Appium mobile test automation

for Google Android and Apple iOS

Last updated: 2 November 2021

© Pepgo Pty Ltd, Level 18, 333 Ann Street, Brisbane, QLD 4000, Australia

Contents

About this document 3
Appium 3
Architecture 3
Enable developer options on an Android device 4
Install Android Studio and SDK 5
Install Appium: 5
Run Appium (example) 5
Create an Android virtual device 6
Installing an app on a virtual device 6
Finding element properties 7
The "Appium Inspector" 7
Using the Appium Inspector 8
The "uiautomatorviewer" 10
The Developer Assistant app 11
A full Appium Java example 11
What the PasswdSafe example does 11
Using the PasswdSafe example 12
Project structure (in IntelliJ IDEA) 13

About this document

This document explains the test automation tool Appium for mobile devices. Appium is powered by Selenium WebDriver, and the same Appium program code can be used for automating both Google Android and Apple iOS platforms.

This document contains a Java example for Google Android as target platform. It uses an Android native app called "PasswdSafe".

Appium

Appium is an open source test automation tool for mobile applications.

Appium also allows you to run automated tests on actual devices, emulators and simulators.

Today when almost every mobile app is available for both Apple iOS and Google Android, you need a tool, which allows testing cross platforms. Having two different frameworks for the same app would both increase the cost of testing and the time to maintain it as well.

The basic philosophy of Appium is that you should be able to reuse code between Apple iOS and Google Android, and that's why the API's are the same across Apple iOS and Google Android. Another important point to highlight is that Appium doesn't modify your app, or need you to recompile the app, or have access to its source code, although having access to the source code is definitely an advantage.

Appium lets you choose the language you want to write your test in. It doesn't dictate the language or framework to be used.

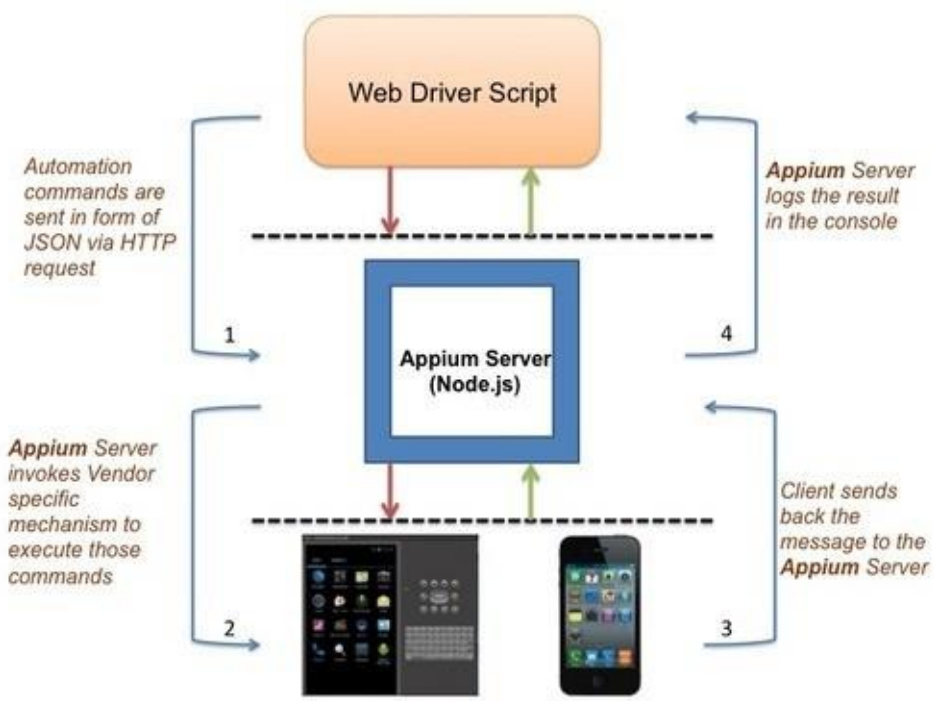
Architecture

When you download Appium, you are basically downloading the server. The server is written in Node.js and implements Selenium WebDriver. It allows you to use Selenium WebDriver clients to drive your tests. Your mobile app acts precisely like a web app, where the DOM is represented by the View hierarchy.

The Appium server exposes a REST API, which performs the following actions:

1. Receives connection from client
2. Listen command
3. Execute command
4. Respond back the command execution status

In terms of an architecture diagram, this is how Appium can be explained:



Enable developer options on an Android device

This article contains helpful background information:
http://www.greenbot.com/article/2457986/how-to-enable-developer-options-on-your-android-phone-or-tablet.html

1. On stock Android devices, go to: "Settings > About phone > Build number".
2. Once you've found the build number section of the settings, tap on the section 7 times. After two taps, a small pop up notification should appear saying "you are now X steps away from being a developer" with a number that counts down with every additional tap. After the 7th tap, the developer options will be unlocked and become available. They can usually be found in the main settings menu.
3. Switch on "Stay awake".
4. Switch on "USB debugging".

To check if the (real or virtual) device can be used, type from command prompt (after you have correctly installed the Android SDK, as described in the next section):

```
adb devices
```

The output should look like this:

```
List of devices attached
00c600c600c600c6        device
```

If a real device is not recognised, then check if the USB driver is correctly installed:
https://developer.android.com/studio/run/oem-usb

Install Android Studio and SDK

Android Studio contains the Android SDK. It is recommended to install Android Studio, although you can also just install the Android SDK.

Android Studio is available from https://developer.android.com/studio . Follow the setup wizard to install Android Studio and the SDK.

Install Appium:

Appium can be downloaded from http://appium.io/downloads.html .

Run Appium (example)

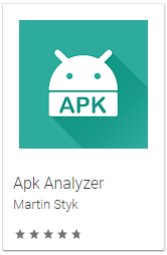
The example in this document uses the open source "PasswdSafe" app (https://www.pwsafe.org/). The app is available for free from the "Google Play" app store. The source code and the "`*.apk`" file for the app are available from https://sourceforge.net/projects/passwdsafe/ (just click on the "Download" link).

The Android app file ("`*.apk`") must be set in the "Application Path" of the Appium settings. However, please note that some versions of Appium have problems with paths that contain spaces, so avoid that if possible.

Newer versions of Appium will find the package name and (starting) activity automatically, but if not, then theses can also be found from command line with the command:
`aapt dump badging <PasswdSafe>.apk`

Note that "`aapt`" is part of the Android SDK "`build-tools`" folder (in the relevant Android SDK version subfolder). If the command is not recognised, then you might have to add the folder to the "path" of your operating system.

An excellent free tool to find the package name and the activities is the free "Apk Analyzer" app from the Google Play store:



Create an Android virtual device

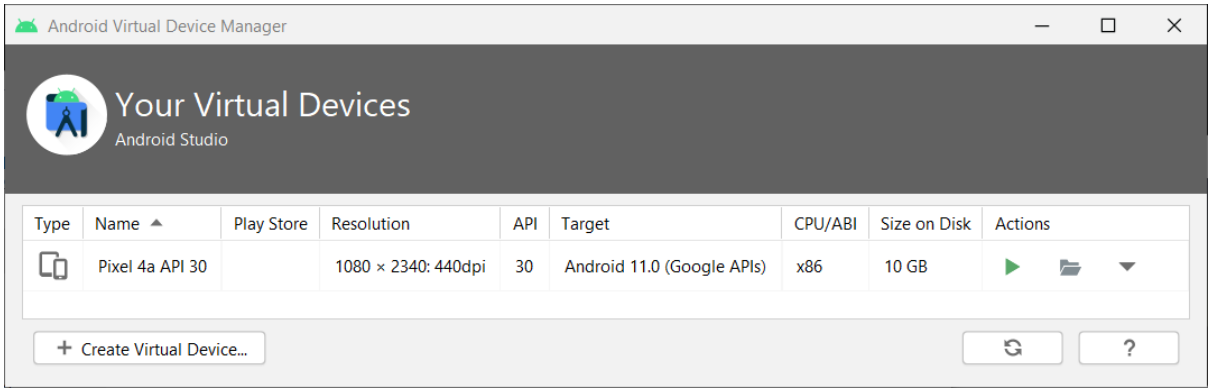
Many developers prefer working with real devices, as the Google virtual devices are often slow and sometimes painful to create and use. However, it is still often required to create and use virtual devices, particularly to cover the wide range of Android devices both with respect to device hardware and screen sizes, as well as Android (API) versions.

You first need to open the Android Virtual Device Manager. You can do this from Android Studio, or from a command prompt with:

```
android avd
```

Then you need to create an Android virtual device. The ideal settings for the device depend on the host machine.

You can start virtual devices right from the Android Virtual Device Manager by selecting the device and clicking the "Start…" button.



As you already know, you can check the name(s) of the running virtual and real devices from command line:

```
adb devices
```

Installing an app on a virtual device

Once your virtual device is ready, you can install an app on the virtual device.

The first step is to run the virtual device, on which you want to install the app. You can start a virtual device from the "AVD Manager" by clicking the "Start…" button, or from the command line:

```
emulator -avd <AVD_Name>
```

This launches the virtual device and the command is running in the command prompt window.

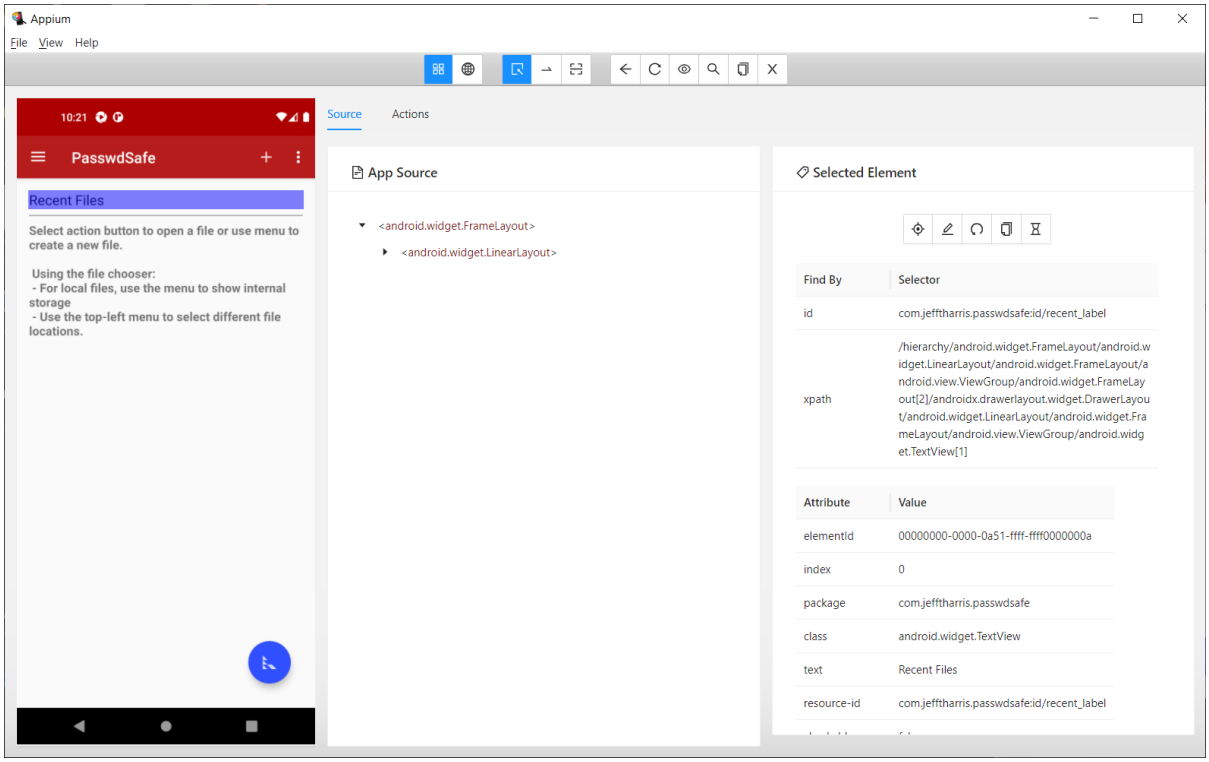
Launch another command prompt window and run the following command (the parameter "`-e`" is for specifically addressing the emulator, for addressing a real device on a USB connection, the corresponding parameter is "`-d`").

```
adb -e install <path_to_apk_file>
```

Finding element properties

The "Appium Inspector"

The Appium Inspector is a GUI inspector for mobile apps and more, powered by a (separately installed) Appium server. When you're using it to inspect a mobile app, it looks like this:

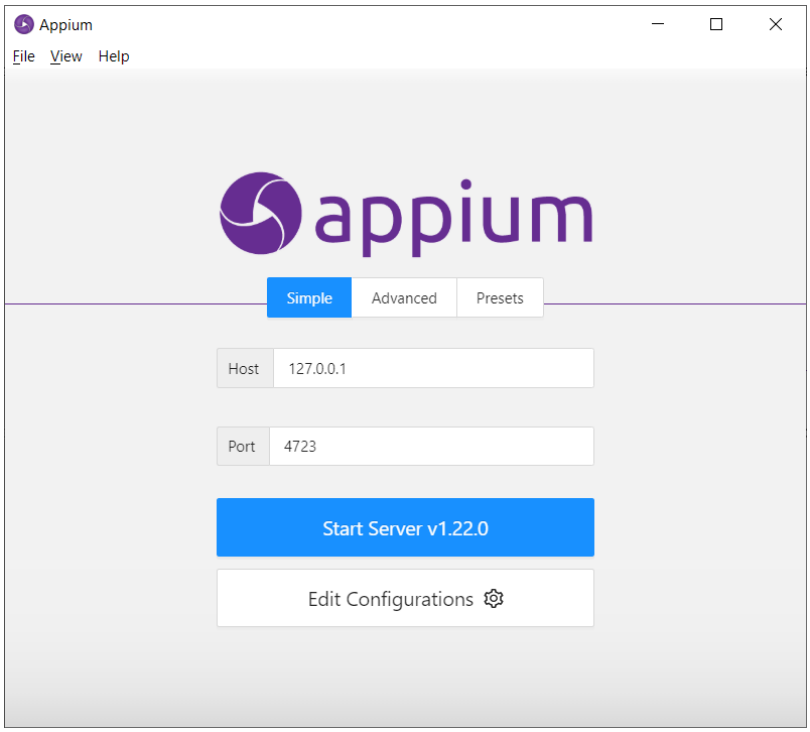


Appium Inspector is an open source app for Apple macOS, Microsoft Windows, and Linux that presents the power of the Appium automation server in a beautiful and flexible User Interface. Being a graphical interface for the Appium server, it is an excellent tool for finding locators.

The Appium Inspector is available from: https://github.com/appium/appium-inspector .

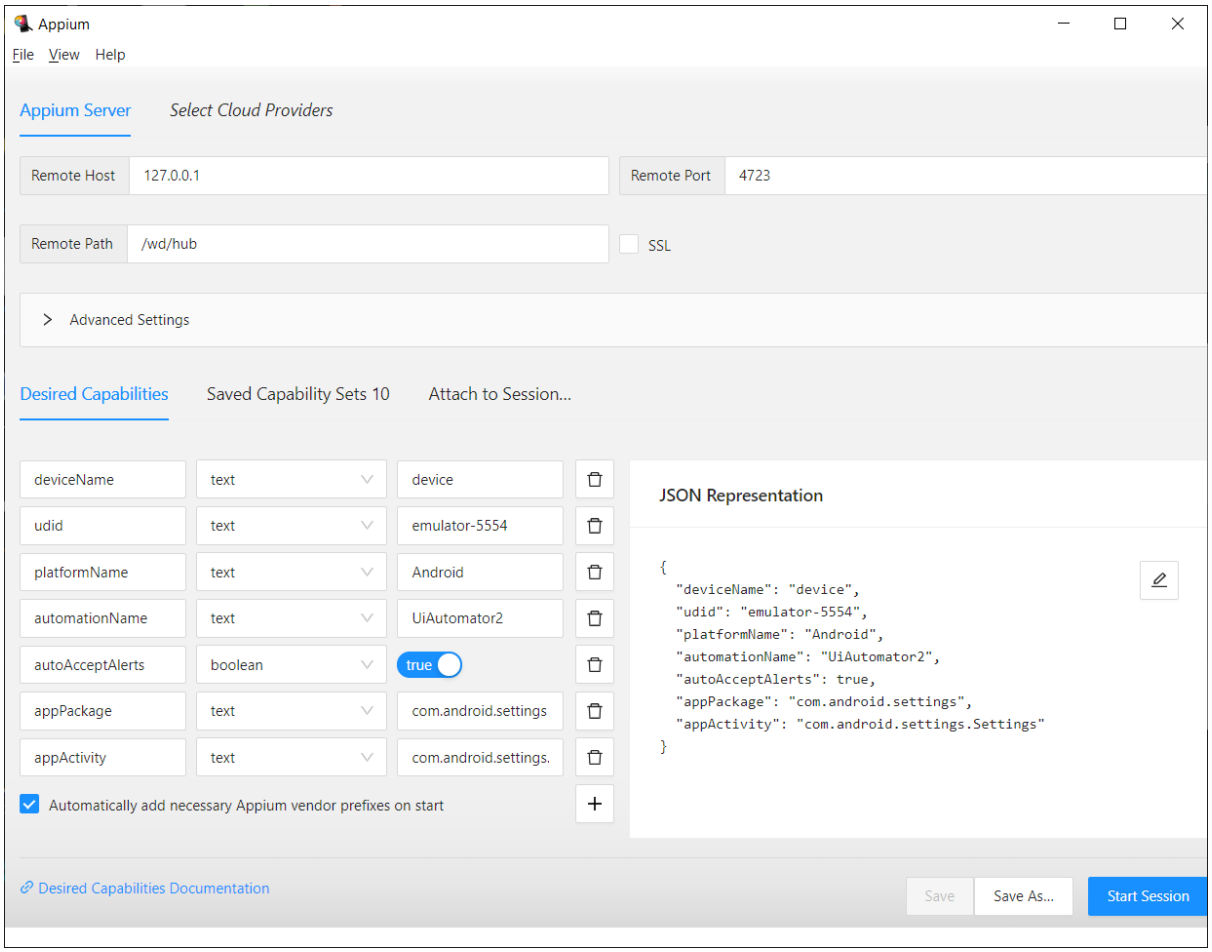
Using the Appium Inspector

1. Start the Appium Server application. If you use the GUI version, then it should look like this:



2. Click the "Start Server …" button.
3. Start the Appium Inspector and set the Desired Capabilities. This example inspects the Settings on a virtual device. If you want to inspect a real device, then you need to replace the "`device name`" with the name of the real device and the "`udid`" with the id that you

get from the "**`adb devices`**" command:



The Appium Inspector can also be used to inspect browser Web/Hybrid apps. Use the second icon ("Web/Hybrid App Mode") for this. The Desired Capabilities for the Google Chrome web browser will look like this:

```
{
  "deviceName": "Pixel 4a",
  "udid": "00c600c600c600c6",
  "platformName": "Android",
  "automationName": "UiAutomator2",
  "autoAcceptAlerts": true,
  "appPackage": "com.android.chrome",
  "appActivity": "com.google.android.apps.chrome.Main"
}
```

The Appium Inspector can record scripts in multiple languages (JavaScript, Java, Python, Ruby). To record, follow these steps:

1. Select the correct mode to either "Native App Mode", or "Web/Hybrid App Mode" (by choosing between the first two icons).
2. Start the recording with the "Start Recording" icon.

3. Record by using the icons from the "Selected Element" icons bar (for example "Tap", "Send Keys" etc.).
4. Stop the recording with the "Pause" icon.

The "uiautomatorviewer"

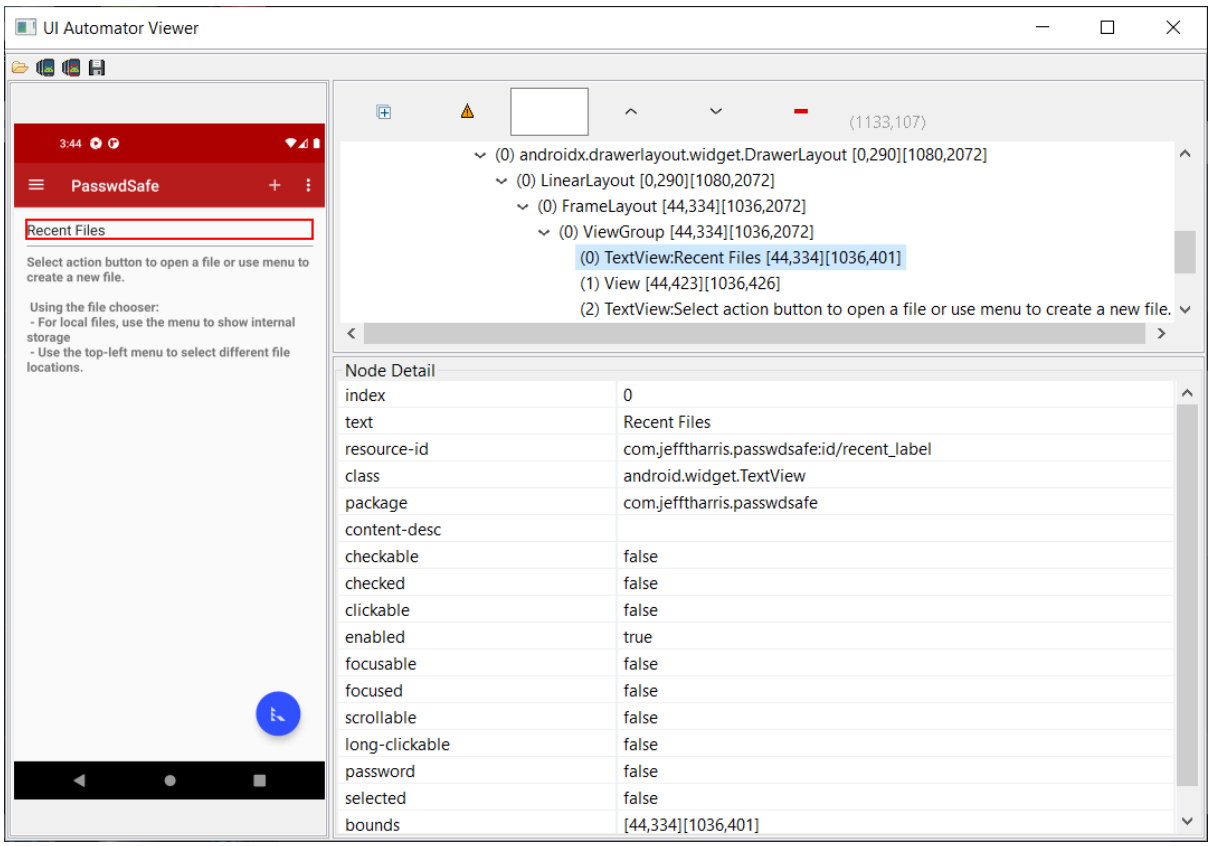
"**`uiautomatorviewer`**" is an application that comes packaged with the Android SDK and can be found under the "`tools`" folder in the "`bin`" subdirectory of the Android SDK. It is a tool that lets you inspect the User Interface (UI) of an application in order to explore the layout hierarchy, and view the properties associated with the controls.

While writing your tests, this tool is essential, as it exposes the "id" and other attributes of an element, which are required for writing good scripts.

Clicking on the devices icon (second from the left, next to the open folder icon) takes a dump of a UI XML snapshot of the screen shown on the device.

The left side of the tool shows how the device looks like. The right side is divided in two parts:

1. The upper half shows the UI XML snapshot and the nodes structure.
2. The lower half shows the details of the selected node with all the attributes.

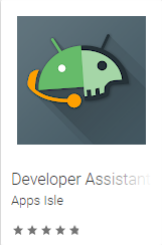


You can explore the properties of UI elements by clicking on them. Usually, the most helpful property for scripting is the "resource-id", but it is not always available. When it is not available, it is

often best to use the class name in combination with another property (such as text) to generate an Xpath expression that uniquely identifies the UI element.

The Developer Assistant app

The Android app "Developer Assistant" (available from the Google Play store) makes debugging of native Android apps as simple as debugging web pages using Google Chrome's Developer Tools. It allows you to inspect the View hierarchy, verify Layout, Style, preview Translations and more. Everything can be done directly from the mobile device.



The app often works better than the Appium Inspector, or the Android "`uiautomationviewer`", particularly when inspecting Android system components.

The "Developer Assistant" app has some free functions, but the full "Pro" functionality requires paid permanent or subscription licensing after 1 trial month.

A full Appium Java example

The Android app "PasswdSafe" is a port of the popular free and open-source password manager program "Password Safe" to Android. PasswdSafe allows you to safely and easily create a secured and encrypted user name/password list. With PasswdSafe all you have to do is to create and remember a single "Master Password" of your choice in order to unlock and access your entire user name/password list.

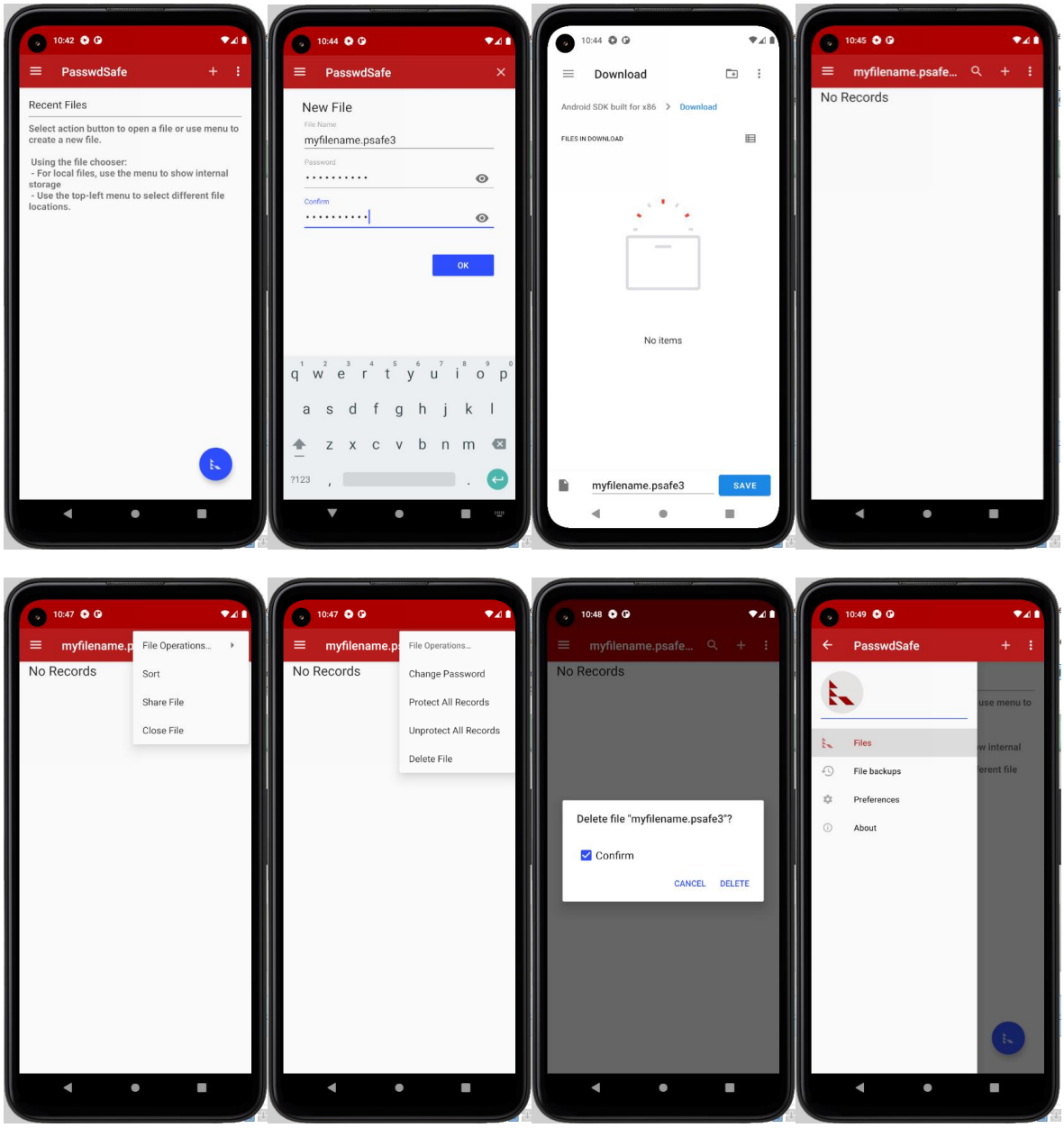
PasswdSafe supports viewing and editing of Password Safe data files. You can therefore exchange your data files between your Personal Computer and your Android device.

The PasswdSafe app can be installed on your Android device from the Google Play store.

The app and the source code can also be downloaded from:
https://sourceforge.net/projects/passwdsafe/

What the PasswdSafe example does

The PasswdSafe example project first creates an empty password file and then deletes the newly create file. Here are some screenshots from a virtual device:



Using the PasswdSafe example

The full example is available from GitHub:
https://github.com/BrunoBosshard/appium-passwdsafe

The example already includes the PasswdSafe app itself (the APK package), as well as the corresponding full source code. Both are located in the "`apps`" folder.

The PasswdSafe app gets automatically deployed on the (real or virtual) device.

Please note that you will have to change the "`device`" and "`udid`" settings in the "`AndroidSetup.java`" file to match your setup.

You can run the example from command line:

```
mvn clean verify
```

Project structure (in IntelliJ IDEA)