# Sovrin Connector App Documentation

Sovrin Foundation

Feb 05, 2019

# Contents

This is an early-stage project to create a functional open source mobile application that works as an edge agent in the Sovrin ecosystem.

The best place to understand more about the Connector app is in our introduction

As the project progresses, we hope that this documentation will serve both users and contributors of the Connector app. At this stage, this project is needs some more initial development work before it is ready to use.

We welcome any contributors to the project. Please visit our #connector-app rocket.chat channel on the sovrin rocket.chat forum to meet the people working on the project, and review the documentation below to familiarize yourself with our contribution guidelines.

CHAPTER 1

# Sovrin Connector App

App to connect Sovrin with 3rd party authentication

> note: While this is currently just a mirror of the README.md, we will want to edit this to have more user/consumer oriented instructions

## 1.1 Introduction

We have an ongoing effort to build and run the Connector App, then make it interoperable with the hyperledger Indy open source protocols.

- Building the Connector App Q&A Video

## 1.2 Pre requisite to run

- Mac machine
- XCode 9
- Node 8+. Preferred way to install node is via nvm
- React native setup. Use tab `Building Projects with Native Code`.
- Ruby
- Make sure `pod` (1.5.3) is installed or run `sudo gem install cocoapods -v 1.5.3`
- Android Studio 3+

## 1.3 Steps to run

- Clone this repository with `SSH`

- `yarn` or `yarn install`
- `yarn start`

### 1.3.1 Run on ios simulator

- `yarn pod:dev:install`
- `yarn react-native run-ios`

### 1.3.2 Run ios on device

- Do not use XCode automatic code signing
- `cd ios/fastlane`
- `sudo gem install bundle`
- `bundle install`
- Make sure you get added to the connectme-callcenter-certs repo so that the following command is successful – git clone 'git@github.com:evernym/connectme-callcenter-certs.git' '/var/folders/dt/sk594jpn40d0097bpg17gwc40000gn/T/d20180705-10510-lw9oue'
- To get the development release certificates do `bundle exec fastlane match development`. DO NOT use `--force` with this command.
- You'll be prompted to enter 2 passwords. Slack a contributor for credentials
- Open Xcode, select your device and run

### 1.3.3 Run on Android simulator

#### Manual steps

- Make sure a simulator is already created. Otherwise create one from Android studio
- `cd android/keystores && keytool -genkey -v -keystore debug.keystore -storepass android -alias androiddebugkey -keypass android -keyalg RSA -keysize 2048 -validity 10000`
- `yarn react-native run-android`

#### By using scripts

- Run `android/scripts/generate.key.sh` one time to generate keystore
- Run `android/scripts/android.build.sh` to build android apk

### 1.3.4 Create a release build for ios

- `yarn pod:install`
- To get the beta release certificates do `bundle exec fastlane match adhoc`
- You'll be prompted to enter 2 passwords. Slack a contributor for what those are.

- Make sure that the ios/ConnectMe/Info.plist, ios/ConnectMeTests/Info.plist, ios/ConnectMe-tvOS/Info.plist, ios/ConnectMe-tvOSTests/Info.plist, have their CFBundleVersion set to the NEXT build number
- Make sure that the ios/ConnectMe.xcodeproj/project.pbxproj has it's 2 CURRENT_PROJECT_VERSION attributes set to the NEXT build number
- To get releasable .ipa launch Xcode and open the ios/ConnectMe.xcworkspace project
- Select the "Generic iOS Device"
- Then run Product -> Archive
- After it is done then login to hockeyapp.net and click on QA ConnectMe and then click add version button and upload it

### 1.3.5 For android local Release build

- add my_keystore.jks file to ~/keystores folder If you get this error during the `bitrise run android` build then you are missing the my_keystore.jks file What went wrong: Execution failed for task ':app:validateSigningRelease'.

  Keystore file /Users/norm/keystores/my_keystore.jks not found for signing config 'release'.
- ask your team members for .bitrise.secrets.yml file and place it in directory with bitrise.yml file
- run `brew install bitrise && bitrise setup`
- Make sure that the android/app/build.gradle has it's versionCode attribute set to the PREVIOUS build number (be aware that a failed "bitrise run android" will STILL increment the number and so you will need to decrement it back to the PREVIOUS build number after a failure)
- run `bitrise run android`
- If you get this error then rm -rf node_modules and re-run `bitrise run android` What went wrong: Failed to capture snapshot of input files for task ':app:bundleReleaseJsAndAssets' property '$1' during up-to-date check.

  Could not list contents of '/Users/norm/forge/work/code/evernym/ConnectMe/node_modules/jest-runtime/node_modules/babel-core/node_modules/.bin/babylon'. Couldn't follow symbolic link.
- run `brew install maven`
- download the vcx.aar file
- later run `mvn install:install-file -Dfile=com.evernym-vcx_1.0.0-12-06-2018T16-17_x86-armv7-release.aar -DgroupId=com.evernym -DartifactId=vcx -Dversion=1.0.0-12-06-2018T16-17 -Dpackaging=aar`
- remember to change file names as required in the above mvn install command

## 1.4 To Read

- Coding guidelines
- Contributing guidelines

### 1.4.1 Tech stack used

- React Native
- React Navigation
- Redux
- Redux Saga
- Flow
- Jest
- Yarn
- Cocoa pods

### 1.4.2 Run functional automated test

- Ensure that Node version greater than 8.5.0 and less than 9 is installed. We recommend using `nvm` to install node
- `$ brew tap wix/brew && brew install applesimutils`
- `$ npm i -g detox-cli`
- Ensure that iphone 7 simulator is installed and running
- Ensure that iphone 7 simulator has hardware keyboard disconnected. `Hardware -> Keyboard -> Un check 'Connect Hardware Keyboard'`
- `$ npm run test:e2e:build && npm run test:e2e`

  If you get an error in logs saying that `image not found.` Run `$ detox clean-framework-cache && detox build-framework-cache`, then run last command in above steps

## 1.5 #IDE

- You may use any IDE you feel more comfortable with.
- Our preferred IDE would be “VS Code” with extensions like
  - Prettier - Code formatter (esbenp.prettier-vscode)
  - VS Code ES7 React/Redux/React-Native/JS snippets
  - Code Spell Checker (streetsidesoftware.code-spell-checker)
  - Better Comments (aaron-bond.better-comments)
  - Path Autocomplete (ionutvmi.path-autocomplete)

## 1.6 Makefile

- If you want to run the iOS or android emulators from a terminal without the need for the Xcode or Android Studio IDE then you can use the Makefile (this is how Norman runs the iOS and android emulator on his MacBook). The commands to setup your environment for the Makefile are:

- export PATH=$PWD/node_modules/.bin:$PATH
- make clean
- make pre-run
- For iOS: SIMULATOR="iPhone 7" make run-ios
- For Android: Start the android emulator in a separate terminal with
- RUST_BACKTRACE=1 /Users/norm/Library/Android/sdk/emulator/emulator -writable-system -avd Pixel_API_26 -netdelay none -netspeed full
- Then run the app in a separate terminal with: make run-android or VARIANT=release make run-android or VARIANT=debug make run-android

## 1.7 Things to improve

- [ ] Hydration needs to replace whole store. As of now it just changes few values in config store
- [ ] Need to save whole store in keychain and implement proper hydration
- [ ] Show loader when user accept/reject invitation
- [ ] Need to consider scenario if user has not allowed permission for push notification or user disable the permission
- [ ] We need to communicate to user on why we need push notification permission
- [ ] Connections key should be remote DID and not user identifier
- [ ] Remove passing newConnection while saving new connection
- [ ] Save whole payload in connection instead of choosing selected props in connection store

## 1.8 Frequently Encountered Problems (FEP)

### 1.8.1 Unit test

- *Problem*: `Invariant Violation: Native module cannot be null.`. *Solution*: You are missing the mocks for native module. Open `<rootDir>/__mocks__/setup.js` and create a mock for native module which is throwing error. You can check which native module is throwing error from call stack printed in terminal

### 1.8.2 iOS Simulator keyboard issue

- *Problem*" `The iOS simulator does not take input from my MacBook Pro keyboard.` *Workaround*: A temporary workaround is to disconnect the hardward keyboard with the Shift+Cmd+K key combination (via the menu it is Hardware -> Keyboard -> Connect Hardware Keyboard to unselect that option). Then only using the menu Hardware -> Shake Gesture will bring up the React Native Developer Menu and then you select the Reload option from the React Native Developer Menu and then the software keyboard will come up and allow you to use the mouse to input characters. After a while you can try to re-enable The MacBook Pro keyboard but if it still fails then use this workaround again.

### 1.8.3 iOS build issue

- *Problem*: `third-party/glog-0.3.4/src/base/mutex.h 'config.h' file not found`. *Solution*: https://github.com/facebook/react-native/issues/16097. Basically from the ConnectMe toplevel source code directory do 1) cd node_modules/react-native/third-party/glog-0.3.4/ && ../../scripts/ios-configure-glog.sh

## 1.9 How to Add Documentation

For new features and pull requests, maintainers should make sure that the **contributor has added an explanation for their changes in the docs folder before merging the PR.**

Contributors should write an addition to a current file or add a new file to the docs/source/ folder that explains what their feature is and how it works. If needed, they may also add a link to more technical README's located nearer to the code.

Whenever additions are made to the docs, make sure to update the `index.rst` in whichever folder the file has been added, and build the docs locally to confirm they work (TODO: add the `sphinx-build` command to our CI/CD flow).

For example, if I wanted to add another file to the indy-sdk docs/ folder named `glossary.md`, I would create the file, and then add a reference to it in the `index.rst`:

```
.. toctree::
  :maxdepth: 1
  :hidden:

  getting-started/index.rst
  ...
  other files
  ...
  glossary.md                   .. <-- this is your new file!
```

To add a new file to a subfolder, simply update the subfolder's `index.rst` with the relative link to your file.

If you'd like to link to a file outside of the docs/ folder, you'll need to provide an external github link (this is by design, to keep our docs organized into a single folder)

## 1.10 Building the docs on your machine

Here are the quick steps to achieve this on a local machine without depending on ReadTheDocs. Note: Instructions may differ depending on your OS. Run these commands within the repository folder

```
pip install Sphinx
pip install sphinx_rtd_theme
pip install recommonmark==0.4.0
cd docs/source # Be in this directory. Makefile sits there.
make html
```

This will generate all the html files in `docs/source/_build/html` which you can then browse locally in your browser. Every time you make a change to the documentation you will need to rerun `make html`.

## 1.11 Additional Instructions

This section is to be used for repo maintainers to add additional documentation guidelines or instructions.

# CHAPTER 2

# Coding Guidelines

## 2.1 Comments

- Comments should describe *WHY* and *NOT WHAT*

```
// BAD
// delete connections
yield call(deleteItem, CONNECTIONS)

// GOOD
// Since this is a new installation of app,
// we don't previously stored connections, so go ahead and delete previously stored␣
↪connections
yield call(deleteItem, CONNECTIONS)
```

### 2.1.1 Be thorough

```
- Imagine the next person landing in your code to have no context whatsoever and VERY␣
↪limited time to change it.
- Explain without the assumption that people already know how your code works
- Explain the hacks you needed to do and why they work
- Explain the internal interdependencies that aren't explicit (eg other systems␣
↪relying on structure or api)
- Be ok w/ writing a longer paragraph whenever needed
```

## 2.2 TDD

- Write tests for everything you create
- Tests should be co-located with feature/component inside `__tests__` folder

- Test filenames should end with `.spec.js`
- A snapshot test should be present for each component

## 2.3 Naming conventions

- NO SHORT NAMES are allowed. Be as explicit as you can be

```
// BAD
const a, req, txt, conReq, res, d,

// GOOD
const invitationRequest, connectionRequest, authenticationRequest
```

## 2.4 Types

- Add flow types for every new file that you create
- Good if you add flow types to existing files as well
- New files should have `// @flow` at the top of the file and must adhere to flow types and fix flow errors
- Do not use `any` for flow types

## 2.5 UI components

- Use common layout components for almost every layout need
- Try to use as many common components as possible
- No inline style
- *Note:* This is a work in progress guidelines and will keep on improving as we move along

# CHAPTER 3

# Contributing Guidelines

## 3.1 Steps to contribute

- Create branch for each feature
- Follow TDD for each component/store
- Use flow to define types
- DO NOT use `npm` to install packages, not even `npm 5`. Always use `yarn` to install dependencies
- Tests are run by default when you try to push. If tests fail then code will not be pushed, not even to local branch.
- Push will fail even if there are any lint errors
- Raise pull request to merge any change to master
- Fill Pull request template properly while raising the PR

## 3.2 Process

- PR template is made for submitting PR
- PR is `squashed and merged`
- Tests are run before each push
- Files are automatically formatted by prettier before committing
- Hooks for pre-commit and pre-push are in `package.json`

### 3.2.1 How to test screens without any external dependency

#### Create connection without generating build and clicking deep link

- Go to config store and find line with something like this `...baseUrls[SERVER_ENVIRONMENT.STAGING]`. Change `STAGING` with `DEVELOPMENT` or `SANDBOX`.
- In deep-link/index.js. Find method `onDeepLinkData` and add this line at the start of method

```
this.props.deepLinkData('6a3e8fd0')
```

The string that we have passed inside `deepLinkData` comes from email. Contact other developers for email account access.

#### Proof request screen

Add these lines to import statement of Home screen

```
// TODO Remove these lines after testing is done
import { proofRequest, proofRequestId } from '../../__mocks__/static-data'
import { proofRequestRoute } from '../common/route-constants'
import { proofRequestReceived } from '../proof-request/proof-request-store'
```

Add this method to Home screen component

```
  // TODO Remove this whole function, this is for testing purpose
  componentDidMount() {
    // $FlowFixMe
    this.props.proofRequestReceived(
      proofRequest.payload,
      proofRequest.payloadInfo
    )
    setTimeout(() => {
      this.props.navigation.navigate(proofRequestRoute, { uid: proofRequestId })
    }, 1000)
  }
```

Bind methods to Home props

```
// TODO: Remove below two line, Only for testing purpose
const mapDispatchToProps = dispatch =>
  bindActionCreators(
    {
      proofRequestReceived,
    },
    dispatch
  )

export default connect(mapStateToProps, mapDispatchToProps)(DashboardScreen)

// TODO: Un-comment this line after testing is done
//export default connect(mapStateToProps)(DashboardScreen)
```

Go to `mapStateToProps` in proof-request.js and find logoUrl

```
// TODO: Un-comment this line and remove line next to this line
// logoUrl: getConnectionLogoUrl(state, remotePairwiseDID),
logoUrl: 'https://image.flaticon.com/icons/png/128/174/174851.png',
```

In proof-store.js. Import static-data for proof

```
// TODO Remove below import, only for testing
import { proofWith10Attributes,homeAddressPreparedProofWithMissingAttribute } from '..
↪/../__mocks__/static-data'
```

Find the call to `prepareProof` in `generateProofSaga`. Comment out the actual call to native bridge and use this instead

```
const preparedProofJson=JSON.stringify(homeAddressPreparedProofWithMissingAttribute)
```

Find the call to `generateProof` in `generateProofSaga`. Comment out the actual call to native bridge and use this instead

```
// TODO Remove below line and un-comment above line, only for testing
const proofJson = JSON.stringify(proofWith10Attributes)
```

### Paid Claim Offer Accepted screen

Add these lines to import statement of Home screen

```
// TODO Remove these lines after testing is done
import {claimOfferId,paidClaimOffer} from '../../__mocks__/static-data'
import { claimOfferRoute } from '../common/route-constants'
import {claimOfferReceived} from '../claim-offer/claim-offer-store'
```

Add this method to Home screen component

```
  // TODO Remove this whole function, this is for testing purpose
   componentDidMount() {
    // $FlowFixMe
   this.props.claimOfferReceived(paidClaimOffer.payload,paidClaimOffer.payloadInfo)
   setTimeout(() => {
     this.props.navigation.navigate(claimOfferRoute, { uid: claimOfferId})
   }, 4000)
  }
```

Follow the process mentioned below in Claim offer screen.

### Claim Offer Accepted screen

Add these lines to import statement of Home screen

```
// TODO Remove these lines after testing is done
import {claimOfferId,claimOffer} from '../../__mocks__/static-data'
import { claimOfferRoute } from '../common/route-constants'
import {claimOfferReceived} from '../claim-offer/claim-offer-store'
```

Add this method to Home screen component

```
  // TODO Remove this whole function, this is for testing purpose
   componentDidMount() {
    // $FlowFixMe
   this.props.claimOfferReceived(claimOffer.payload,claimOffer.payloadInfo)
   setTimeout(() => {
     this.props.navigation.navigate(claimOfferRoute, { uid: claimOfferId})
   }, 4000)
  }
```

Bind methods to Home props

```
//TODO remove this, only for testing purpose
const mapDispatchToProps = dispatch =>
  bindActionCreators(
    {
      claimOfferReceived,
    },
    dispatch
  )

export default connect(mapStateToProps, mapDispatchToProps)(DashboardScreen)
//export default connect(mapStateToProps)(DashboardScreen)
```

Go to `mapStateToProps` in claim-offer.js and find logoUrl

```
 // TODO: Un-comment this line and remove line next to this line
  //const logoUrl = getConnectionLogoUrl(state, claimOfferData.remotePairwiseDID)
  const logoUrl ='https://image.flaticon.com/icons/png/128/174/174851.png'
```

In claim-offer-store.js

Go to claimOfferAcceptedSaga use hardcoded userPairwiseDid and parsedClaimRequest. Also Comment out the sendMessage native bridge call

```
// TODO use hardcoded userPairwiseDid, only for testing
// const userPairwiseDid: string | null = yield select(
  //   getUserPairwiseDid,
  //   remoteDid
  // )
  const userPairwiseDid: string | null = 'userPairwiseDid'

   //TODO uncomment these lines and remove hardcoded parsedClaimRequest
      // const stringifiedClaimRequest: string = yield call(
      //   generateClaimRequest,
      //   remoteDid,
      //   indyClaimOffer,
      //   poolConfig
      // )
      // TODO:KS Add error handling if claim request parse fails
      // const parsedClaimRequest: IndyClaimRequest = JSON.parse(
      //   stringifiedClaimRequest
      // )

      const parsedClaimRequest: IndyClaimRequest = {
        blinded_ms: {
          prover_did: 'prover_did',
          u: 'u',
        },
```

(continues on next page)

(continued from previous page)

```
        issuer_did: 'issuer_did',
        schema_seq_no: 36,
      }
```

### To trigger claim offer success pop up

in claim-request-modal.js add the following code

```
    componentDidMount() {
    // if modal is opened when it was unmounted or freshly mounted
    // this.toggleModal(this.props.claimRequestStatus)

    setTimeout(() => {
      this.setState({isVisible: true})
    }, 4000)
  }
```

### Test credentials and proofs on simulator

In componentDidMount of push-notification.js. Add below code at end

```
this.props.fetchAdditionalData({
      // forDID can be extracted from redux logs inside connection reducer, it is␣
↪named as `identifier`
      forDID: 'Mua2EG3rZphtwj88LWRpcw',
      // uid can get from network request log in verity-ui and checking the response␣
↪for that particular credential or proof request
      uid: 'njaxymv',
      // types supported are credOffer cred, proofReq
      type: 'credOffer',
      // can be extracted from same place as forDID, key named as senderDID
      remotePairwiseDID: '4yKr2YScm8o7nRXRvbHvzA',
      // can be kept the same
      senderLogoUrl: 'https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQEc_
↪lUYZc74j-ArO9tldKiyLusNRGQE1X4dBR8Yz-J1ZcrAduLYg',
    })
```

## 3.2.2 How to enable Android RUST logs

- un-comment this line in android/app/build.gradle

```
// compile group: 'com.mobidevelop.robovm', name: 'robovm-rt', version: '2.3.4'
```

- un-comment these line in `MainActivity.java`

```
//import libcore.io.ErrnoException;
//import libcore.io.Libcore;
...
//        try {
//            Libcore.os.setenv("RUST_LOG", "TRACE", true);
//        } catch (ErrnoException e) {
```

(continues on next page)

(continued from previous page)

```
//            e.printStackTrace();
//        }
```

### 3.2.3 How to enable RUST logs in ios

- Open connectme workspace in xcode
- From Menu Product -> Scheme -> Scroll down to bottom -> Edit Scheme
- A window will open. On left side of this window, select “Run”, on right side click on “Arguments”
- Expand “Environment Variables”
- Check checkboxes for RUST_LOG and RUST_BACKTRACE

## 3.3 How to know what tests to write

Check this small video https://drive.google.com/open?id=1t83ZTe4RdgozuIRs6Pi140fU7cIp-8wx

# CHAPTER 4

# Pull Request Template

## 4.1 Reason for Pull Request

- [ ] Bug fix
- [ ] Feature/Change request

## 4.2 Link for jira ticket

CO-

## 4.3 In case of Bug

### 4.3.1 Root cause

- Backend API change
- Requirement was not clear

## 4.4 In case of Feature/Change request

### 4.4.1 High level description of changes done

- Added a store to save user response for push notification and then use it while onboarding user. We can’t start onboarding process until we get push notification token from user

## 4.5 Tests written for Bug/Feature/Refactoring

- `<Please write test file/test name here>`

## 4.6 Checklist

- [ ] I have created new screen, and I have checked the header of new screen on Android. I have also set the status bar color in app.js
- [ ] I have checked the back button functionality required for the screens in Android. I have added the routeNames that are need to be handled for the back button in app.js
- [ ] I have checked the back behaviour by swiping left or swiping down on screens and they behave as expected. Back behaviour of these screens has been approved by product team

# CHAPTER 5

# Test Recipes

## 5.1 Mock, already globally mocked native module

Let's say we have mocked `react-native-touch-id` in setup.js. This is how we mocked it in setup.js

```
jest.mock('react-native-touch-id', () => {
  return {
    authenticate: jest.fn(message => Promise.resolve()),
  }
})
```

If you notice here, we are always resolving module. If we run our test using `TouchId` module, then we can't test error conditions.

To mock this implementation again which rejects this promise and allow us to test error condition. We can do something as follows:

```
import TouchId from 'react-native-touch-id'
...
it('test error condition', async () => {
  TouchId.authenticate.mockImplementation(_ => Promise.reject())
  // authenticate is internally using TouchId, we test component interface
  await component.authenticate()
  // make some assertions
})
```