UHC Mobile (LOWA) Developer Onboarding

Android Emulator Setup

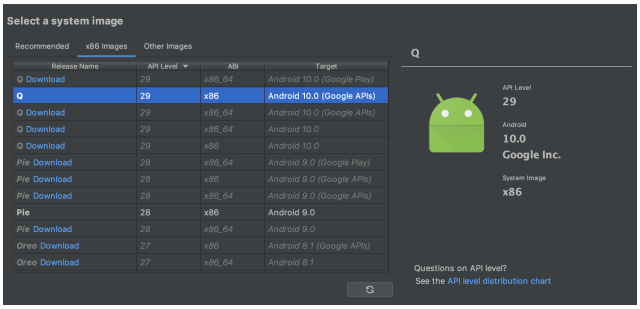
1. Download Android Studio
2. Setup your AVD manager (you will only have to do this once)
   a. Open Android Studio
   b. Select "File → New → New Project..." and go with all of the default settings (it doesn't matter)
   c. Wait for the project to build, then press CMD + SHIFT + A, search for "avd manager" and click it to launch
      i. If you get a popup that says "Do you want the application "java" to accept incoming network connections?" you can click Allow, it's the OSX firewall
   d. Select "Create Virtual Device" and select a device
   e. After clicking 'next', you should be on a page that says "Select a system image". Download and select API 29 for the system image
3. From now on, you can access 'Configure → AVD Manager' from the main menu on startup to run your emulator
4. Alternatively, you can run `emulator -avd DeviceName` from the terminal to open your emulator without running Android Studio. You can find your DeviceName by running `emulator -list-avds`. But before doing that, ensure your .bash_profile has the following entries:
   a. export ANDROID_SDK=$HOME/Library/Android/sdk
   b. export PATH=$ANDROID_SDK/emulator:$ANDROID_SDK/tools:$PATH

Copy friendly

```
export ANDROID_SDK=$HOME/Library/Android/sdk
export PATH=$ANDROID_SDK/emulator:$ANDROID_SDK/tools:$PATH
```

   c. If you get an error similar to `PANIC: Missing emulator engine program`, you may also need to set your ANDROID_SDK env variable (can just be the same value as $ANDROID_HOME)

Note: Sometimes Android emulators run slow. When app just freezes on your android emulator, try to modify your RAM or CPU allocations of your emulator. Go to Tools → AVD Manager → Click on the pencil Icon on your emulator → Show Advanced Settings → See Memory and Storage section to control your resources. If the UI doesn't let you modify these options, try to install the emulator again. When installing image, make sure to install x86 images from x86 images tab, and not the "Recommended" tab seen here:



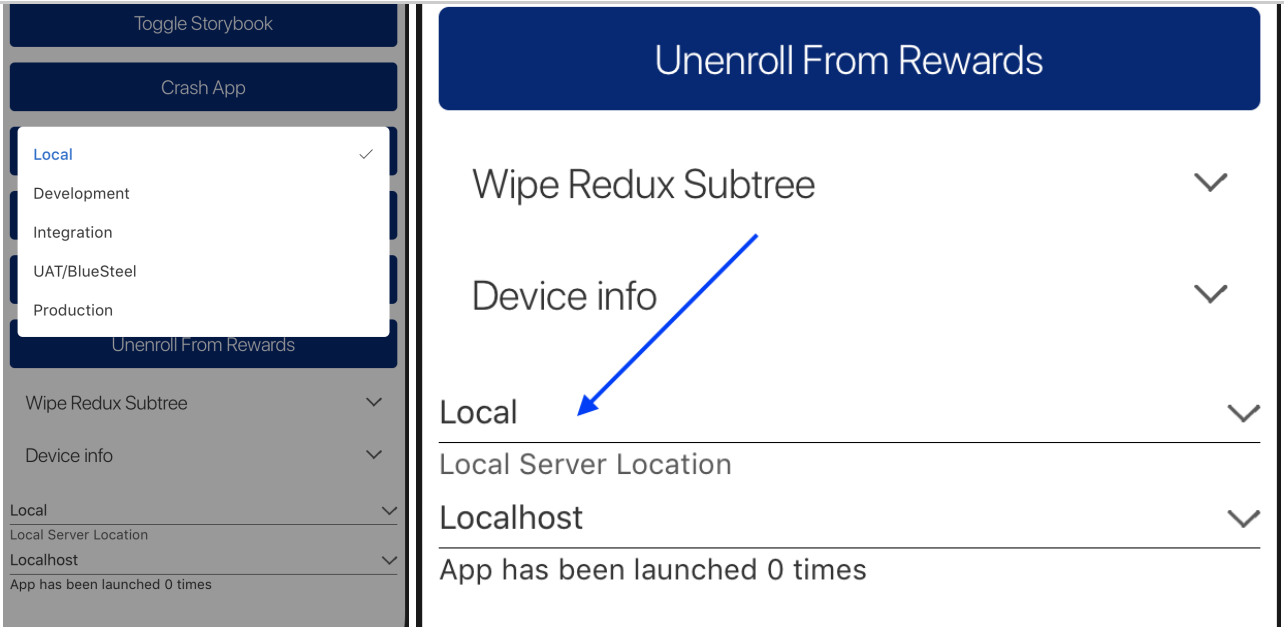
If you need to install an APK for TalkBack, try the x86_64 images

---

1. Install Node (LTS v12)
2. Set up Artifactory as proxy
    a. Additionally, export these environment variables from ~/.bash_profile (or similar): `ARTIFACTORY_CREDENTIALS_USR` with your artifactory username, `ARTIFACTORY_CREDENTIALS_PSW` with your api key (`source` the file or restart your terminal after adding them)
3. (optional) install homebrew
4. Install yarn: `sudo npm install -g yarn` or `brew install yarn --without-node`
5. Follow the React Native Getting Started guide.
    a. IMPORTANT: In environment-setup, make sure to follow the "React Native CLI Quickstart" section, not the "Expo CLI Quickstart". Select "Android" (or iOS project - whatever you are building) as the target OS.
        i. When prompted to install the JDK, choose Java 8
    b. stop when you get to the "Creating a new application" section - we've already got one!
6. Install Xcode from the App Store.
    a. Once installation is complete, go to Preferences > Locations. In the Command Line Tools dropdown, select Xcode X.X.X
7. Install cocoapods: `brew install cocoapods` (Or use `sudo gem install cocoapods` if you don't have homebrew)
8. Clone the uhc-react-native repo `git clone git@github.com:AudaxHealthInc/uhc-react-native.git.`
9. You will also need to make sure you have setup a personal access token in Github for SSO auth
    a. https://help.github.com/en/github/authenticating-to-github/creating-a-personal-access-token-for-the-command-line
    b. GitHub Setup - Wiki
    c. Troubleshooting:
        i. Sometimes your pod install might fail due to a repository branch version issue within pods. In order to remedy this a possible solution is to remove your ~/.cocoapods folder and then re-run pod install in the codebase packages/arcade/ios folder.
10. And setup access through SSH keys, if you haven't already for some of the internal repos we are dependent on (https://help.github.com/en/enterprise/2.15/user/articles/adding-a-new-ssh-key-to-your-github-account)
    a. You may need to set git to use SSH instead of HTTPS, use this command:

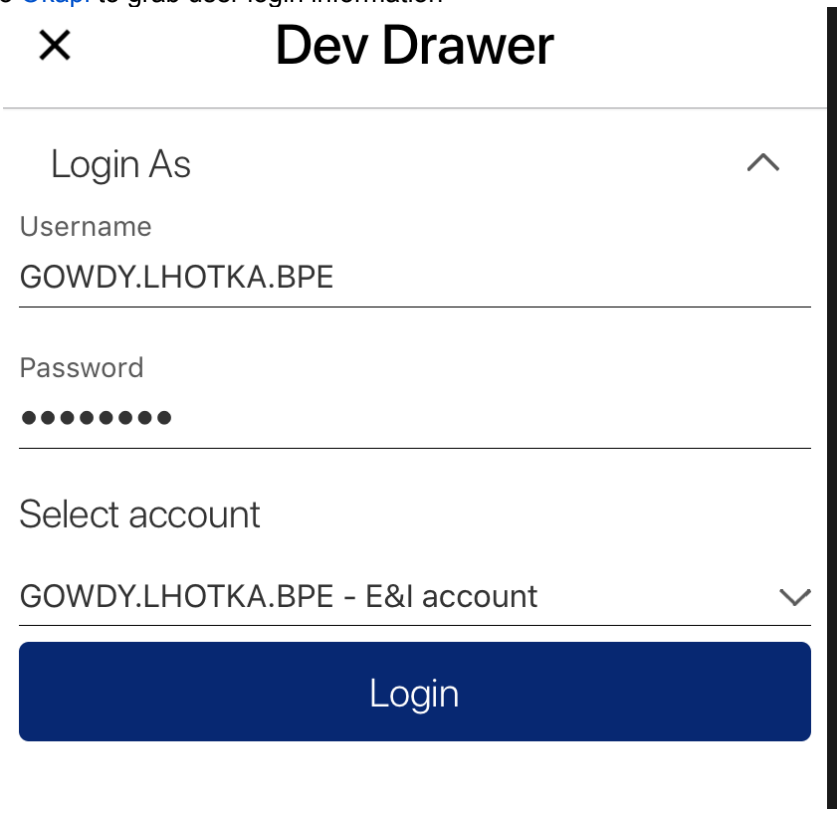
```
git config --global url.ssh://git@github.com/.insteadOf https://github.com/
```

11. In the root directory, restore packages: `yarn install` or `yarn` (the two commands are equivalent and interchangeable). At the end of the install, `scripts/post-install.sh` will run automatically to handle patching & pods.
    a. There are times where `pod install` may fail and you'll have to run it manually (`cd packages/arcade/ios && pod install`)
12. Run the tests to make sure everything's set up correctly: `yarn test`
    a. If the test runner starts up but appears to be hanging before any of the tests actually start running, you may need to uninstall watchman and reinstall it
13. Start the javascript bundler: `yarn start` (or `yarn run start` ; the two commands are equivalent)
    a. This will start a local server which will provide the compiled iOS or Android app with dynamic javascript
14. Compile the app: `yarn a or yarn i`
    a. The majority of UHC Mobile code is dynamic javascript which can be hot-reloaded, so you should only need to recompile the app for dependency updates or native code changes.
    b. Have an Android device or emulator connected to adb before running yarn a
        i. On Android, open the dev menu by pressing cmd + M, or by running `adb shell input keyevent 82` in a terminal
    c. An iOS sim will start automatically when running yarn i
        i. On iOS, open the dev menu by clicking thru Hardware → Shake Gesture or pressing cmd + D
15. Triple click the UHC logo to open the dev drawer.



a.
b. Here you can select which environment to connect to (probably development). You can also develop locally without a tenant by selecting "Local" and starting up the mock server with **(cd packages/arcade && yarn server)**
c. Leaving the tenant blank will assign a default when you try to login.

16. Head to Okapi to grab user login information



a.

Troubleshooting

When in doubt, `yarn clean && yarn` . Clean early, clean often!

To troubleshoot issues with your react-native setup, `yarn react-native doctor` from the directory packages/arcade (NB: Facebook says this command is still experimental)

Request Tracking

For local development, it's often helpful to track web requests for development and debugging. Reactotron is particularly helpful for React Native apps as it allows you to see both web requests and Redux activity.

Installation:

1. `brew update`
2. `brew cask install reactotron`

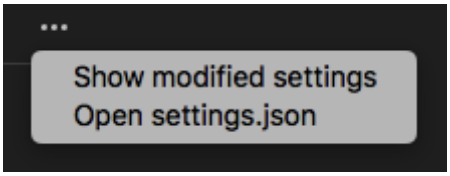
Usage (Android):

1. Launch "Reactotron" from Applications
2. Launch your Android Emulator
3. `Run: adb reverse tcp:9090 tcp:9090`
4. If Reactotron displays "Waiting for connection", relaunch the UHC Debug application (R + R while emulator is focused)
5. Use the application and note requests being logged

Recommended Editor Setup

VS Code is recommended for React Native development because of the strong tooling support.

1. Install VS Code
2. Open the root of the project in VS Code: `cd uhc-react-native && code .`
3. Install the following extensions:
    a. React Native Tools
    b. Flow Language Support
    c. ESLint
    d. Prettier
    e. Path Intellisense
    f. GitLens
4. To enable flow support in VS Code
    a. make sure you have installed dependencies first via `yarn install` in project's folder
    b. open VS Code preferences and edit user's settings. Enter Command Palette (⇧⌘P) and type `Preferences: Open User Settings`, hit menu and press "Open settings.json". If you can't find "Open settings.json" then from the Command Palette, search for `Open Settings (JSON)` and update the settings.json. See (4.b.i) for the json reference.



        i. Under USER SETTINGS tab add following

```
{
    "flow.useNPMPackagedFlow": true,
    "flow.pathToFlow": "flow",
    "editor.formatOnSave": true,
    "javascript.validate.enable": false,
    "javascript.format.enable": false,
    "eslint.autoFixOnSave": true
}
```

5. Restart your editor

- React Native development tips

No labels

1 Comment

Mike Young

The Artifactory setup method that worked for me personally was running the shell command `curl -u USERNAME:APIKEY https://artifacts.werally.in/artifactory/api/npm/auth` and then copying the output + `email = MYEMAIL` into `~/.npmrc`