This guide is meant for all Suitest users that plan to test the HTML app used on Android devices through WebView and are using Cordova for WebView app creation.

Introduction

In order to instrument HTML hosted (WebView) applications with Suitest, there are 3 distinct steps needed (as described in our docs):

1. Adding Suitest Instrumentation Library (shortly IL) as dependency (and few other 3rd party libraries)
2. Instrumenting the Application class
3. Instrumenting the WebView(s) that display the content.

Apart from that, every webpage also needs to contain a small javascript which will instrument the webpage, allowing it to be tested with Suitest.

NOTE: This assumes that you do not have any Android plugin added and will be adding a new one. However, if you already have Android plugin added, you should have no problem following this guide and just modifying/adding the respective files. For the purpose of this guide, the custom Android plugin is placed inside SuitestApplication/ folder. If you already have Android plugin folder, or you decide to name yours differently, just replace SuitestApplication/ with <yourAndroidPluginFolder>/.

1. Adding Suitest IL library dependency

1.1. Add custom Android plugin to instrument the Cordova Android app with Suitest IL

If you already have an Android plugin successfully added to your Cordova project, you should be able to completely skip this step, and continue from #1.2.

- In case plugman is not installed, install it:

```
npm -g install plugman
```

- Create custom plugin with `plugman create` command, e.g:

```
plugman create -name SuitestApplication -plugin_id st.suite.il -
plugin_version 1.0.0
```

- Navigate into the newly created folder (`cd SuitestApplication`), add Android platform:

```
plugman platform add --platform_name android
```

- Create package.json, so that the plugin can be added into parent Cordova project. When prompted, either confirm the default values by hitting enter, or type in whatever you feel appropriate.

```
plugman createpackagejson ./ name: (SuitestApplication)
```

1.2. Adding IL library archives & custom build.gradle

- Within the /SuitestApplication folder (or within the folder of your already existing Android plugin), create new folder for .aar archives, for example `libs-aar`.

- Copy both IL archives (`SuitestIL.aar` and `SuitestILproduction.aar`) into /SuitestApplication/libs-aar/
- Create new file `build.gradle` inside the plugin's folder, more precisely `SuitestApplication/src/android/build.gradle`. This gradle file will add all the dependencies needed for Suitest IL. Copy-paste the following code to make the content look like this:

```
repositories {
  jcenter()
  flatDir {
    dirs 'src/main/libs'
  }
}
dependencies {
  debugImplementation 'com.android.support:appcompat-v7:25.2.0'
  debugImplementation 'com.google.code.gson:gson:2.8.0'
  debugImplementation(name:'SuitestIL', ext:'aar')
  releaseImplementation(name:'SuitestILproduction', ext:'aar')
}
```

(Alternatively, you can modify contents of yourPlugin/src/android/build.gradle file if you already have Android plugin).

NOTE: we will soon migrate from appcompat support lib to AndroidX. The dependency will need to be updated then.

1.3. Use IL library archives and custom build.gradle

- Modify contents of `SuitestApplication/plugin.xml` - add following lines inside `<platform name="android">` in order to use our custom build.gradle and bundle the .aar archives into resulting package:

```
<platform name="android">
  ...
  <framework src="src/android/build.gradle" custom="true"
type="gradleReference" />
  <resource-file src="libs-aar/SuitestIL.aar"
target="libs/SuitestIL.aar" />
  <resource-file src="libs-aar/SuitestILproduction.aar"
target="libs/SuitestILproduction.aar" />
</platform>
```

2. Create custom Suitest-instrumented Application class

By now, you should have a `SuitestApplication.java` class within /SuitestApplication/src/android/ folder. It extends `CordovaPlugin`, but we do not need that - we will modify the class to extend `android.app.Application`:

- Modify contents of `SuitestApplication/src/android/SuitestApplication.java` class to extend Application class, and add IL instrumentation code. Resulting class should look something like this:

```
package st.suite.il;

import android.app.Application;

import
st.suite.android.suitestinstrumentalservice.SuitestInstrumentalApplicat
ion;

public class SuitestApplication extends Application {

  @Override
  public void onCreate() {
     super.onCreate();
     SuitestInstrumentalApplication.webViewInit(this);

  }
}
```

- Within /SuitestApplication folder, update `plugin.xml` in order to make AndroidManifest.xml use our custom Application class - inside the `<platform name="android">` tag, add the following lines:

```
<platform name="android">
  ...
  <edit-config file="AndroidManifest.xml" target="/manifest/application"
mode="merge">
  <application android:name="st.suite.il.SuitestApplication" />
  </edit-config>
</platform>
```

- (If you already have Android plugin, just create a new Application class at /yourPlugin/src/android/ folder and use the code from above. Make sure to use correct package name and also make sure to use correct values for the `<edit-config>` tag inside /yourPlugin/plugin.xml file, as shown above.
- Increase minSdk to 21 to match Suitest IL's requirements. In the root of the Cordova project, modify `config.xml`: inside `<platform name="android">` tag, add following line:

```
<platform name="android">
  ...
  <preference name="android-minSdkVersion" value="21" />
</platform>
```

3. Instrumenting WebView that Cordova uses to display the applications's webpage.

We need to modify contents of /platforms/android/app/src/main/java/your-package-path/MainActivity.java, and instrument the WebView that Cordova uses to display content.

- It is done by calling `SuitestWebViewInstrumentation.instrument(webView);`. For best results with Suitest, the WebView should be instrumented before any content is loaded into WebView.

- Contents of this file could be overwritten by Cordova. Specifically this happen if you remove and then again add Android platform to your project (`cordova platform remove android`, `cordova platform add android`). If this happens, you will have to insert the following code again.
- The resulting class should look something like this (do not forget about the import statements):

```
package some.package; //leave your default value here

import android.content.pm.ApplicationInfo;
import android.os.Bundle;
import android.util.Log;
import android.webkit.WebView;

import org.apache.cordova.*;

import
st.suite.android.suitestinstrumentalservice.view.web.SuitestWebViewInst
rumentation;

public class MainActivity extends CordovaActivity {

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // enable Cordova apps to be started in the background
    Bundle extras = getIntent().getExtras();
    if (extras != null && extras.getBoolean("cdvStartInBackground",
    false)) {
       moveTaskToBack(true);
    }

    // start of Suitest instrumentation
    if (0 != (getApplicationInfo().flags &
    ApplicationInfo.FLAG_DEBUGGABLE)) { //use only for debug builds

      //call to ensure appView is created and initialized
      init();

      try {
        SuitestWebViewInstrumentation.instrument((WebView)
        appView.getView());
      } catch (ClassCastException e) {
        Log.e(this.getClass().getSimpleName(), "Suitest instrument
        WebView FAILED!", e);
      }
    }
    // end of Suitest instrumentation

    // Set by <content src="index.html" /> in config.xml
    loadUrl(launchUrl);
  }
}
```

4. Add the new custom Android plugin into parent project

- In the root directory of the project, add the newly created plugin

```
cordova plugin add SuitestApplication
```

- IMPORTANT: every time you update your Java code (as well as most of the other parts of the plugin), you actually have to remove the plugin and re-add, in order to update the source code with the latest changes. You can do it with following commands:

```
cordova plugin remove st.suite.il
cordova plugin add SuitestApplication
```

5. Instrument the webpage displayed by Cordova

Suitest instrumentation javascript has to be added to every webpage that is to be tested.

- By default, Cordova will use /www/index.html to display application content. Add the following code into <head> tag, ideally as the very first element:

```
<head>
  <script src="https://the.suite.st/app/your-app-id.js"></script>
  ...
</head>
```

- By default, the /www/index.html page contains the following line:

```
<meta http-equiv="Content-Security-Policy" content="default-src 'self'
data: gap: https://ssl.gstatic.com 'unsafe-eval'; style-src 'self'
'unsafe-inline'; media-src *; img-src 'self' data: content:;">
```

- However 2 rules prevent Suitest from working/displaying correctly, more precisely `default-src 'self'` and `img-src 'self'`. Removing them should make Suitest run smoothly.

Now you can build the application, and upload the apk file /platforms/android/app/build/outputs/apk/debug/app-debug.apk to Suitest.

IMPORTANT: When you upload the .apk build into Suitest, do not forget to check "This is an HTML based application", as shown here in our docs.

IMPORTANT: Every time you upload new build to Suitest, you have to increase the versionCode of your .apk file, otherwise the previous build will be used. To specify android versionCode, add `android-versionCode` to `<widget>` tag inside `config.xml` file in the root of your project, for example like this:

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="your.id.name" version="1.0.0" android-versionCode="3"
xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
...
```

Bonus: Using cleartext traffic in Android app

If the app uses any cleartext traffic, it should be explicitly enabled.

Please note that Suitest instrumentation should work with both http and https in the script (both `<script src="http://the.suite.st/app/your-app-id.js"></script>` and `<script src="https://the.suite.st/app/your-app-id.js"></script>` are ok.). If you decide to use https, you should however make sure that cleartext is enabled, otherwise Suitest will not run on Android 8+ devices.

Starting with Android 8, cleartext traffic is disabled by default. Enabling cleartext can be done either by specifying `android:usesCleartextTraffic="true"` in `<application>` tag inside `AndroidManifest.xml` (this enables all cleartext traffic), or for more specific approach, add custom network security config file, and specify it with `android:networkSecurityConfig` property inside `<application>` tag as well (see here).

To do the former, add following lines into `/SuitestApplication/plugin.xml` file inside `<platform name="android">` tag:

```
...
<platform name="android">
  ...
  <edit-config file="AndroidManifest.xml" target="/manifest/application"
  mode="merge">
    <application android:usesCleartextTraffic="true"/>
  </edit-config>
  ...
</platform>
```