

Avarta Integration Library

Android Integration Guide (Version 2.5)

Internal Use Only

*Avarta*

*22 Cross Street, Singapore*

<http://www.avarta.io>

Contents

Introduction 3

Device Requirements 3

Setup 4

Usage 5

Introduction 5

Initialization 5

Start workflow 5

Continue workflow after handling pending activity 6

Cancel workflow 6

Stop all requests 6

Change password 7

Reset password 7

Calculate users score 7

Additional methods for sending requests 7

Change base url 8

Change organization key 8

Listen to workflow events 9

Introduction

This document briefly describes the SDK for the Android Avarta IntegrationLibrary Client. This demonstration client should enable developers to quickly integrate the IntegrationLibrary SDK into their native applications.

Device Requirements

SDK Integration v1.0

Min. Android version: 16

Setup

1. Add ‘integration-library.aar’ into your project dependencies

2. Check the ‘internet permission’ in the manifest file

<uses-permission android:name="android.permission.INTERNET" />

Usage

Introduction

**IntegrationApiManager** is the main wrapper class of the integrationLibrary SDK. All operations with library take place only through it. Below we’ll describe all available methods.

Initialization

IntegrationApiManager apiManager = new IntegrationApiManager(String baseUrl,

String organizationKey);

baseUrl - server url (e.g. "[https://google.com/](https://solus-connect-platform-test-gemini.azurewebsites.net/)")

organizationKey - your organization key

Start workflow

Call the IntegrationApiManager#startWorkflow() method. Note, that previously you should attach listener #IWorkflowProccessListener(see «Listen to workflow events» block for detailed description), with which you can control the process

void startWorkflow(@ApplicationCode String applicationCode, Context context, String workflowType, String userName)

applicationCode - predefined string, with which the server identifies your application

workflowType - set workflow, which you want to start

context - strongly recommended to use getApplicationContext()

userName - set user, for which you will start the workflow

For now, we have next types of the workflow:

* ENROL;
* AUTH;
* DE-ENROL;
* STEP-UP;
* STEP-UP-ELEVATED;

Note, that in this case integration library will check on exist workflow processes. If you want to prevent this check use the method below and pass «checkPendingActivities» as false:

void startWorkflow(@ApplicationCode String applicationCode, Context context,

String workflowType, String userName,

boolean checkPendingActivities)

Continue workflow after handling pending activity

Worklflow process constists of few stages - process activities. After starting workflow, users of this SDK get the task from the integration library in the callback (IWorkflowProccessListener#onContinueEnroll(WorkflowType, String)) and after handling the corresponding activity developer should pass the data back into the integration library through this method

void processActivityData(Context context, final String activityCode,

String activityData)

Cancel workflow

void cancelWorkflow(Context context)

This method will cancel current workflow, and if the request will be successful, user will get the callback in the IWorkflowProccessListener#onWorkflowAbort() (see «Listen to workflow events» block for detailed description), otherwise IWorkflowProccessListener#onWorkflowError() will fire.

Stop all requests

void cancelAllRequests(Context context)

This method will cancel all requests from the queue. You can use it to handle your lifecycle in the application

Change password

void changeUserPassword(Context context, String userName,

String oldPassword, String newPassword,

IBaseApiListener<BaseResponse> listener)

context - prefer to use application context

userName - user for whom reset password will be executed

oldPassword - current user password

newPassword - new user password

listener - callback for the result

Reset password

void resetPassword(String applicationCode, Context context,

String userName, String workflowKey)

applicationCode - predefined string, with which the server identifies your application

context - prefer to use application context

workflowKey - workflow key for resetting password

Note, that you previously you should attach workflow process listener to handle the result of the resetting password

(see «Listen to workflow events» block for detailed description)

Calculate users score

This method will get the user scores. Note, that you can execute it only after successful completed workflow, in other cases you will get IllegalStateException the {IWorkflowProccessListener#onWorkflowError(Throwable, String, String)}

void calculateUserScore(Context context,

IBaseApiListener<ScoreResponse> responseListener)

Additional methods for sending requests

Method that executes multipart request:

void sendMultipartRequest(Context context, String url, byte[] data, IBaseApiListener<byte[]> responseListener)

context - recommended to use application context

url - the endpoint of the request

data - byte array of the data

responseListener - callback of the request to handle result

Method for sending POST requests:

<T extends BaseResponse> void sendPostRequest(Context context, String endpoint, Class<T> responseClass, Map<String, String> params, IBaseApiListener<T> responseListener)

context - prefer to use application context

url - the endpoint of the request

params - map with request params

responseListener - callback of the request to handle result

Change base url

void switchBaseUrl(String baseUrl)

baseUrl - base url for the executing requests(e.g. - <https://google.com/>)

Change organization key

void switchOrganizationKey(String organizationKey)

organizationKey - new organization key

Class<T> responseClass - create class the extends from the BaseResponse and override the method «parse(JSONObject obj)» according to your expected response from server

public class BaseResponse {

public BaseResponse() {

}

public BaseResponse parse(JSONObject obj) throws JSONException {

return this;

}

}

Listen to workflow events

Attach listener to the IntegrationApiManager class via method:

void setProcessListener(IWorkflowProccessListener proccessListener)

IWorkflowProccessListener is the main listener of the workflow, which notifies about all events where the user should participate. It contains five callbacks:

1.

void onWorkflowError(Throwable error, String workflowType, String userName);

This method will fire when there is any error during the current workflow. Brief description of the params:

error - exception, which was thrown during the workflow. It could be an instance of:

* **ApiErrorException**:  
  this indicates that server response contains some of the predefined errors (see com.Avarta.integrationlibrary.interfaces.ApiErrorCode);
* **JsonParserException**:  
  crash during parsing server responses;
* **WorkflowFailedException**:  
   indicates that workflow was failed, but also it was completed according to the server logic;
* **WorkflowAbortedException**:   
  indicated that workflow was aborted manually after the fail;
* **VolleyError**:   
  network error;
* In other cases SDK will throw exception which is not handled by IntegrationLibrary SDK;

You can find all exceptions in the com.Avarta.integrationlibrary.exceptions package

userName - name of the user, for which workflow was started

2.

void onConfigLoaded(AvartaConfig AvartaConfig);

This method will be called, when SDK will get the configuration from the server. For now, it contains:

* eyeVerifyLicense;

3.

void onWorkflowAbort(String workflowType, String userName);

The method 'onWorkflowAbort()' is called when workflow was aborted after getting an error or after calling method com.Avarta.integrationlibrary.data.api.IntegrationApiManager#cancelWorkflow(Context context)

4.

void onContinueEnroll(String workflowType, String requestedActivity);

The method 'onContinueEnroll()' is called when SDK gets the new process activity from the server and library`s user should handle it and pass back the result via

com.Avarta.integrationlibrary.data.api.IntegrationApiManager#processActivityData(Context context, final String activityCode, String activityData)

5.

void onWorkflowCompleted(String workflowType, String userName, boolean isDeviceBased);

This method will fire when workflow process is completed successfully.

isDeviceBased - true if the workflow was started from the device, false if started from the web application