing is accomplished in two files:

Code

Config.cs – This file contains every configuration value worth changing. It covers colors, fonts, styles, etc.

Strings.cs – This file contains all the “language” used throughout the app. It is a central place to change all terminology.

Each setting in the files above includes comments explaining what it controls. It’s best to open the files and begin experimenting.

### Artwork Customization

Now that the app is configured to more closely fit your organization, it’s time to begin updating art assets.

In the “Assets” folder, you will find dozens of “default” images that are used when Rock data isn’t available, or permanently where it makes sense. (Such as logos.)

Most of these should be self explanatory based on their names.

The images with the “@2x” or “@3x” suffix are used by iOS for higher resolution devices.

#### Splash Screen and animation

If you run the sample app, you will notice that the logo in the center of the splash screen “zooms” toward you as it fades out.

This effect is achieved by creating a view that appears identical to the splash screen. The difference in this view is that the background and logo are separate images.

That way, the cross dissolve from the actual splash screen to our “fake” splash screen is invisible to the user, and we can then apply a zoom effect to only the logo.

You will find the splash screens in “/iOS/Resources/Image.xcassets/LaunchImage.launchimage”.

Notice that their naming convention is “splash\_combo\_device\_orientation.png”

This is to make it simple to create the “sepereated” versions.

The separated images exist in “/Assets” and have the naming convention “splash\_**layer**\_device\_orientation” where layer is either “bg” or”logo”.

Let’s look at the iPhone 6 as a quick example:

Splash Screen: “/iOS/Resources/Image.xcassets/LaunchImage.launchimage /splash\_combo\_iphone6.png”

Separate Background Image: “/Assets/splash\_bg\_iphone6.png”

Separate Logo Image: “/Assets/splash\_logo\_iphone6.png”

By creating the initial splash screen, and then separating the elements in to Background and Logo images, the app will have everything it needs to create the effect.

**Note for Android:** Due to so many varying screen sizes, the effect on Android uses only a single logo on a solid background. This makes it much easier for Android setup.

The Android splash screen is located in “/Assets/splash\_combo\_android.png”

The single logo it uses for its effect is located in “/Assets/splash\_logo\_android.png”

#### Icons

Icons are very straight forward to customize.

For iOS, they are located in “/iOS/Resources/Images.xcassets/AppIcons.appiconset/”

For Android, they are located in

“/Assets/Icons/”