02 - Native App Integration (Android)

Procedures

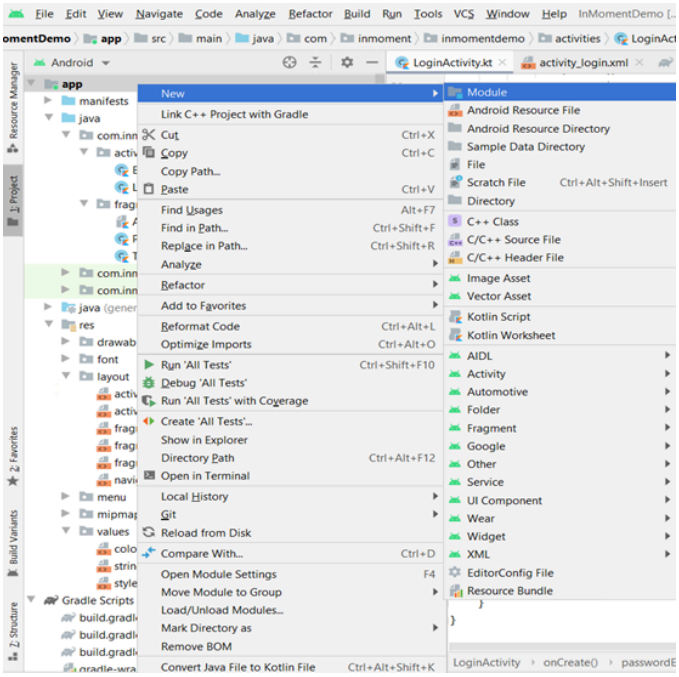
Here are the steps necessary to integrate the in-app survey Android SDK (In-App Survey Module):

1. Download the .aar file.

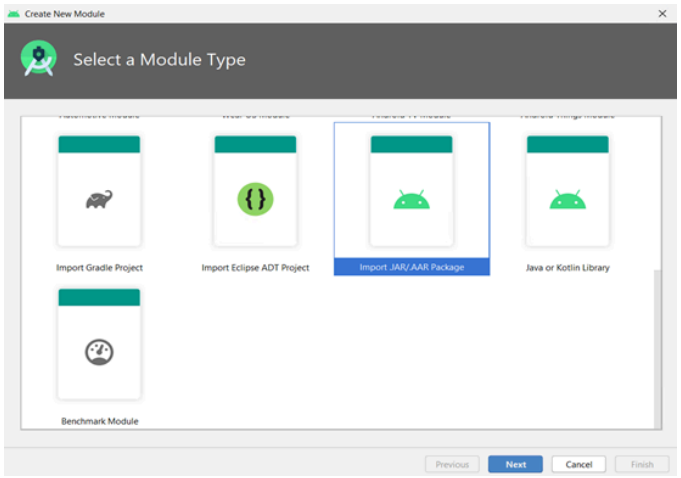
2. Open your existing application in Android Studio.

3. Perform the following steps:

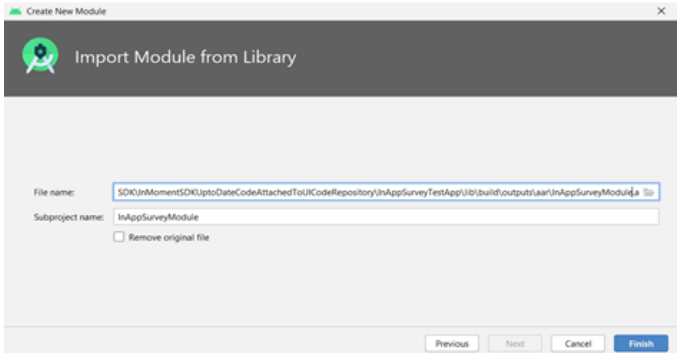
a. Add “New Module”.



b. Choose “Import .JAR/.AAR Package”.



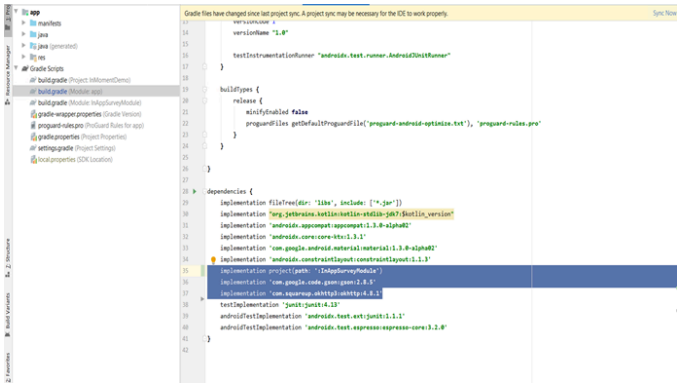
c. Choose the file.



d. It should pre-fill the **Subproject name** with “InAppSurveyModule”. If not, name it as such.

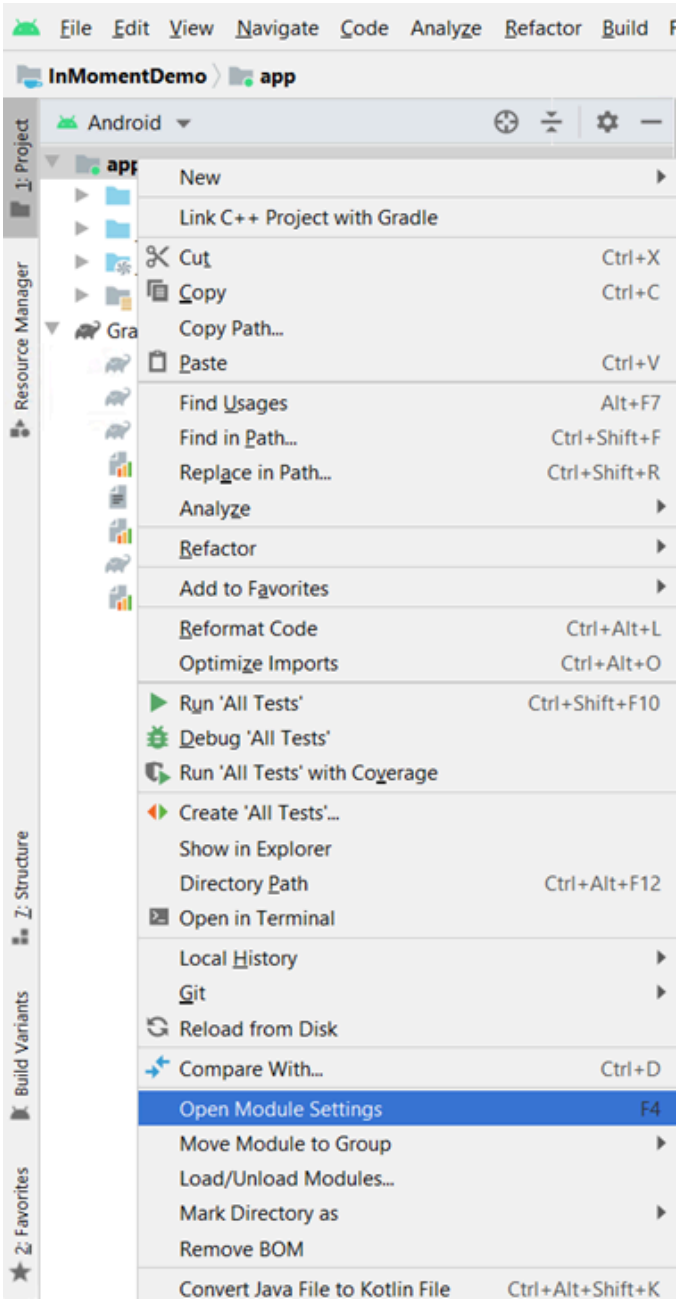
4. Change your module-specific Gradle file to include the following in the dependencies section.

a. *implementation project(path: ':InAppSurveyModule')*
b. *implementation 'com.google.code.gson:gson:2.8.5'*
c. *implementation 'com.squareup.okhttp3:okhttp:4.8.1'*

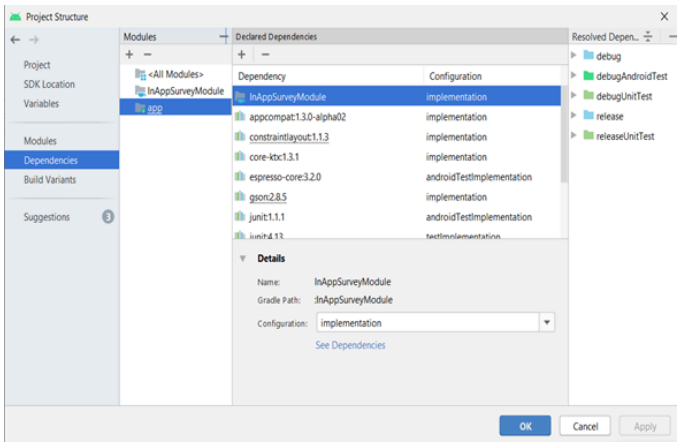


5. Ensure that the module in which you're using the In-App Survey Module includes the module provided.

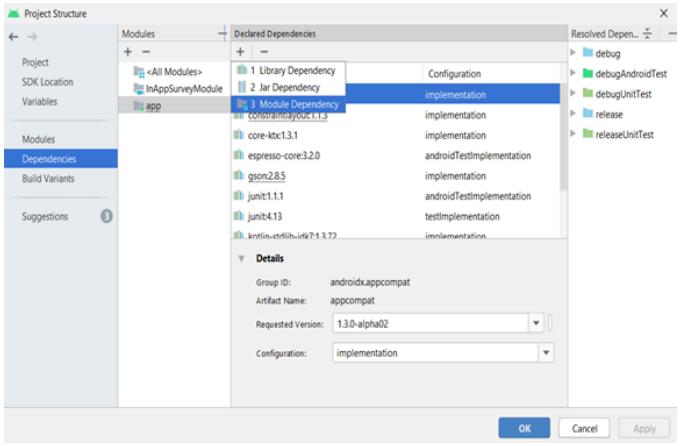
a. Right-click on the module where you're intending to use it and choose "Open Module Settings".



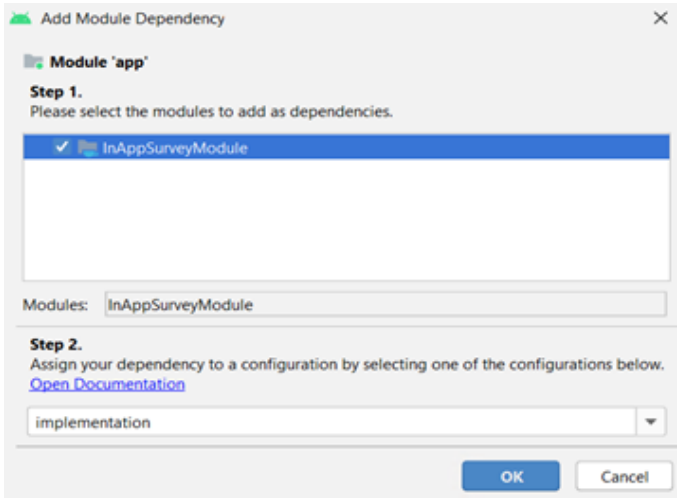
b. Choose the module, then select the **Dependencies** tab.



c. Choose the plus sign and choose “3 Module dependency”.



d. Select "InAppSurveyModule".



6. To inject a survey, do two things:

a. Add the necessary imports:

i. *import com.maritzcx.inappsurvey.SurveyPresentation.Models.RuleRequestModel;*

ii. *import com.maritzcx.inappsurvey.SurveyPresentation.Presenter;*

iii. *import com.maritzcx.inappsurvey.SurveyPresentation.Models.CustomPromptDataModel*

b. Add the call to possibly present the survey to the user. Here are steps with code:

Step 1 – Create RuleRequestModel object

RuleRequestModel aRuleRequestModel = new RuleRequestModel(<programToken>, <eventAlias>,<requestCode>, <locale>);
For eg. - RuleRequestModel aRuleRequestModel = new RuleRequestModel("19283", "user_viewed_account_details",1, null);

Step 2 – (Optional) Add prepopulated data in rule request if needed

aRuleRequestModel.addPrePopulationData("<namedfield>", "<namedfieldvalue>");
for eg. - aRuleRequestModel.addPrePopulationData("firstName", "abc");

Step 3 – (Optional) Provide custom prompt configuration as below if custom invitation needs to display

If you want to use the default prompt, then skip this step. For more details see the table Custom Prompt Parameters and Descriptions.

String customPromptJson =
"{\"headerImageURL\":\"https://maritzcxenterpriseoperations...ero_large.jpg\",\"bodyText\":\"Your feedback is incredibly valuable and will allow us to focus on those aspects that are most important to you.

 Please take our 1 minute survey.\",\t\"bodyBackgroundColor\":\"#ffffff\",\"mainBackgroundColor\":\"#7122ff\", \"noButtonBackgroundColor\":\"#efb237\",\"noButtonText\":\"No Thanks\", \"noButtonTextColor\":\"#ffffff\",\"yesButtonBackgroundColor\":\"#efb237\", \"yesButtonText\":\"Start Survey\",\"yesButtonTextColor\":\"#ffffff\", \"footerBackgroundColor\":\"#0d55d0\",\"footerText\":\"Please contact if need more details.\", \"footerTextColor\":\"#ffffff\",\"footerHeaderText\":\"Thank you very much.\", \"optOutButtonText\":\"Opt-Out\", \"optOutButtonTextColor\":\"#efb237\", \"disclosureText\": \"If you decline this invitation, we will not offer this survey again while using this platform for a period of time.\"," +
"\"disclosureTextColor\":\"#000000\"}";

CustomPromptDataModel customPromptDataModel = new Gson().fromJson(customPromptJson, CustomPromptDataModel.class);

Step 4 – Present survey with alternate approaches

1. Present a survey with the default prompt (no custom invitation prompt).

Presenter.getInstance().initiateSurvey(this, aRuleRequestModel);

2. Present survey with custom prompt.

Presenter.getInstance().initiateSurvey(this, aRuleRequestModel, customPromptDataModel);

3. Present the survey with the custom prompt and the next activity which needs to display after the survey is completed or fails.

Presenter.getInstance().initiateSurvey(this, aRuleRequestModel, new Intent(this, NextActivity.class),customPromptJson);

Not Passing in the Locale, Resuming to Subsequent Activity, Request Code = 1

RuleRequestModel aRuleRequestModel = new RuleRequestModel("19283", "user_viewed_account_details",1, null);
Presenter.getInstance().initiateSurvey(this, aRuleRequestModel, new Intent(this, NextActivity.class),customPromptJson);
Passing in the Locale, Resuming to Subsequent Activity, Request Code = 2
Locale aLocale = this.getResources().getConfiguration().locale;
RuleRequestModel aRuleRequestModel = new RuleRequestModel("19283", "user_viewed_notifications",2, aLocale);
Presenter.getInstance().initiateSurvey(this, aRuleRequestModel, new Intent(this, NextActivity.class),customPromptJson);

Passing in the Locale, staying on same activity, request code = 5

Locale aLocale = this.getResources().getConfiguration().locale;
RuleRequestModel aRuleRequestModel = new RuleRequestModel("19283", "clicked_logout",5, aLocale);
Presenter.getInstance().initiateSurveyAndFinish
(this, aRuleRequestModel,customPromptJson);

Keep the following in mind:

- “customPromptDataModel” is an instance of “CustomPromptDataModel” configured in Step 3 above.
- “customPromptJson” is a JSON string for the custom prompt.
- Pass “null” if you don’t want a custom invitation. The SDK will then display the default alert dialogue.

Note: If no locale is provided, the SDK will use the current locale.

So, what is the call doing? You create a model and provide the In-App Survey Module some details. They are:

- The optional locale.
- The program token; provided to you by your business owner or manager.
- The event alias. This is an agreed-upon value between you and your business owner/manager. It’s the way you indicate the context of the user (just hit a button, landed on a screen, etc.).

- A request code. This is used to identify, later, which call was made. Using the onActivityResult() method, the developer must check for that value and determine what to do when the survey is completed. Here is an example:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
   switch (requestCode) {
      case 1:  startActivity(new Intent(this, NextActivity.class));
         break;
      case 2:  startActivity(new Intent(this, NextActivity.class));
         break;
      case 3:  startActivity(new Intent(this, NextActivity.class));
         break;
      case 4:  startActivity(new Intent(this, NextActivity.class));
         break;
      case 5:  finish();
         break;
   }
}
```

Then, you're passing off control to the In-App Survey Module. You may or may not pass in the activity that you would for the user to land on, once the survey is completed. There are multiple constructors for this.

Do you need to pass in agreed-upon [the MaritzCX implementation team and the client's app implementation team must agree upon a naming convention] survey prefill values? If so, do this to the RuleRequestModel() instance before calling the initiateSurveyAndFinish() or the initiateSurvey() methods:

```
aRuleRequestModel.addPrePopulationData("<namedfield>", "<namedfieldvalue>");
```

This is explained in Step 2 above. Execute this method for each name/value that you wish to prefill.