Integration Guide QdsInappWrapperLib for Android

The integration of QdsInappWrapperLib takes place in the same way as IOLib of INFOnline (see INFOnline SZM Library Integration Guide Android).

Note:
Please note that the library for in-app surveys of agof may only be initialized if the user has agreed to participate in agof studies (TCF 2.0 - Purpose 1 and 9 or identical in content) (see Integration Guide to the InApp Measurement Library).

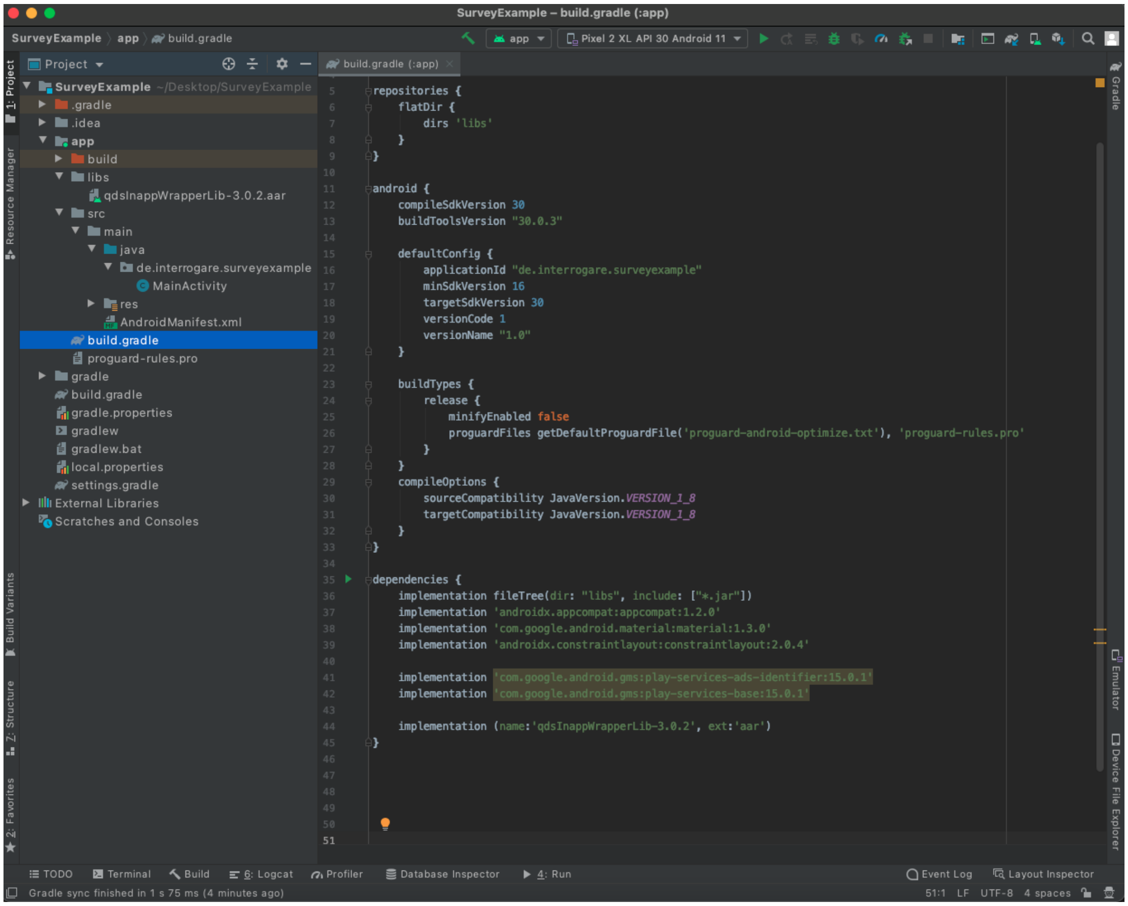
Integration with Android Studio (or IntelliJ IDEA)

The QdsInappWrapperLib is imported as an Android Library Project (.aar). To do this, copy the file qdsInappWrapperLib-xxx.aar to the libs folder of the application module.

The library itself and google-play-services (at least version 15) must be added to the build.gradle file of the application module as dependencies:

```
dependencies {
    implementation 'com.google.android.gms:play-services-ads-
identifier:15.0.1'
    implementation 'com.google.android.gms:play-services-base:15.0.1'

    implementation (name:'qdsInappWrapperLib-3.0.2', ext:'aar')
    ...
}
```



```
repositories {
    flatDir {
        dirs 'libs'
    }
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "de.interrogare.surveyexample"
        minSdkVersion 16
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'

    implementation 'com.google.android.gms:play-services-ads-identifier:15.0.1'
    implementation 'com.google.android.gms:play-services-base:15.0.1'

    implementation (name:'qdsInappWrapperLib-3.0.2', ext:'aar')
}
```

Add the following entry to the property `repositories` to include the dependency correctly:

```
repositories {
    ...
    flatDir {
        dirs 'libs'
    }
}
```

The library must be in the libs folder of the project, otherwise the path will not match.

Integration in Eclipse

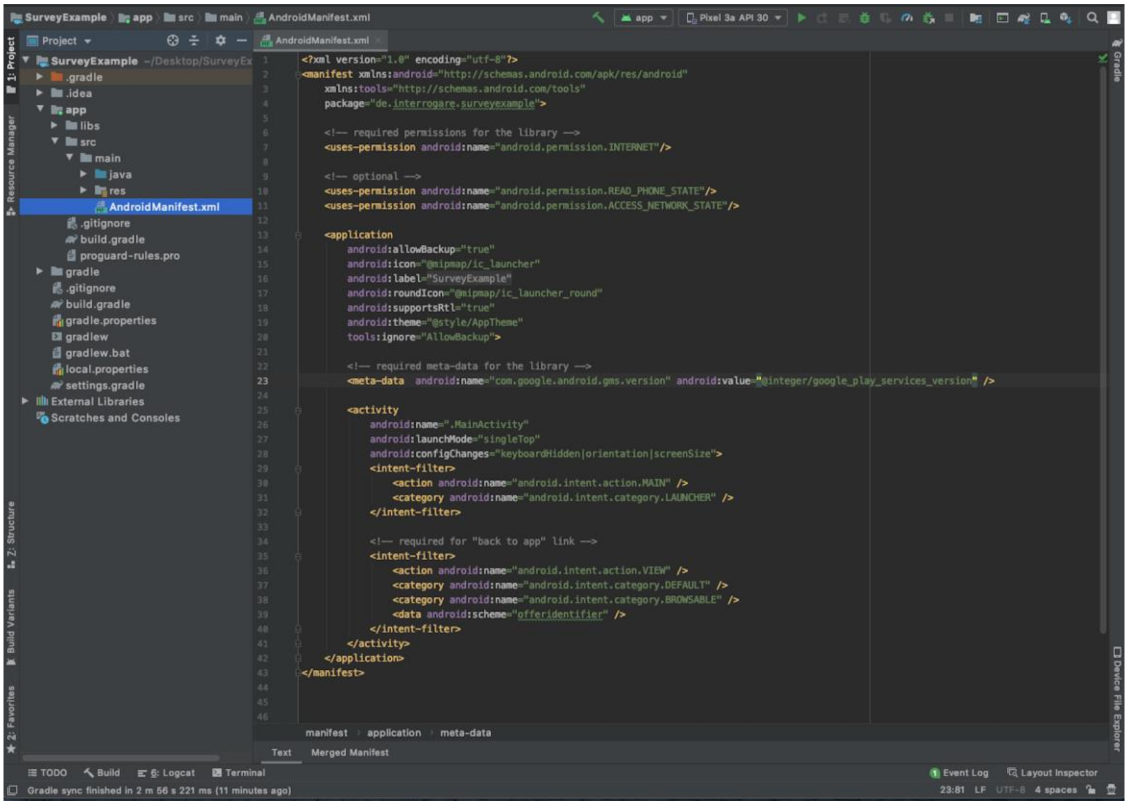
As part of the transition to the Gradle build system and the release of Android Studio, Google does not officially support the import of .aar files into Eclipse. There is an unofficial method using the gradle-eclipse-aar-plugin (see: https://github.com/ksoichiro/gradle-eclipse-aar-plugin).

Important:
The following information must be entered in `AndroidManifest.xml` of the app:

If IOLib is already included, the permissions must have been set beforehand. Enter the following permissions into the file `AndroidManifest.xml`, if none are present.

```
<!-- required permissions for the library -->
<uses-permission android:name="android.permission.INTERNET" />

<!-- optional -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
...
<application>
    ...
    <!-- required meta-data for the library -->
    <meta-data android:name="com.google.android.gms.version"
               android:value="@integer/google_play_services_version" />
    ...
</application>
```



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="de.interrogare.surveyexample">

    <!-- required permissions for the library -->
    <uses-permission android:name="android.permission.INTERNET"/>

    <!-- optional -->
    <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="SurveyExample"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:ignore="AllowBackup">

        <!-- required meta-data for the library -->
        <meta-data  android:name="com.google.android.gms.version" android:value="@integer/google_play_services_version" />

        <activity
            android:name=".MainActivity"
            android:launchMode="singleTop"
            android:configChanges="keyboardHidden|orientation|screenSize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>

            <!-- required for "back to app" link -->
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />
                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />
                <data android:scheme="offeridentifier" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Gradle sync finished in 2 m 56 s 221 ms (11 minutes ago)

To ensure the correct display of the survey dialogue after screen rotation, add two further properties to the corresponding activity (which calls the QdsInappWrapperLib):

```
<activity
              ...
              android:launchMode="singleTop"
              android:configChanges="orientation|screenSize"
>
  ...
</activity>
```

INFO: From the (minimal) API level 11 (Android 3.0), setting the property `android:configChanges` is no longer required.

Integration of the return address (optional)

The return to the app takes place by registration of a custom URL scheme. Add an intent filter to the corresponding activity for this purpose:

```
<activity
              …
              android:launchMode="singleTop"
              android:configChanges="orientation|screenSize">
    …
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="offeridentifier" />
    </intent-filter>
</activity>
```

The term `offerIdentifier` must be replaced with the identifier assigned by INFOnline (see: Functionality).

Functionality

The QdsInappWrapperLib is represented by the class `QdsInappWrapper`. A concrete instance of this class is created with:

```
    new QdsInappWrapper(Activity activity);
```

The QdsInappWrapperLib must be initialized when calling the app. This takes place by way of the above-mentioned call. `MainActivity` should be passed on for `activity`. To enable display of a survey invitation, the method

```
    showSurveyInvitation(String offerIdentifier, String countryCode, Boolean
userHasAcceptedPermissionForTracking);
```

must be called for the instance; e.g. in the MainActivity of the method `onStart()`. The `offerIdentifier`, country code (with "de") and the flag whether the user gave permission for tracking must be passed on in this context. The parameter `offerIdentifier` is the unique identifier of the offer for the respective app. The `offerIdentifieris` assigned by INFOnline for each app and each operating system. The parameter `userHasAcceptedPermissionForTracking` is the flag whether the user gave permission for tracking.

Important:
Any initial loading processes should ideally be completed, since a poll invitation may be displayed immediately after calling `showSurveyInvitation()`.

```
    QdsInappWrapper qdsInappWrapper = new QdsInappWrapper(this);
    qdsInappWrapper.showSurveyInvitation("offerIdentifier", "de", true);
```

If loading dialogues are shown when starting the app, `showSurveyInvitation()` should only be called subsequently, since the invitation dialogue is displayed prematurely otherwise. Generally, calling `showSurveyInvitation()` means that a dialogue with a survey invitation may be displayed after calling of the method. `showSurveyInvitation()`should be called once the app is ready to display a questionnaire if needed. Generally, `showSurveyInvitation()` can be called repeatedly.

Notes:

The QdsInappWrapperLib should be initialized at the start of the "MainActivity". If this is not possible for technical or content reasons, a suitable alternative entry point is to be selected by the developer. The important factor is that the user must get the opportunity to interact with the corresponding activity and the potential survey dialogue.

The **opt-out function** is controlled by QdsInappWrapperLib via the survey invitation, so no further action is necessary for this. Survey and data protection information on the agof study are optionally shown to the survey participant within the survey.

Test - Display dialogue window:

Please use the INFOnline SZM checker to mark the device as a test device. The documentation for the INFOnline SZM checker is available at https://www.infonline.de/downloads/9-1356-942/INFOnline_Manual_SZM-Checker_InAPP-Befragung.pdf. Please check correct display of the survey invitation, clickability of all buttons, and opening of the questionnaire by clicking on the `participate` button.

The customer is responsible for installation of QdsInappWrapperLib in accordance with the valid guidelines of the testing organization. The customer is responsible for ensuring that the implementation of the library is correct even if the app is changed.

Support:
**agof Service Center
Phone: 0800 / 410 29 77 (free of charge)
From abroad: 0049 -- (0)228 / 410 29 47
E-mail: servicecenter@agof.de