SDK Integration Full Flow

- Android SDK Guide

Android SDK Guide

This guide provides detailed instructions for integrating the Scanovate Colombia SDK into your Android application, enabling robust identity validation processes through facial biometric verification.

Requirements and Compatibility

Before starting the integration process, ensure your development environment meets the following requirements:

- **Android Studio**: The latest version is recommended for optimal compatibility.
- **Minimum SDK Version**: Android SDK version 21 (Lollipop) or higher.
- **Target SDK Version**: Android SDK version 34 (Android 14) to ensure your app is compatible with the latest Android OS.
- **Compile SDK Version**: Android SDK version 34.

Installation

1. Add the library

Download the "scanovate_colombia_@latest.aar" library and add it to your project's `libs` folder. Ensure you configure your project's `build.gradle` file to include the library as a dependency:

> ```
> dependencies {
>     implementation(name: 'scanovate_colombia_@latest', ext: 'aar')
> }
> ```

2. Import Required Libraries

Add the following imports in your activity or fragment where you intend to use the Scanovate SDK:

> Java
>
> ```
> import mabel_tech.com.scanovate_demo.ScanovateHandler;
> import mabel_tech.com.scanovate_demo.ScanovateSdk;
> import mabel_tech.com.scanovate_demo.model.CloseResponse;
> import mabel_tech.com.scanovate_demo.network.ApiHelper;
> ```

```
import mabel_tech.com.scanovate_demo.network.RetrofitClient;
```

The `CloseResponse` object will contain the results of the transaction, providing detailed feedback on the validation process.

Example Implementation

For a practical example of how to implement the Scanovate SDK in your Android application, refer to the following steps:

- **Setup UI Elements**: Initialize buttons, text views, and other UI elements in your activity's `onCreate` method. This setup includes buttons for starting the enrollment and verification processes, a text view for displaying results, and an edit text for user input.
- **Invoke the SDK**: Use the `ScanovateSdk.start` method to launch the Scanovate SDK. This method requires several parameters, including language, project name, API key, product ID, and the SDK URL. It also allows you to specify the type of capture (e.g., liveness detection, document capture) and whether to capture the front or back side of a document.
- **Handle Callbacks**: Implement `ScanovateHandler` to manage success and failure callbacks. On success, process the `CloseResponse` object to display the transaction result. On failure, handle errors accordingly.

> Example
>
> // Example capture method implementation

```
ScanovateSdk.start(
        this,         // Contex
        "1",         // documentType
        1,          //productId
        "1",        //RiskId
        "https://api-qa.ado-tech.com/api/EventTracer/",   //Url_TracerBackendServices
        customerID,       //CustomerID (CID)
        sessionID,          //SessionID (SID)
        "Example",       //projectName
        "F99264E00A2FEA7",     //apiKey
        "https://adocolumbia.ado-tech.com/Example/api/", //UrlBase
        numberIdentification,    //numberIdentification
```

```
            ImmersiveMode,          // Inmersive Modo
            verification,          //verification
            "admin",               //userName
            "0f2ebb2d8b575d53251ba6704f762cd789bb592b", //password
            object : ScanovateHandler {
               override fun onSuccess(response: CloseResponse?, code: Int, uuidDevice:
String?) {
                    // Respuesta las salidas del SDK
               }

               override fun onFailure(response: CloseResponse?) {
                  // Respuesta las salidas del SDK
               }
            }
         )
```

Parameters Explained

- **projectName**: Unique identifier for your project.
- **Context**: Context of the activity from which the SDK application is launched.
- **apiKey**: Authentication key provided by Scanovate.
- **productId**: Identifies the specific Scanovate product/service being used.
- **sdkUrl**: The base URL for making API calls to the Scanovate services.
- **Url_TracerBackendServices**: Url for the event reporting service is not required and is only an extra service. (**Optional**)
- **ImmersiveMode:** Mode to make the component consume all available space while hiding the system UI.
- **Process_ID**: Process identifier to perform the events mapped at the SDK level. (**Optional**)
- **Verification**: A parameter used to perform validation or verification within the system.
- **UserName**: The username or identifier required for authentication using the OAuth 2.0 protocol.
- **Password**: A password that has been hashed using the SHA-1 encryption algorithm for secure storage or validation.
- **CustomerID:** client identifier (Optional)

Process Transaction Results

After capturing the necessary data, use the `RetrofitClient` to send the data for validation and display the final state of the transaction to the user.

The SDK will complete the transaction when it is part of an enrollment process. It will return a `stateName` with a pending status code, which can be accessed using the following in Java:

java

```
response.getExtras().getStateName();
```

Or using Kotlin properties:

kotlin

```
val stateName = response?.extras?.stateName
val idState = response?.extras?.idState
val idTransaction = response?.transactionId
val additionalInfo = response?.extras?.additionalProp1
```

With these values,This `transactionId` should be used to verify the final information by invoking the **ValidationId** service to query the final transaction result.

In the case of a verification process, the system will respond with a `stateName` indicating that the person is already registered, assigning state **14**.

Overview

To retrieve the results of an identity verification transaction, you will need the `transactionId` that was provided in the callback after the verification process. This `transactionId` serves as a unique identifier for the transaction.

CURL Command Structure

The `curl` command to retrieve the transaction results is structured as follows:

> curl -X GET
> "{URL_Base}/api/{ProjectName}/Validation/{id}?returnImages=false" \
> -H "accept: application/json" \
> -H "apiKey: your_api_key" \
> -H "returnDocuments: true" \
> -H "returnVideoLiveness: false"

Parameters Explained

- **{URL_Base}**: The base URL of the identity verification service. This should be replaced with the actual URL provided to you.
- **{ProjectName}**: The name of your project as registered with the identity verification service. Replace `{ProjectName}` with your specific project name.
- **{id}**: The unique identifier (`codeId`) for the transaction you wish to retrieve. This ID is typically provided in the callback after the verification process.
- **returnImages** (Query Parameter): Specifies whether to include images in the response. Setting this to `false` excludes images from the response, while `true` includes them.

Headers

- **accept**: Indicates the expected media type of the response, which is `application/json` for JSON-formatted data.
- **apiKey**: Your API key for authentication with the identity verification service. Replace `your_api_key` with the actual API key assigned to your project.
- **returnDocuments**: A header that determines whether document data should be included in the response. Setting this to `true` includes document data, while `false` excludes it.
- **returnVideoLiveness**: Indicates whether the response should contain video data from the liveness verification process. `true` includes video data, and `false` excludes it

Json Example Response


```
{
  "Uid": "4a5528fe-4dbe-4864-993e-b4ed50e7622c",
  "StartingDate": "2024-07-17T09:39:56.07",
  "CreationDate": "2024-07-17T09:40:44.527",
  "CreationIP": "54.86.50.139",
  "DocumentType": 1,
  "IdNumber": "IdNumberNumber",
  "FirstName": "FirstNameuUser",
  "SecondName": "SecondNameUser",
  "FirstSurname": "FirstSurnameUser",
  "SecondSurname": "SecondSurnameUser",
  "Gender": "M",
  "BirthDate": "2001-10-24T00:00:00",
  "Street": null,
  "CedulateCondition": null,
  "Spouse": null,
  "Home": null,
  "MaritalStatus": null,
  "DateOfIdentification": null,
  "DateOfDeath": null,
```

```
"MarriageDate": null,
"Instruction": null,
"PlaceBirth": "PlaceBirthUser",
"Nationality": null,
"MotherName": null,
"FatherName": null,
"HouseNumber": null,
"Profession": null,
"ExpeditionCity": null,
"ExpeditionDepartment": null,
"BirthCity": null,
"BirthDepartment": null,
"TransactionType": 1,
"TransactionTypeName": "Enroll",
"IssueDate": "2019-11-06T00:00:00",
"BarcodeText": null,
"OcrTextSideOne": null,
"OcrTextSideTwo": null,
"SideOneWrongAttempts": 0,
"SideTwoWrongAttempts": 0,
"FoundOnAdoAlert": false,
"AdoProjectId": "2",
"TransactionId": "2299",
"ProductId": "1",
"ComparationFacesSuccesful": false,
"FaceFound": false,
"FaceDocumentFrontFound": false,
"BarcodeFound": false,
"ResultComparationFaces": 0.0,
"ResultCompareDocumentFaces": 0.0,
"ComparationFacesAproved": false,
"ThresholdCompareDocumentFaces": 0.0,
"CompareFacesDocumentResult": null,
"Extras": {
    "IdState": "2",
    "StateName": "Proceso satisfactorio"
},
"NumberPhone": null,
"CodFingerprint": null,
"ResultQRCode": null,
```

```
    "DactilarCode": null,
    "ReponseControlList": null,
    "Latitude": "4.710988599999999",
    "Longitude": "-74.072092",
    "Images": [],
    "SignedDocuments": [],
    "Scores": [
      {
        "Id": 2,
        "UserName": null,
        "StateName": "Proceso satisfactorio",
        "CausalRejectionName": null,
        "StartingDate": "0001-01-01T00:00:00",
        "Observation": null
      }
    ],
    "Response_ANI": null,
    "Parameters": null,
    "StateSignatureDocument": null,
    "SessionId": null,
    "CustomerIdFromClient": null,
    "ProcessId": null,
    "DocumentTypeFromClient": 0,
    "IdNumberFromClient": null,
    "NotEnrolledForComparisonWithClientData": false
}
```


Usage Tips

- Ensure all placeholders in the `curl` command are replaced with actual values specific to your project and the transaction you're querying.
- Execute the `curl` command in a terminal or command-line interface. The server's response will include the transaction details and validation results, according to the parameters you've set.
- Carefully process the JSON response to extract and utilize the verification information as needed in your application or for compliance purposes.

By following these guidelines and using the corrected URL structure and parameters, you can effectively retrieve detailed information about identity verification transactions, enhancing your application's security and user management processes.