IDMISSION
empowering identity

eVolv Android SDK Implementation Guide

7.2.x

August 2020

eVolv Android SDK Implementation Guide

Table of Contents

1 Introduction

This document introduces the eVolv Android SDK within the IDmission product suite. Developers, project managers and QA testers should reference this document for information on configuration and use of the eVolv SDK on the Android platform. IDmission recommends reading this entire document as an implementation guide to fully understand the eVolv SDK functionality and its respective capabilities.

This document details processes and procedures for embedding the eVolv SDK into your host application and utilizing its current features. For additional eVolv SDK support, please contact our Customer Support team at support@idmission.com or submit a service request via the My eVolv Portal.

2 Overview and Key Features

2.1 Overview

The IDmission eVolv SDK is a comprehensive toolkit that enables the use of any combination of factors of identity to complete digital transformation goals. The goal of the eVolv SDK is to offer seamless integration into an existing digital paradigm where the end-to-end customer experience is still owned and managed in-house.

The main features supported in this SDK are:

- ID Capture and Scanning with OCR
- Biometric Capture and Verification for:
  - Face (with Liveness detection)
  - Voice
  - Fingerprint
- **Customer**
  - Enrollment
  - Search
  - Verification
- **Employee**
  - Enrollment
  - Search
  - Verification

Note: When using the eVolv SDK, you do not need to create a request for XML; it is automatically generated by the SDK based on the function that you are calling.

2.2 Service IDs

Each of these features are available through various Service IDs. A full list of current available services is provided in the table below. The most commonly used Services from this list are included in Section 4 of this document.

Table 1: Service ID

Service ID	Service Description	Included Features
20	ID Validation Only	Capturing ID document(s), validate the ID and extract data from it.
10	ID Validation + Face Match	Capturing ID document(s), validate the ID and extract data from it. Additionally, capture a selfie and match it against the photo on the ID.
50	ID Validation + Face Match w/Customer Enrollment	Capturing ID document(s), validate that ID and extract data from it. Additionally, capture a selfie of the customer and match it against the photo on the ID, then enroll all captured biometrics with the customer if the match is positive.
55	ID Validation + Face Match w/Employee Enrollment	Capturing ID document(s), validate the ID and extract data from it. Additionally, capture a selfie of the employee and match it against the photo on the ID, then enroll all captured biometrics with the employee if the match is positive.
155	ID Validation + Video Match	Capturing ID document(s), validate the ID and extract data from it. Additionally, capture a video stream and use it to match the face against the ID.
160	ID Validation + Video Match w/Customer Enrollment	Capturing ID document(s) for a customer, validate the ID and extract data from it. Additionally, capture a video stream and use it to match the face against the ID, then enroll all captured biometrics with the customer.
165	ID Validation + Video Match w/Employee Enrollment	Capturing ID document(s) for an employee, validate the ID and extract data from it. Additionally, capture a video stream and use it to match the face against the ID, then enroll all captured biometrics with the employee.
175	Customer Enrollment w/ Biometrics	Capture customer data and biometrics and enroll into the customer database. No ID documents are captured.
180	Employee Enrollment w/Biometrics	Capture employee data and biometrics and enroll into the employee database. No ID documents are captured.
70	Customer Update	Capture of any new/updated customer data to update the previously enrolled customer record.
75	Employee Update	Capture of any new/updated customer data to update the previously enrolled employee record.

105	Customer Verification	Capture of biometric data to verify against a previously enrolled customer.
305	Employee Verification	Capture of biometric data to verify against a previously enrolled employee.
185	Identify Customer with Biometrics	Capture biometric data for a Customer (face, fingerprint, voice or iris) and search the customer DB to see if its already enrolled.
190	Identify Employee with Biometrics	Capture biometric data for a Customer (face, fingerprint, voice or iris) and search the customer DB to see if its already enrolled.
186	Customer Search	Capture customer data (non-biometric) to search against previously enrolled customers in the database. If a match is found, the data associated with that customer is returned.
191	Employee Search	Capture employee data (non-biometric) to search against previously enrolled employees in the database. If a match is found, the data associated with that employee is returned.
500	Video Conference	Initiate a video conference session with a back office operator that will be recorded for audit purposes.
505	IDV + Video Conference Match	Capturing ID document(s), validate the ID and extract data from it. Additionally, initiate a video conference session with a back office operator that will be recorded for audit purposes. A photo image from the video conference will be extracted and used to match the face against the ID.
510	IDV + Video Conference Match + Customer Enrollment	Capturing ID document(s) for a customer, validate the ID and extract data from it. Additionally, initiate a video conference session with a back office operator that will be recorded for audit purposes. A photo image from the video conference will be extracted and used to match the face against the ID, then enroll all captured biometrics with the customer.
515	IDV + Video Conference Match + Employee Enrollment	Capturing ID document(s) for an employee, validate the ID and extract data from it. Additionally, initiate a video conference session with a back office operator that will be recorded for audit purposes. A photo image from the video conference will be extracted and used to match the face against the ID, then enroll all captured biometrics with the employee.
620	ID Data Extraction	Capturing ID document(s) and using ONLY the barcode, MRZ or QR code to extract data to pre-populate data into an application or online form.

660	Offline Liveness Detection	Capturing a selfie to detect liveness without having to send the images to the server. A short video is captured of an individual and used to check for liveness.

2.3 ID Validation Manual Review

For each of the services above that support ID Validation, an optional request can be made through this SDK for a manual review to be performed in the event the ID does not pass our automated system checks. There is an additional fee for this service if performed. There is also an additional POST API Request that contains the results of that manual review once completed. You will have to provide a URL where that information will be posted to. The details for requesting this manual review can be found in the sections that outline how to call the function for processing images and the POST API Request Details.

3 Minimum Requirements and Initial Setup

The minimum requirements for utilizing this SDK are:

- Android 4.0 or higher
- Active internet connection

The following permissions and features should be present in your **AndroidManifest.xml**:

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.RECORD_VIDEO" />
<uses-feature android:name="android.hardware.camera" android:required="false"
/>
<uses-feature android:name="android.hardware.camera.autofocus"
android:required="false" />
<uses-feature android:name="android.hardware.camera.flash"
android:required="false" />
<uses-feature android:name="android.hardware.camera.front"
android:required="false"/>
```

Note: If you are utilizing a device that has an Android Marshmallow (6.0) version, you will need to acquire `android.permissions.CAMERA`, `android.permission.WRITE_EXTERNAL_STORAGE` and `android.permission.READ_EXTERNAL_STORAGE` dynamically before initializing the SDK.

Use `getRequiredPermissions` to get a list of required dangerous permissions.

The SDK requires there to be following declarations in your **AndroidManifest.xml**. In Manifest tag:

```
xmlns:tools="http://schemas.android.com/tools"
```

In Application tag:

```
android:theme="@style/AppTheme"
tools:replace="android:icon,android:theme"
```

3.1 Dependencies

1. Navigate to **File** > **Import** > **New Module**
2. Click on Import **.JAR/.AAR** Package
3. Add **AAR** file path in **File name**
4. Add Firebase to your app
5. Open the **Firebase Assistant** in Android Studio
6. Click **Tools** > **Firebase** to open the Assistant window
7. Click to expand one of the listed features (e.g., Analytics)
8. Click the **Connect to Firebase** button to connect to Firebase
9. Add the necessary code to your app
10. For more details please reference: https://firebase.google.com/docs/android/setup

Table 2: Dependencies

Dependency	Description
`idm-imgproc-#.#.##.#@aar` (e.g. `idm-imgproc-1.1.99.15.aar`)	Image Processing Library (Use latest released version)
`opencv_#.#@aar` (e.g. `opencv_3.2.aar`)	OpenCV Library (Use latest released version)
`card.io-#.#.#.aar` (e.g. `card.io-5.4.2.aar`)	CardIO Library (Use latest released version)

`compile 'com.google.android.gms:play-services-vision:16.2.0'`
`compile 'com.google.android.gms:play-services-location:16.0.0'`
Add above dependency in `build.gradle` of your main project.
	Support library for face-detection

`compile 'com.google.firebase:firebase-ml-vision:18.0.1'`
`compile 'com.google.firebase:firebase-ml-vision-face-model:17.0.2'`
Add above dependency in `build.gradle` of your main project.
	Support library for face-detection

`apply plugin: 'com.google.gms.google-services'` Add above line as the last line in `build.gradle` of your main project.	Support library for face-detection

Add below code in `build.gradle`
 	
`ndk {`
`   abiFilters 'armeabi-v7a', 'arm64-v8a'`
`}`
 	
`apply plugin: 'com.google.gms.google-services'`	

`libjinglewrapper.aar` (e.g `libjinglewrapper.aar`)	Support library for video conferencing
`webrtcpeer-android`	Support library project for video conferencing. `libjinglewrapper.aar` will be added as a module in this project.
`compile 'com.neovisionaries:nv-websocket-client:1.30'`	Support library for video conferencing
`compile 'io.fotoapparat:fotoapparat:2.6.1'`	Support library for camera frame processing. (Note: This dependency should be added for release version 6.4.4.2 or above) NOTE: 6.4.4.2 and above support AndroidX

Recommended IDE version and gradle plugins:
 	
Android studio 3.2.1
 	
Gradle:
`classpath 'com.android.tools.build:gradle:3.2.0'`
 	
`distributionUrl=https\://services.gradle.org/distributions/gradle-4.6-all.zip`	

If you are using SDK version 7.1.1.22 or above, you need to update `build.gradle` with below additional node.
`android {`
`   aaptOptions {`
`        noCompress "tflite"`
`    }`
`    compileOptions {`
`          sourceCompatibility = '1.8'`
`          targetCompatibility = '1.8'`
`    }`
`}`
Additionally, you need to add the following dependencies.
`implementation 'org.tensorflow:tensorflow-lite:0.0.0-nightly'`
`implementation 'com.google.firebase:firebase-storage:19.1.0'`
Also, we suggest you update the existing dependencies with below version.
`implementation 'com.google.android.gms:play-services-location:17.0.0'`
`implementation 'com.google.firebase:firebase-ml-vision:24.0.1'`
`implementation 'com.google.firebase:firebase-ml-vision-face-model:19.0.0'`
`implementation 'com.google.firebase:firebase-core:17.2.1'`
You can also use the Android studio 3.5.1 and above with gradle 3.5.3 and **gradle-5.4.1-all.zip**
	`build.gradle` changes for 7.1.1.22 and above SDK version.

Check the Android Studio sample project to learn the most common use.

3.2 Response Status Codes

Below is the list of all status codes which the SDK will return in the response.

Table 3: Response Status Codes

Status Code	Status Message
0	Image captured successfully
1	SDK not initialized
2	Image not captured
3	No internet connection
4	Poor internet connectivity
5	Live face not detected
6	Request parameter not available
7	Invalid request parameter
8	Some error occurred, please try again
9	Card not detected
10	Device not supported.
14	Original call is in process.
15	Connect fingerprint device
16	Play service exception
21	Write permission require
500	Error in processing your request. Internal Server Error
502	Error in processing your request. Bad gateway
503	Error in processing your request. Please try after some time
504	Error in processing your request. Gateway timeout
404	Resource URL could not find. Please correct URL
100	Socket connection exception. {connect timeout}
101	Timeout for Response
102	Error in processing your request. {Unhandled Exception message}
103	Error in processing your request. Empty response received.
104	Error in processing your request. Unknown Host Exception.

3.3 Initializing Labels

```
ImageProcessingSDK.initializeLabels(HashMap<String, String> englishLabelMap,
HashMap<String, String> spanishLabelMap);
```

Table 4: Label Parameters

Parameter	Type	Default	Description
`englishLabelMap`	HashMap	Blank Map	You can add following key with your own message in value to display on UI, this map is for changing English labels. "align_document_img_capture" "subject_is_too_dark_img_capture" "out_of_focus_img_capture" "too_much_glare_img_capture" "subject_is_too_dark_fc_detect" "out_of_focus_fc_detect" "move_camera_closer_to_your_face" "camera_movement_fc_detect" "keep_face_steady" "smile_please" "face_detected" "light" "focus" "glare" "smile"
`spanishLabelMap`	HashMap	Blank Map	Keys are same as above. This map is for changing Spanish labels.

Table 5: Label definitions

Label key	Used for	Label default value
`align_document_img_capture`	For aligning the ID/Document in the frame.	Align document inside the green corners and wait, we'll take the photo for you.
`subject_is_too_dark_img_capture`	For letting user know the low light while ID/Document capture.	It's too dark to take a good image. Find a place with better lighting.
`out_of_focus_img_capture`	For letting user know out of focus while ID/Document capture.	Tap screen to focus or move camera closer/away
`too_much_glare_img_capture`	For letting user know of too much glare while ID/Document capture.	Too much light, move document away from direct light
`subject_is_too_dark_fc_detect`	For letting user know the low light while Live face capture.	It's too dark to take a good image. Find a place with better lighting.

`out_of_focus_fc_detect`	For letting user know out of focus while Live face capture.	Tap screen to focus or move camera closer/away
`move_camera_closer_to_your_face`	For letting user know if face is too far behind while Live face capture.	Move camera closer to face and look towards light.
`camera_movement_fc_detect`	For letting user know to keep face steady and smile while Live face capture.	Hold camera steady.
`keep_face_steady`	For letting user know to keep face steady and smile while Live face capture.	Keep your face steady
`smile_please`	For letting user know to keep face steady and smile while Live face capture.	Hold camera steady and smile please.
`face_detected`	For letting user know Live face is captured.	Face detected
`light`	For letting user know Light %. (only visible in debug mode)	Light %
`focus`	For letting user know Focus %. (only visible in debug mode)	Focus %
`glare`	For letting user know glare %. (only visible in debug mode)	Glare %
`smile`	For letting user know smile %. (only visible in debug mode)	Smile %
`page_title_image_capture`	For setting title text for ID/Document capture.	Capturing identification
`page_title_face_detection`	For setting title text for Live face capture.	Detecting face

3.4 Customizing the User Interface

To customize the UI component position on ID capture and face detection screen.

```
@Deprecated
ImageProcessingSDK.customizeUserInterface(boolean faceTitleOnTop, boolean
faceHintMessageOnTop, boolean faceHintIconOnTop, boolean
faceTitleImageOnTop,String idTitleAlignment,String idHintMsgAlignment,String
idTitleImgMsgAlignment)

@Deprecated
ImageProcessingSDK.customizeUserInterface(boolean faceTitleOnTop, boolean
hideFaceTitle,
     boolean faceHintMessageOnTop, boolean hideFacehintMsg,
     boolean faceHintIconOnTop, boolean hideFacehintIcon,
     boolean faceTitleImageOnTop, boolean hideFacetitleImage,
     String idTitleAlignment, boolean hideIdCaptureTitle,
     String idHintMsgAlignment, boolean hideIdCaptureHintMsg,
     String idTitleImgMsgAlignment, boolean hideIdCaptureTitleImg)
```

Table 6: Custom UI Parameters

Parameter	Type	Default	Description
`faceTitleOnTop`	boolean	false	Face title message
`faceHintMessageOnTop`	boolean	true	Hint message
`faceHintIconOnTop`	boolean	true	Hint icons (Light and Face)
`faceTitleImageOnTop`	boolean	true	Face title image
`idTitleAlignment`	String (Top, Center, Bottom)	Top	ID capture title
`idHintMsgAlignment`	String (Top, Center, Bottom)	Center	Hint message
`idTitleImgMsgAlignment`	String (Top, Center, Bottom)	Bottom	ID capture title image
`hideFaceTitle`	boolean	false	Hide face title (Text)
`hideFacehintMsg`	boolean	false	Hide face hint message (Text)
`hideFacehintIcon`	boolean	false	Hide face hint icon (Image)
`hideFacetitleImage`	boolean	false	Hide face title image (Image)
`hideIdCaptureTitle`	boolean	false	Hide ID capture title (Text)
`hideIdCaptureHintMsg`	boolean	false	Hide ID capture hint message (Text)
`hideIdCaptureTitleImg`	boolean	false	Hide ID capture title image (Image)

Above UI customization methods are **Deprecated** instead of that you can use below method and customize UI for all the feature in one call. Sample UI configuration json is available on the following URL:
https://drive.google.com/file/d/1Epgl6vqr7WEtdrZhPjK1Nasy15iaAI7f/view?usp=sharing

```
ImageProcessingSDK.customizeUserInterface(String uiConfigurationJsonString);
```

Also find the supported JSON below with description:

Table 7: Single Call UI Customization

```
{
  "id_capture_front": {
    "id_show_instruction": "N",
    "id_outline_color": "FFAD36",
    "id_outline_color_alpha": "1",
    "id_detected_id_outline_color": "6EB24C",
    "id_detected_id_outline_color_alpha": "1",
    "id_outside_outline_color": "487D95",
    "id_outside_outline_color_aplha": "1",
    "id_detected_id_outside_outline_color": "487D95",
    "id_detected_id_outside_outline_color_alpha": "1",
    "id_back_button_color": "FFAD36",
    "id_back_button_color_alpha": "1",
    "id_retry_button_color": "FFAD36",
    "id_retry_button_color_alpha": "1",
    "id_retry_button_border_color": "FFAD36",
    "id_retry_button_border_color_alpha": "1",
    "id_confirm_button_color": "FFAD36",
    "id_confirm_button_color_alpha": "1",
    "id_confirm_button_background_color": "FFFFFF",
    "id_confirm_button_background_color_alpha": "1",
    "id_instruction_button_color": "FFFFFF",
    "id_instruction_button_alpha": "1",
    "id_instruction_button_txt_color": "487D95",
    "id_instruction_button_txt_alpha": "1",
    "id_label_text_color": "FFFFFF",
    "id_label_text_alpha": "1",
    "id_label_text_typeface_type": "DEFAULT",
    "id_label_text_typeface_style": "NORMAL",
    "id_label_text_size": "",
    "id_header_text_label_color": "FFFFFF",
    "id_header_text_label_alpha": "1",
    "id_header_text_typeface_type": "DEFAULT",
    "id_header_text_typeface_style": "NORMAL",
    "id_header_text_label_size": "",
    "id_capture_border_style": "Thick",
    "id_title_img_bitmap_base64": "",
    "id_instruction_img_resource_id": "",
    "id_title_label_alignment": "Top",
    "id_hint_message_alignment": "Center",
    "id_title_image_alignment": "Bottom",
    "id_hide_id_title_label": "N",
    "id_hide_id_hint_message": "N",
    "id_hide_id_title_image": "N",
    "id_enable_label_shadow": "Y",
    "id_capture_button_color": "FFFFFF",
    "id_capture_button_alpha": "1",
    "labels": {
      "light": "Light",
      "focus": "Focus",
      "glare": "Glare",
      "align_document_img_capture": "Align document inside the green corners and
wait, we'll take the photo for you.",
      "subject_is_too_dark_img_capture": "It's too dark to take a good image. Find
a place with better lighting.",
      "out_of_focus_img_capture": "Tap screen to focus or move camera closer/away",
      "too_much_glare_img_capture": "Too much light, move document away from direct
light",
      "page_title_image_capture": "Capturing identification"
    }
  },
  "id_capture_back": {
    "id_show_instruction": "N",
    "id_outline_color": "FFAD36",
    "id_outline_color_alpha": "1",
    "id_detected_id_outline_color": "6EB24C",
    "id_detected_id_outline_color_alpha": "1",
    "id_outside_outline_color": "487D95",
    "id_outside_outline_color_aplha": "1",
    "id_detected_id_outside_outline_color": "487D95",
    "id_detected_id_outside_outline_color_alpha": "1",
    "id_back_button_color": "FFAD36",
    "id_back_button_color_alpha": "1",
    "id_retry_button_color": "FFAD36",
    "id_retry_button_color_alpha": "1",
    "id_retry_button_border_color": "FFAD36",
    "id_retry_button_border_color_alpha": "1",
    "id_confirm_button_color": "FFAD36",
    "id_confirm_button_color_alpha": "1",
    "id_confirm_button_background_color": "FFFFFF",
    "id_confirm_button_background_color_alpha": "1",
    "id_instruction_button_color": "FFFFFF",
    "id_instruction_button_alpha": "1",
    "id_instruction_button_txt_color": "487D95",
    "id_instruction_button_txt_alpha": "1",
    "id_label_text_color": "FFFFFF",
    "id_label_text_alpha": "1",
    "id_label_text_typeface_type": "DEFAULT",
    "id_label_text_typeface_style": "NORMAL",
    "id_label_text_size": "",
    "id_header_text_label_color": "FFFFFF",
    "id_header_text_label_alpha": "1",
    "id_header_text_typeface_type": "DEFAULT",
    "id_header_text_typeface_style": "NORMAL",
    "id_header_text_label_size": "",
    "id_capture_border_style": "Thick",
    "id_title_img_bitmap_base64": "",
    "id_instruction_img_resource_id": "",
    "id_title_label_alignment": "Top",
    "id_hint_message_alignment": "Center",
    "id_title_image_alignment": "Bottom",
    "id_hide_id_title_label": "N",
    "id_hide_id_hint_message": "N",
    "id_hide_id_title_image": "N",
    "id_enable_label_shadow": "Y",
    "id_capture_button_color": "FFFFFF",
    "id_capture_button_alpha": "1",
    "labels": {
      "light": "Light",
      "focus": "Focus",
      "glare": "Glare",
      "align_document_img_capture": "Align document inside the green corners and
wait, we'll take the photo for you.",
      "subject_is_too_dark_img_capture": "It's too dark to take a good image. Find
a place with better lighting.",
      "out_of_focus_img_capture": "Tap screen to focus or move camera closer/away",
      "too_much_glare_img_capture": "Too much light, move document away from direct
light",
      "page_title_image_capture": "Capturing identification"
    }
  },
  "document_capture": {
    "id_show_instruction": "N",
    "id_outline_color": "FFAD36",
    "id_outline_color_alpha": "1",
    "id_detected_id_outline_color": "6EB24C",
    "id_detected_id_outline_color_alpha": "1",
    "id_outside_outline_color": "487D95",
    "id_outside_outline_color_aplha": "1",
    "id_detected_id_outside_outline_color": "487D95",
    "id_detected_id_outside_outline_color_alpha": "1",
    "id_back_button_color": "FFAD36",
    "id_back_button_color_alpha": "1",
    "id_retry_button_color": "FFAD36",
    "id_retry_button_color_alpha": "1",
    "id_retry_button_border_color": "FFAD36",
    "id_retry_button_border_color_alpha": "1",
    "id_confirm_button_color": "FFAD36",
    "id_confirm_button_color_alpha": "1",
    "id_confirm_button_background_color": "FFFFFF",
    "id_confirm_button_background_color_alpha": "1",
    "id_instruction_button_color": "FFFFFF",
    "id_instruction_button_alpha": "1",
    "id_instruction_button_txt_color": "487D95",
    "id_instruction_button_txt_alpha": "1",
    "id_label_text_color": "FFFFFF",
    "id_label_text_alpha": "1",
    "id_label_text_typeface_type": "DEFAULT",
    "id_label_text_typeface_style": "NORMAL",
    "id_label_text_size": "",
    "id_header_text_label_color": "FFFFFF",
    "id_header_text_label_alpha": "1",
    "id_header_text_typeface_type": "DEFAULT",
    "id_header_text_typeface_style": "NORMAL",
    "id_header_text_label_size": "",
    "id_capture_border_style": "Thick",
    "id_title_img_bitmap_base64": "",
    "id_instruction_img_resource_id": "",
    "id_title_label_alignment": "Top",
    "id_hint_message_alignment": "Center",
    "id_title_image_alignment": "Bottom",
    "id_hide_id_title_label": "N",
    "id_hide_id_hint_message": "N",
    "id_hide_id_title_image": "N",
    "id_enable_label_shadow": "Y",
    "id_capture_button_color": "FFFFFF",
    "id_capture_button_alpha": "1",
    "labels": {
      "light": "Light",
      "focus": "Focus",
      "glare": "Glare",
      "align_document_img_capture": "Align document inside the green corners
and wait, we'll take the photo for you.",
      "subject_is_too_dark_img_capture": "It's too dark to take a good image.
Find a place with better lighting.",
      "out_of_focus_img_capture": "Tap screen to focus or move camera
closer/away",
      "too_much_glare_img_capture": "Too much light, move document away from
direct light",
      "page_title_image_capture": "Capturing identification"
    }
  },
  "snippet_capture": {
    "id_show_instruction": "N",
    "id_outline_color": "FFAD36",
    "id_outline_color_alpha": "1",
    "id_detected_id_outline_color": "6EB24C",
    "id_detected_id_outline_color_alpha": "1",
    "id_outside_outline_color": "487D95",
    "id_outside_outline_color_aplha": "1",
    "id_detected_id_outside_outline_color": "487D95",
    "id_detected_id_outside_outline_color_alpha": "1",
    "id_back_button_color": "FFAD36",
    "id_back_button_color_alpha": "1",
    "id_retry_button_color": "FFAD36",
    "id_retry_button_color_alpha": "1",
    "id_retry_button_border_color": "FFAD36",
    "id_retry_button_border_color_alpha": "1",
    "id_confirm_button_color": "FFAD36",
    "id_confirm_button_color_alpha": "1",
    "id_confirm_button_background_color": "FFFFFF",
    "id_confirm_button_background_color_alpha": "1",
    "id_instruction_button_color": "FFFFFF",
    "id_instruction_button_alpha": "1",
    "id_instruction_button_txt_color": "487D95",
    "id_instruction_button_txt_alpha": "1",
    "id_label_text_color": "FFFFFF",
    "id_label_text_alpha": "1",
    "id_label_text_typeface_type": "DEFAULT",
    "id_label_text_typeface_style": "NORMAL",
    "id_label_text_size": "",
    "id_header_text_label_color": "FFFFFF",
    "id_header_text_label_alpha": "1",
    "id_header_text_typeface_type": "DEFAULT",
    "id_header_text_typeface_style": "NORMAL",
    "id_header_text_label_size": "",
    "id_capture_border_style": "Thick",
    "id_title_img_bitmap_base64": "",
    "id_instruction_img_resource_id": "",
    "id_title_label_alignment": "Top",
    "id_hint_message_alignment": "Center",
    "id_title_image_alignment": "Bottom",
    "id_hide_id_title_label": "N",
    "id_hide_id_hint_message": "N",
    "id_hide_id_title_image": "N",
    "id_enable_label_shadow": "Y",
    "id_capture_button_color": "FFFFFF",
    "id_capture_button_alpha": "1",
    "labels": {
      "light": "Light",
      "focus": "Focus",
      "glare": "Glare",
      "align_document_img_capture": "Align document inside the green corners and
wait, we'll take the photo for you.",
      "subject_is_too_dark_img_capture": "It's too dark to take a good image. Find
a place with better lighting.",
      "out_of_focus_img_capture": "Tap screen to focus or move camera closer/away",
      "too_much_glare_img_capture": "Too much light, move document away from direct
light",
      "page_title_image_capture": "Capturing identification"
    }
  },
  "face_capture": {
    "fd_show_preview_screen": "N",
    "fd_show_instruction_screen": "Y",
    "fd_outline_color": "FFAD36",
    "fd_outline_color_alpha": "1",
    "fd_detected_face_outline_color": "6EB24C",
    "fd_detected_face_outline_color_alpha": "1",
    "fd_outside_face_outline_color": "487D95",
    "fd_outside_face_outline_color_alpha": "1",
    "fd_detected_outside_face_outline_color": "487D95",
    "fd_detected_outside_face_outline_color_alpha": "1",
    "fd_back_button_color": "FFAD36",
    "fd_back_button_color_alpha": "1",
    "fd_retry_button_color": "FFAD36",
    "fd_retry_button_color_alpha": "1",
    "fd_retry_button_border_color": "FFAD36",
    "fd_retry_button_border_color_alpha": "1",
    "fd_confirm_button_color": "FFAD36",
    "fd_confirm_button_color_alpha": "1",
    "fd_confirm_button_background_color": "FFFFFF",
    "fd_confirm_button_background_color_alpha": "1",
    "fd_instruction_button_color": "FFFFFF",
    "fd_instruction_button_alpha": "1",
    "fd_instruction_button_txt_color": "487D95",
    "fd_instruction_button_txt_alpha": "1",
    "fd_label_text_color": "FFFFFF",
    "fd_label_text_alpha": "1",
    "fd_label_text_typeface_type": "DEFAULT",
    "fd_label_text_typeface_style": "NORMAL",
    "fd_label_text_size": "",
    "fd_header_text_label_color": "FFFFFF",
    "fd_header_text_label_alpha": "1",
    "fd_header_text_typeface_type": "DEFAULT",
    "fd_header_text_typeface_style": "NORMAL",
    "fd_header_text_label_size": "",
    "fd_title_img_bitmap_base64": "",
    "fd_instruction_img_resource_id": "",
    "fd_face_contours": "Zero",
    "fd_face_title_label_on_top": "N",
    "fd_face_hint_message_on_top": "Y",
    "fd_face_hint_icon_on_top": "Y",
    "fd_title_image_on_top": "Y",
    "fd_hide_face_title_lable": "N",
    "fd_hide_face_hint_message": "N",
    "fd_hide_face_hint_icon": "N",
    "fd_hide_title_image": "N",
    "fd_show_custom_ui":"N",
    "fd_face_outline_image_id":"",
    "fd_outside_face_outline_image_id":"",
    "fd_face_outline_progress_images":"",
    "fd_face_outline_progress_images_delay":"500",
    "fd_face_turn_arrow_list":"",
    "fd_toggle_camera_button_icon":"",
    "labels": {
      "too_much_glare_face_capture": "Too much light, move face away from direct
light",
      "light": "Light",
      "focus": "Focus",
      "smile": "Smile",
      "out_of_focus_fc_detect": "Tap screen to focus or move camera closer/away",
      "subject_is_too_dark_fc_detect": "It's too dark to take a good image. Find a
place with better lighting.",
      "move_camera_closer_to_your_face": "Move camera closer to face and look
towards light.",
      "camera_movement_fc_detect": "Hold camera steady.",
      "smile_please": "Hold camera steady and smile please.",
      "face_detected": "Face detected",
      "keep_face_steady": "Keep your face steady",
      "page_title_face_detection": "Detecting face",
      "camera_error": "Unable to start the camera, please restart the device.",
      "turn_left": "Turn your head left",
      "turn_right": "Turn your head right",
      "move_up": "Move your head up",
      "move_down": "Move your head down",
      "look_straight": "Please look straight into the phone"
    }
  },
  "camera_finger_capture": {
    "cfc_label_text_typeface_type": "DEFAULT",
    "cfc_label_text_typeface_style": "NORMAL",
    "cfc_label_text_color": "FFFFFF",
    "cfc_label_text_color_alpha": "1",
    "cfc_show_instruction_screen": "Y",
    "cfc_instruction_button_color": "FFFFFF",
    "cfc_instruction_button_alpha": "1",
    "cfc_instruction_button_txt_color": "487D95",
    "cfc_instruction_button_txt_alpha": "1",
    "cfc_finger_capture_background_color_alpha": "1",
    "cfc_back_button_color": "FFAD36",
    "cfc_back_button_color_alpha": "1",
    "labels": {
      "camera_finger_capture_title": "Capture Fingerprint",
      "move_closer": "Move closer",
      "move_away": "Move away",
      "incorrect_hand": "Incorrect hand",
      "hold_steady": "Hold Steady",
      "capturing_detail": "Capturing Detail",
      "finger_too_close": "TOO CLOSE",
      "finger_too_far": "TOO FAR"
    }
  },
  "barcode_capture": {
    "barcode_capture_portrait": "N",
    "labels": {
     "barcode_text_message_footer": "Barcode Scan"
    }
  },
  "voice_recording": {
     "voice_button_color": "FFFFFF",
     "voice_button_color_alpha": "1",
     "voice_background_color": "000000",
     "voice_background_color_alpha": "1",
     "voice_title_label_on_top": "Y",
     "voice_display_position": "Center",
     "voice_auto_play": "Y",
     "voice_recording_time": "15",
     "voice_text_label_color": "FFFFFF",
     "voice_text_label_color_alpha": "1",
     "voice_title_label_color": "FFFFFF",
     "voice_title_label_color_alpha": "1",
     "voice_title_label_size": "18",
     "voice_text_label_size": "18",
     "voice_counter_label_size": "38",
     "labels": {
       "voice_title_label_message": "",
       "voice_start": "Start",
       "voice_play": "Play",
       "voice_pause": "Pause",
       "voice_stop": "Stop",
       "voice_delete": "Delete",
       "voice_save": "Save",
       "voice_retry": "Retry"
     }
    }
}
```

Table 8: Custom ID Configurations

ID Front/Back/Doc./Snippet Config	Description
id_show_instruction	To display the instruction screen.
id_outline_color id_outline_color_alpha	ID capture outline (border) color and transparency configuration. -Color is hex string (e.g. FFFFFF). -Alpha value is (1 to 100 where bigger value means more transparency). **Note:** Above color and Alpha value definition is same for another color/alpha as well
id_detected_id_outline_color id_detected_id_outline_color_alpha	Outline (Border) color and transparency to be shown when ID is detected.
id_outside_outline_color id_outside_outline_color_aplha	Background color and transparency of ID capture screen.
id_detected_id_outside_outline_color id_detected_id_outside_outline_color_alpha	Background color and transparency to be shown when ID is detected.
id_back_button_color id_back_button_color_alpha	Back/Exit button color and transparency (ID capture)
id_retry_button_color id_retry_button_color_alpha	Retry button color and transparency (Preview screen)
id_retry_button_border_color id_retry_button_border_color_alpha	Retry button border color and transparency (Preview screen)
id_confirm_button_color id_confirm_button_color_alpha	Confirm/Done button color and transparency (Preview screen)
id_confirm_button_background_color id_confirm_button_background_color_alpha	Confirm/Done background color and transparency (Preview screen)
id_instruction_button_color id_instruction_button_alpha	Continue button color and transparency (Instruction screen)
id_instruction_button_txt_color id_instruction_button_txt_alpha	Continue button text color and transparency (Instruction screen)
id_label_text_color id_label_text_alpha	Color and Transparency of all the labels on instruction, ID capture and preview screen.
id_label_text_typeface_type	Following values are supported for label typeface. DEFAULT, DEFAULT_BOLD, SANS_SARIF, SERIF, MONOSPACE

id_label_text_typeface_style	Following values are supported for label typeface style. NORMAL, BOLD, ITALIC, BOLD_ITALIC
id_label_text_size	Label text size.
id_header_text_label_color id_header_text_label_alpha	Color and Transparency of header text on ID capture screen
id_header_text_typeface_type	Following values are supported for label typeface. DEFAULT, DEFAULT_BOLD, SANS_SARIF, SERIF, MONOSPACE
id_header_text_typeface_style	Following values are supported for label typeface style. NORMAL, BOLD, ITALIC, BOLD_ITALIC
id_header_text_label_size	Header text size.
id_capture_border_style	ID Capture border style support below values. Thick, Thin
id_title_img_bitmap_base64	If you would like to show custom title image/logo on ID capture screen you can add base64 string of bitmap image here.
id_instruction_img_resource_id	If you would like to show custom instruction image pass image resource ID here.
id_title_label_alignment	ID title label (String) position can be configured using below configuration. Top, Center, Bottom
id_hint_message_alignment	ID capture hint/instruction message (String) position can be configured using below configuration. Top, Center, Bottom
id_title_image_alignment	ID title image position can be configured using below configuration. Top, Center, Bottom
id_hide_id_title_label	Hide ID title label (Y/N)
id_hide_id_hint_message	Hide ID hint/instruction label (Y/N)
id_hide_id_title_image	Hide ID title image (Y/N)
labels	Currently following labels are shown on **ID Capture** screen, that can be customized with your own custom message. light focus glare align_document_img_capture subject_is_too_dark_img_capture out_of_focus_img_capture too_much_glare_img_capture page_title_image_capture

Table 9: Custom Face Capture Configurations

Face capture config	Description
`fd_show_preview_screen`	To display the preview screen after capturing face.
`fd_show_instruction_screen`	To display the instruction screen.
`fd_outline_color` `fd_outline_color_alpha`	Face capture outline (face shape outline) color and transparency configuration. -Color is hex string (e.g. `FFFFFF`). -Alpha value is (1 to 100 where bigger value means more transparency). **Note:** Above color and Alpha value definition is same for another color/alpha as well
`fd_detected_face_outline_color` `fd_detected_face_outline_color_alpha`	Outline (Border) color and transparency to be shown when face is detected.
`fd_outside_face_outline_color` `fd_outside_face_outline_color_alpha`	Background color and transparency of face capture screen.
`fd_detected_outside_face_outline_color` `fd_detected_outside_face_outline_color_alpha`	Background color and transparency to be shown when face is detected.
`fd_back_button_color` `fd_back_button_color_alpha`	Back/Exit button color and transparency (Face capture)
`fd_retry_button_color` `fd_retry_button_color_alpha`	Retry button color and transparency (Preview screen)
`fd_retry_button_border_color` `fd_retry_button_border_color_alpha`	Retry button border color and transparency (Preview screen)
`fd_confirm_button_color` `fd_confirm_button_color_alpha`	Confirm/Done button color and transparency (Preview screen)
`fd_confirm_button_background_color` `fd_confirm_button_background_color_alpha`	Confirm/Done background color and transparency (Preview screen)
`fd_instruction_button_color` `fd_instruction_button_alpha`	Continue button color and transparency (Instruction screen)
`fd_instruction_button_txt_color` `fd_instruction_button_txt_alpha`	Continue button text color and transparency (Instruction screen)
`fd_label_text_color` `fd_label_text_alpha`	Color and Transparency of all the labels on instruction, face-capture and preview screen.
`fd_label_text_typeface_type`	Following values are supported for label typeface. `DEFAULT, DEFAULT_BOLD, SANS_SARIF, SERIF, MONOSPACE`

fd_label_text_typeface_style	Following values are supported for label typeface style. NORMAL, BOLD, ITALIC, BOLD_ITALIC
fd_label_text_size	Label text size.
fd_header_text_label_color fd_header_text_label_alpha	Color and Transparency of header text on face capture screen
fd_header_text_typeface_type	Following values are supported for label typeface. DEFAULT, DEFAULT_BOLD, SANS_SARIF, SERIF, MONOSPACE
fd_header_text_typeface_style	Following values are supported for label typeface style. NORMAL, BOLD, ITALIC, BOLD_ITALIC
fd_header_text_label_size	Header text size.
fd_title_img_bitmap_base64	If you would like to show custom title image/logo on face capture screen you can add base64 string of bitmap image here.
fd_instruction_img_resource_id	If you would like to show custom instruction image pass image resource ID here.
fd_face_contours	Face contours count can be customized using below configuration. Low, Medium, ALL, Zero
fd_face_title_label_on_top	Face title label on top (Y/N)
fd_face_hint_message_on_top	Face hint/instruction message on top (Y/N)
fd_title_image_on_top	Face title image on top (Y/N)
fd_hide_face_title_lable	Hide face title label (Y/N)
fd_hide_face_hint_message	Hide face hint / instruction label (Y/N)
fd_hide_title_image	Hide face title image (Y/N)
fd_show_custom_ui	In case you want to show your own custom overlay over face detection screen, you can pass your custom image and show it. First you need to pass (Y/N) in this parameter, based on your choice to show custom overlay or not. We have predefined image sizes for overlay images, please check it on the URL below. You can create similar size transparent PNG overlay image and pass it. https://drive.google.com/file/d/1qd2DrIIDkNSAeQLjcibVP3Xlrj3-27e5/view?usp=sharing
fd_face_outline_image_id	Drawable resource ID for overlay border image.

`fd_outside_face_outline_image_id`	Drawable resource ID for overlay background image.
`fd_face_outline_progress_images`	In case you want to show animation/gif on face detection screen you need to pass the comma separated list of image resource ID in this parameter.
`fd_face_outline_progress_images_delay`	You can customize gif image delay here in milisecond.
`fd_face_turn_arrow_list`	In case you want to customize face turn arrow you need to pass four commas separated drawable resource IDs. First image will be used as left arrow and second image will be used as top arrow and so on from left to bottom.
`fd_toggle_camera_button_icon`	You can customize toggle camera icon by passing image resource ID here.

`labels`	Currently following labels are shown on Face Capture screen, that can be customized with your own custom message.
	
	`light`
	`focus`
	`smile`
	`too_much_glare_face_capture`
	`out_of_focus_fc_detect`
	`subject_is_too_dark_fc_detect`
	`move_camera_closer_to_your_face`
	`camera_movement_fc_detect`
	`smile_please`
	`face_detected`
	`keep_face_steady`
	`page_title_face_detection`
	`camera_error`
	`turn_left`
	`turn_right`
	`move_up`
	`move_down`
	`look_straight`

Table 10: Custom Camera Fingerprint Capture Configurations

Camera fingerprint capture config	Description
cfc_label_text_typeface_type	Following values are supported for label typeface. DEFAULT, DEFAULT_BOLD, SANS_SARIF, SERIF, MONOSPACE
cfc_label_text_typeface_style	Following values are supported for label typeface style. NORMAL, BOLD, ITALIC, BOLD_ITALIC
cfc_label_text_color cfc_label_text_color_alpha	Color and Transparency of label text on instruction screen.
cfc_instruction_button_color cfc_instruction_button_alpha	Color and Transparency of instruction screen continue button.
cfc_instruction_button_txt_color cfc_instruction_button_txt_alpha	Color and Transparency of instruction screen continue button text.
id_enable_label_shadow	Enable/Disable label shadow
id_capture_button_color id_capture_button_alpha	Color and Transparency of ID capture button.

Table 11: Custom Voice Recording Configurations

Voice Recording	Description color
voice_button_color voice_button_color_alpha	Button color of voice recording screen
voice_background_color voice_background_color_alpha	Background color of voice recording screen
voice_title_label_on_top	Title position
voice_display_position	Voice recording pop up position
voice_auto_play	Record voice automatically when function is called or take user action(button press)
voice_recording_time	Max recording time
voice_text_label_color voice_text_label_color_alpha	Label color on voice recording screen
voice_title_label_color voice_title_label_color_alpha	Title label color on voice recording screen
voice_title_label_size voice_text_label_size	Title label size
voice_counter_label_size	Voice recording counter label size

4 Sequence Diagrams

Now that the basic setup and customization is understood, use the sequence diagrams in this section to setup and utilize the most commonly used services through the SDK. In Sections 5-7, each of the functions within these diagrams are explained in detail.

4.1 ID Validation Only (Service ID 20)

This diagram depicts Service ID 20 where the initiating application needs to utilize the SDK features to capture the front and back of the ID document, then send it the IDmission server for validation. The response from the server will include the validation result as well as data extracted from the ID documents.

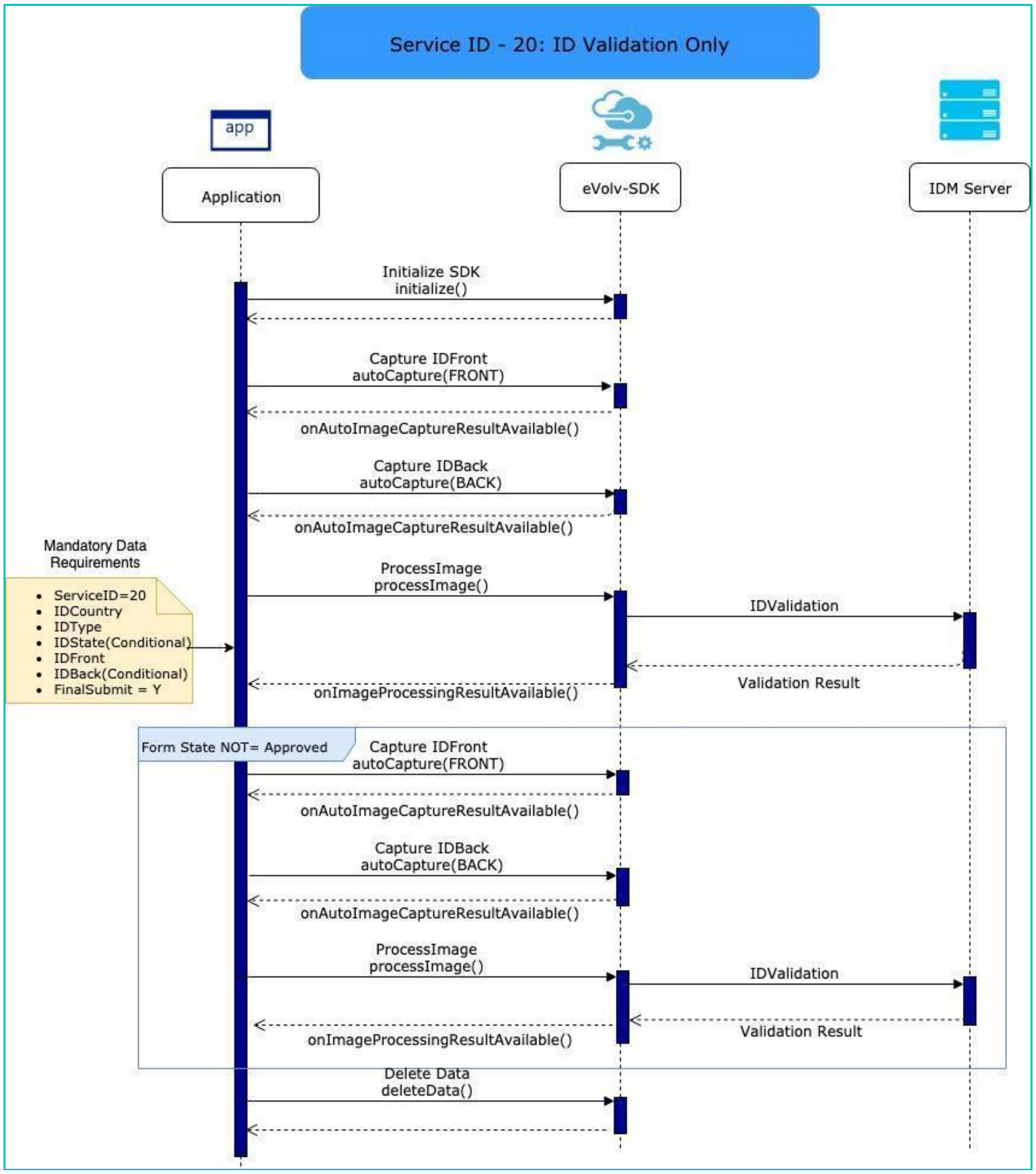


Figure 1: ID Validation Only sequence diagram

4.2 ID Validation + Face Match (Service ID 10)

This diagram depicts Service ID 10 where the initiating application needs to utilize the SDK features to capture the front and back of the ID document, capture a selfie and then send it the IDmission server for validation and biometric matching. The response from the server will include the validation results as well as data extracted from the ID documents.

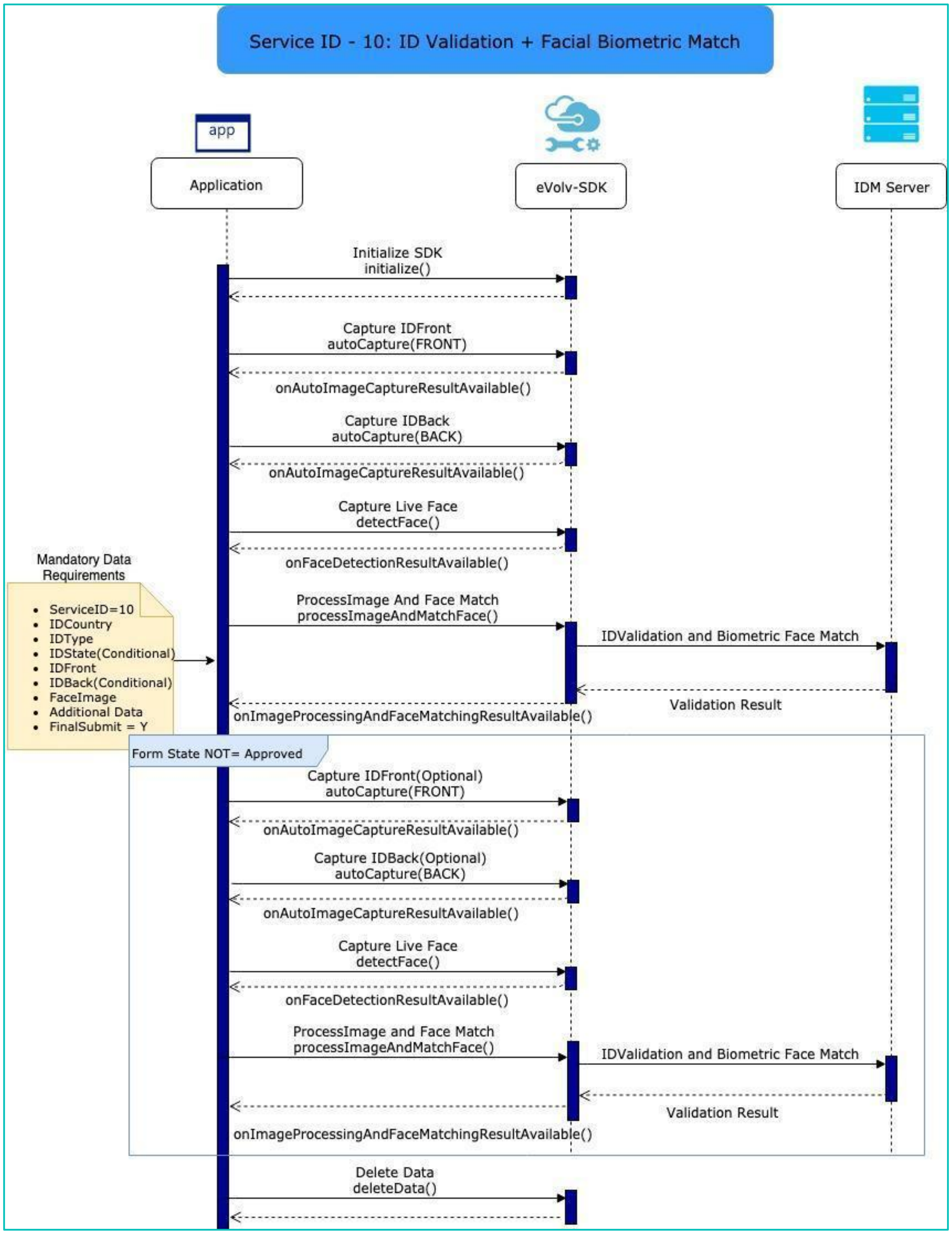
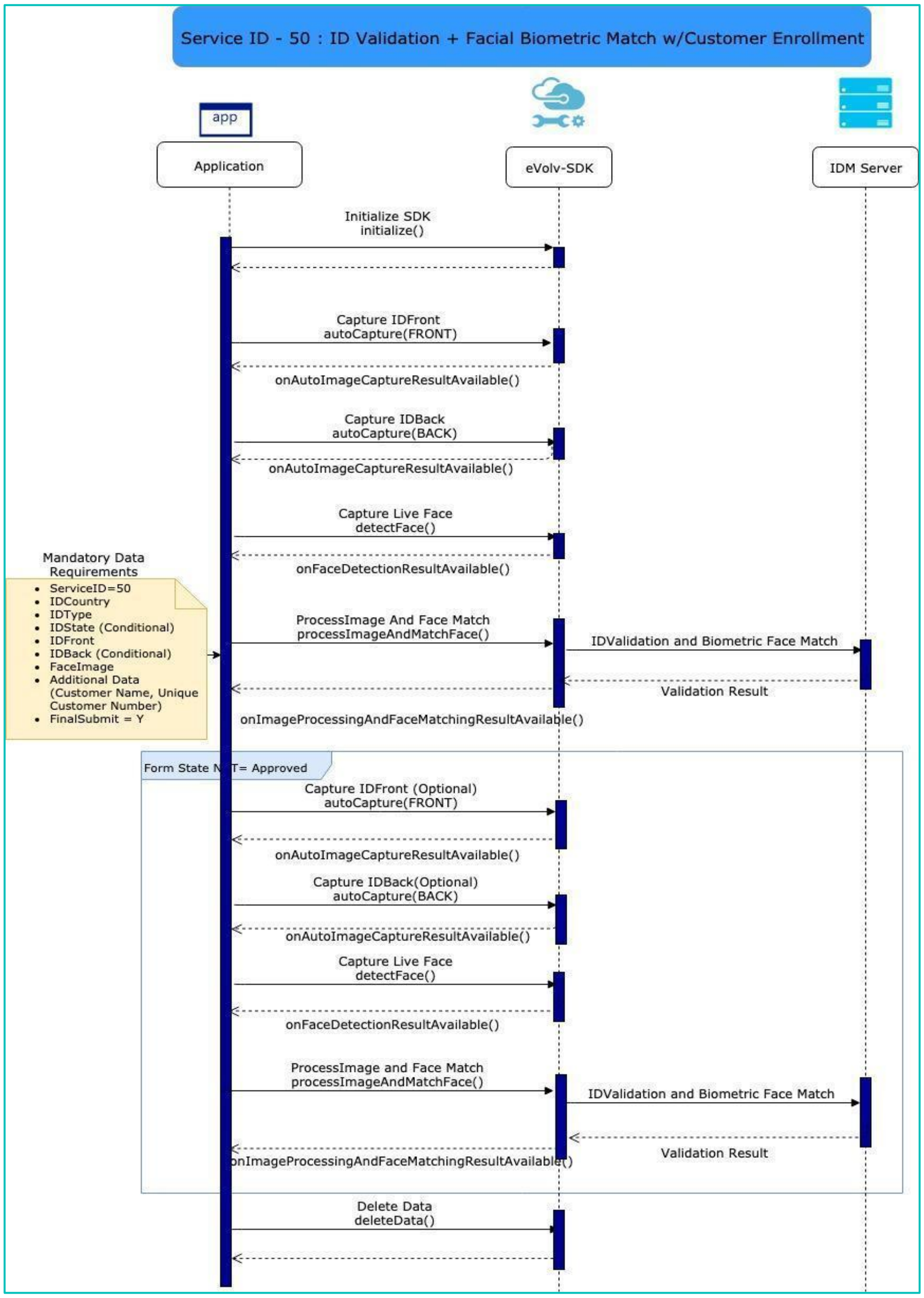

Service ID - 10: ID Validation + Facial Biometric Match
app
Application
eVolv-SDK
IDM Server
Initialize SDK
initialize()
Capture IDFront
autoCapture(FRONT)
onAutoImageCaptureResultAvailable()
Capture IDBack
autoCapture(BACK)
onAutoImageCaptureResultAvailable()
Capture Live Face
detectFace()
onFaceDetectionResultAvailable()
Mandatory Data Requirements
ServiceID=10
IDCountry
IDType
IDState(Conditional)
IDFront
IDBack(Conditional)
FaceImage
Additional Data
FinalSubmit = Y
ProcessImage And Face Match
processImageAndMatchFace()
IDValidation and Biometric Face Match
Validation Result
onImageProcessingAndFaceMatchingResultAvailable()
Form State NOT= Approved
Capture IDFront(Optional)
autoCapture(FRONT)
onAutoImageCaptureResultAvailable()
Capture IDBack(Optional)
autoCapture(BACK)
onAutoImageCaptureResultAvailable()
Capture Live Face
detectFace()
onFaceDetectionResultAvailable()
ProcessImage and Face Match
processImageAndMatchFace()
IDValidation and Biometric Face Match
Validation Result
onImageProcessingAndFaceMatchingResultAvailable()
Delete Data
deleteData()

Figure 2: ID Validation + Face Match sequence diagram

4.3 ID Validation + Face Match w/Customer Enrollment (Service ID 50)

This diagram depicts Service ID 50 where the initiating application needs to utilize the SDK features to capture the front and back of the ID document, capture a selfie and basic customer details. Additionally, other biometrics can be captured to subsequently use in future verification service. All of this information will be sent to the IDmission server for validation and biometric matching. The response from the server will include the validation results as well as data extracted from the ID documents and a unique assigned customer code.



Service ID - 50 : ID Validation + Facial Biometric Match w/Customer Enrollment
app
Application
eVolv-SDK
IDM Server
Initialize SDK
initialize()
Capture IDFront
autoCapture(FRONT)
onAutoImageCaptureResultAvailable()
Capture IDBack
autoCapture(BACK)
onAutoImageCaptureResultAvailable()
Capture Live Face
detectFace()
onFaceDetectionResultAvailable()
Mandatory Data Requirements
ServiceID=50
IDCountry
IDType
IDState (Conditional)
IDFront
IDBack (Conditional)
FaceImage
Additional Data (Customer Name, Unique Customer Number)
FinalSubmit = Y
ProcessImage And Face Match
processImageAndMatchFace()
IDValidation and Biometric Face Match
Validation Result
onImageProcessingAndFaceMatchingResultAvailable()
Form State NOT= Approved
Capture IDFront (Optional)
autoCapture(FRONT)
onAutoImageCaptureResultAvailable()
Capture IDBack(Optional)
autoCapture(BACK)
onAutoImageCaptureResultAvailable()
Capture Live Face
detectFace()
onFaceDetectionResultAvailable()
ProcessImage and Face Match
processImageAndMatchFace()
IDValidation and Biometric Face Match
Validation Result
onImageProcessingAndFaceMatchingResultAvailable()
Delete Data
deleteData()

Figure 3: ID Validation + Face Match w/ Customer Enrollment sequence diagram

4.4 Customer Enrollment with Biometrics (Service ID 175)

This diagram depicts Service ID 175 where the initiating application wants to enroll a customer and their biometrics (e.g. face, fingerprint, etc.) but does not require the collection and validation of ID documents.

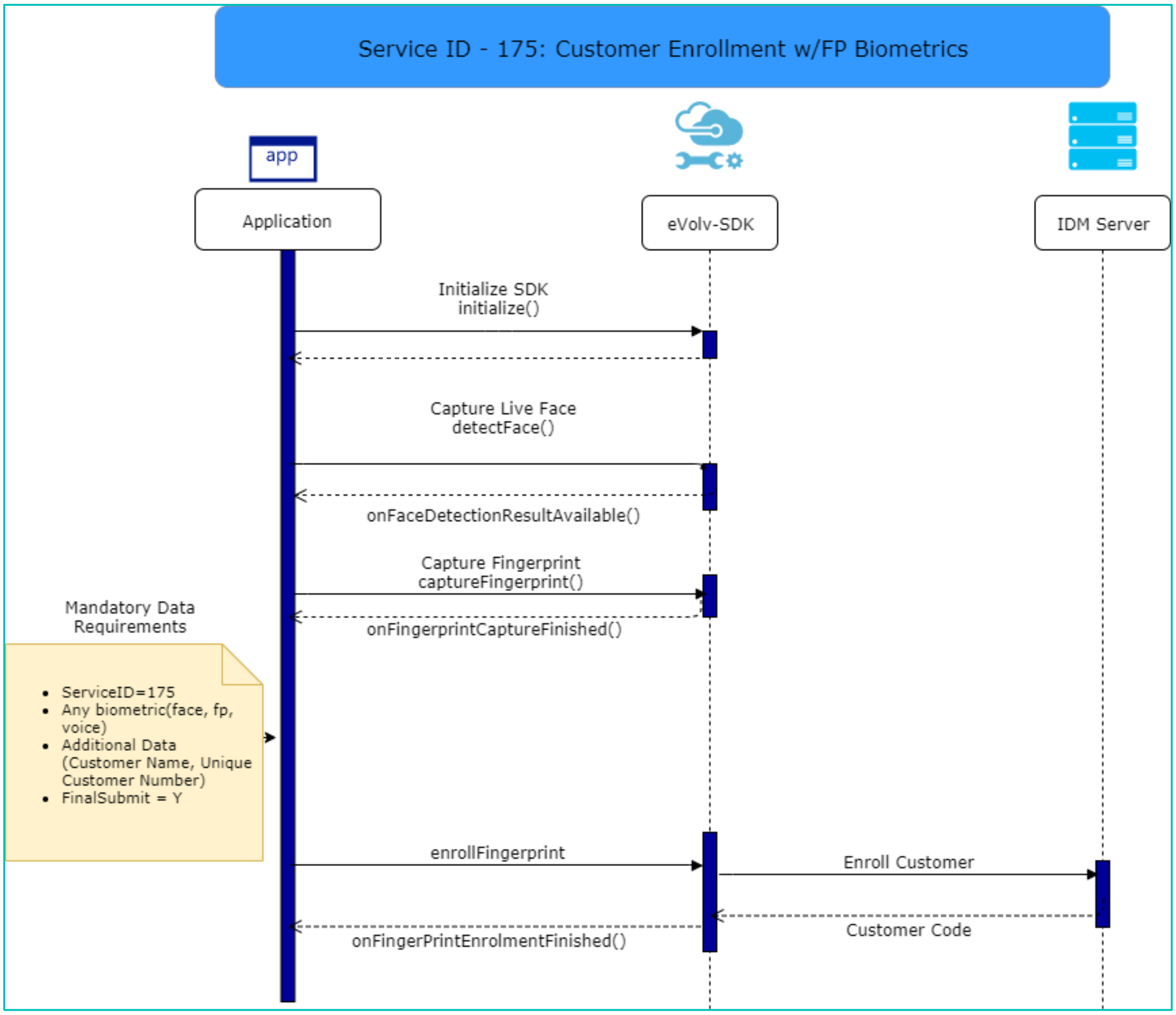

Service ID - 175: Customer Enrollment w/FP Biometrics
app
Application
eVolv-SDK
IDM Server
Initialize SDK
initialize()
Capture Live Face
detectFace()
onFaceDetectionResultAvailable()
Capture Fingerprint
captureFingerprint()
onFingerprintCaptureFinished()
Mandatory Data Requirements
ServiceID=175
Any biometric(face, fp, voice)
Additional Data (Customer Name, Unique Customer Number)
FinalSubmit = Y
enrollFingerprint
Enroll Customer
Customer Code
onFingerPrintEnrolmentFinished()

Figure 4: Customer enrollment with biometrics sequence diagram

4.5 Customer Verification (Service ID 105)

This diagram depicts Service ID 105 where the initiating application will send a unique identifier for the customer along with captured biometrics. This information will be used to verify the customer against previously enrolled biometrics. The response from the server will include the verification result.

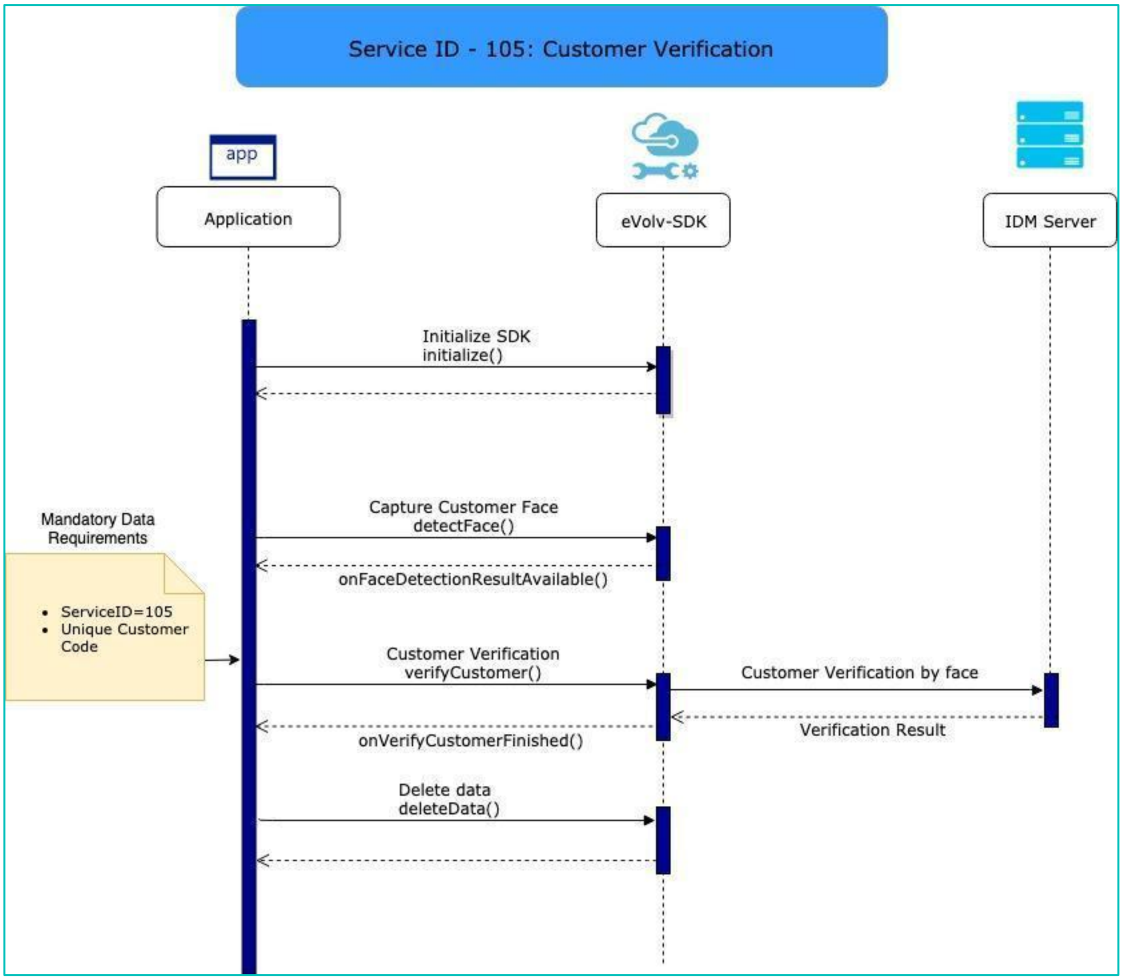


Figure 5: Customer verification sequence diagram

4.6 Customer Update (Service ID 70)

This diagram depicts Service ID 70 where the initiating application wants to update the information for a previously enrolled customer.

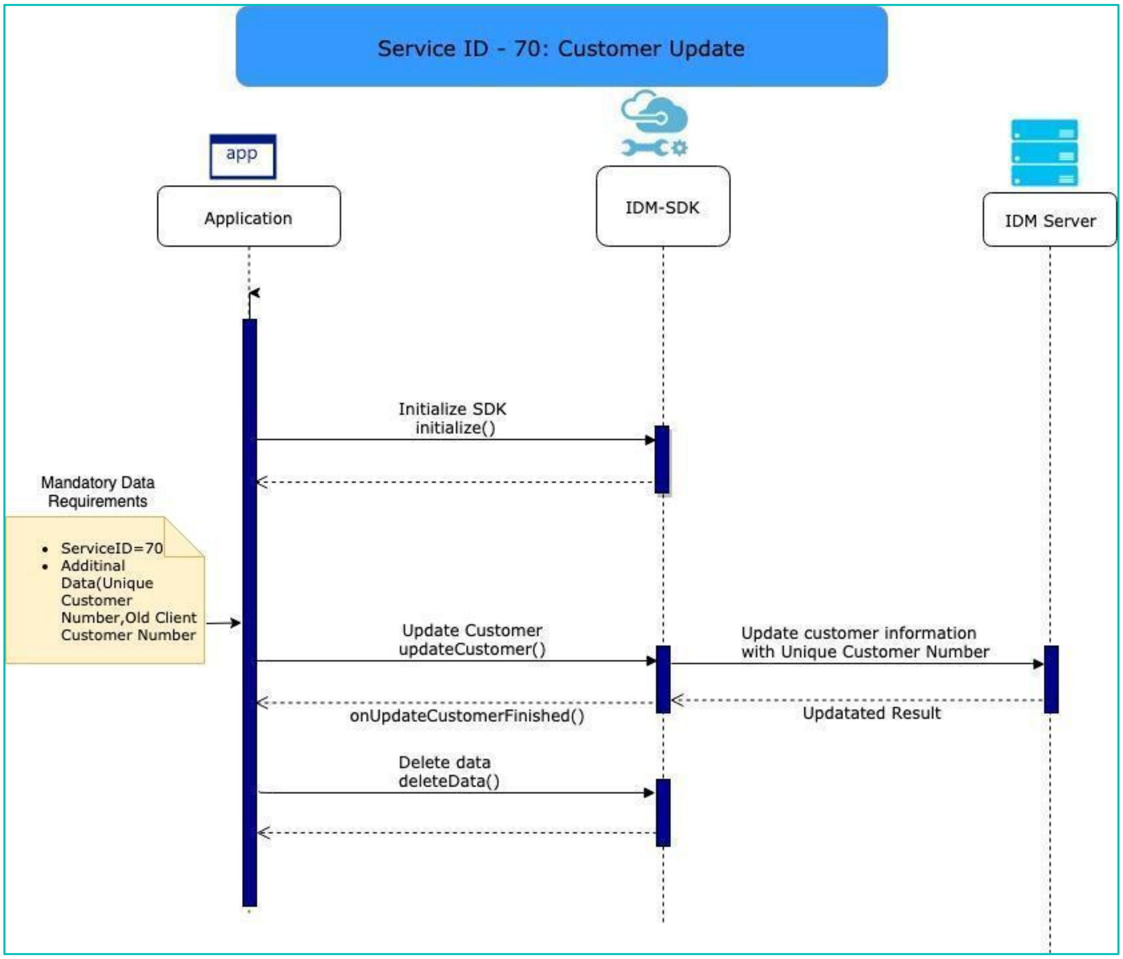

Service ID - 70: Customer Update
app
Application
IDM-SDK
IDM Server
Initialize SDK
initialize()
Mandatory Data Requirements
• ServiceID=70
• Additinal Data(Unique Customer Number,Old Client Customer Number
Update Customer
updateCustomer()
Update customer information with Unique Customer Number
onUpdateCustomerFinished()
Updatated Result
Delete data
deleteData()

Figure 6: Customer update sequence diagram

5 SDK Main Functions

The following sections describe each of the function calls that are used in each service. Many of these functions are mandatory for a specific service, but others can be optionally included as required by the initiating application.

5.1 Automatic Image Capture

To capture an image, use the following calls. First call will launch camera in portrait mode by default. This function is used for the following Service IDs: 10, 20, 50, 55, 155, 160, 165, 505, 510, 515.

```
@Deprecated
ImageProcessingSDK.autoCapture(activityContext, imageType);

@Deprecated
ImageProcessingSDK.autoCapture(activityContext, imageType, capturePortrait);

@Deprecated
ImageProcessingSDK.autoCapture(activityContext,imagesTypes,capturePortrait,l
ightThreshold,minFocusThreshold,maxFocusThreshold,glarePercentage,enableCapt
ureButtonTime,maxImageSize,imageHeight,imageWidth);

@Deprecated
ImageProcessingSDK.autoCapture(Activity activityContext, ImageType
imagesTypes, boolean capturePortrait, int lightThreshold, int
minFocusThreshold, int maxFocusThreshold, int glarePercentage, int
enableCaptureButtonTime, int maxImageSize, int imageHeight, int imageWidth,
JSONObject additionalJSON, ColorCode idOutlineColor, ColorCode
idOutsideOutlineColor)

@Deprecated
ImageProcessingSDK.autoCapture(Activity activityContext, ImageType
imagesTypes, boolean capturePortrait, int lightThreshold, int
minFocusThreshold, int maxFocusThreshold, int glarePercentage, int
enableCaptureButtonTime, int maxImageSize, int imageHeight, int imageWidth,
JSONObject additionalJSON, ColorCode idOutlineColor, ColorCode
idOutsideOutlineColor, ColorCode detectedIdOutsideOutlineColor,  Bitmap
titleImageCaptureBitmap)

@Deprecated
ImageProcessingSDK.autoCapture(Activity activityContext, ImageType
imagesTypes, boolean capturePortrait, int lightThreshold, int
minFocusThreshold, int maxFocusThreshold, int glarePercentage, int
enableCaptureButtonTime, int maxImageSize, int imageHeight, int imageWidth,
JSONObject additionalJSON, ColorCode idOutlineColor, ColorCode
idOutsideOutlineColor, ColorCode detectedIdOutsideOutlineColor,  Bitmap
titleImageCaptureBitmap, boolean showInstruction, int imageresourceid)
@Deprecated


ImageProcessingSDK.autoCapture(Activity activityContext, ImageType
imagesTypes, JSONObject additionalJSON, Bitmap titleImageCaptureBitmap, int
imageresourceid, HashMap<String, String> snippetNameMap, JSONObject
commonConfigJSON)
```

In the case where capture of the front/back of the ID is done multiple times, the old image will be deleted.

Note: Front image should be captured before capturing back image as back image (if present) will get deleted after capturing front image.

Table 12: Image Capture Parameters

Parameter	Type	Default	Range	Description
activityContext	Context			Instance of your activity
imageType	Enum			ImageType.FRONT or ImageType.BACK
capturePortrait	boolean			For launching camera in portrait/landscape mode
maxImageSize	integer	500	256-2048	Maximum image size in kb
maxFocusThreshold	integer	25	10-50	Minimum focus required
minFocusThreshold	integer	15	5-30	Bare minimum (fallback) focus required
imageHeight	integer	830	640-1920	Image Height in pixels
imageWidth	integer	1170	480-1280	Image Width in Pixels
lightThreshold	integer	60	50-100	Minimum light required
enableCaptureButtonTime	integer	15	5-200	No. of seconds after which manual capture is enabled
glarePercentage	integer	1	0-50	Maximum glare in percentage allowed
ColorCode idOutlineColor : Color for face outline idOutsideOutlineColor : Color for area around outline detectedIdOutsideOutlineColor: Color for detected screen	Object		Hex ColorCode, Transparency (1-100)	HexColorCode - Hexcolor string with RRGGBB format without # Transparency: int value (1-100), where 1 is no transparency and 100 is max transparency.
titleImageCaptureBitmap	Bitmap	null		Send a bitmap image to display on screen
showIntruction	boolean	true	true/false	To show instruction screen

`imageresourceid`	Resource file ID	Default instruction image		Custom instruction image

`commonConfigJSON`	JSONObject			/***We can use below config JSON to pass all the configuration parameter, if no value is sent it will use default value, All the value should be sent as string***/
				`{`
				`capturePortrait : "Y", /***supported parameter Y/N ***/`
				`lightThreshold : "70",`
				`minFocusThreshold : "12",`
				`maxFocusThreshold : "35",`
				`glarePercentage : "5",`
				`enableCaptureButtonTime : "60",`
				`maxImageSize : "500",`
				`imageHeight : "1170",`
				`imageWidth : "800",`
				`idOutlineColor : "FFAD36",`
				`idOutlineColorAlpha : "1",`
				`detectedIdOutlineColor : "6EB24C",`
				`detectedIdOutlineColorAlpha : "1",`
				`idOutsideOutlineColor : "487D95",`
				`idOutsideOutlineColorAlpha : "1",`
				`detectedIdOutsideOutlineColor : "487D95",`
				`detectedIdOutsideOutlineColorAlpha : "1",`
				`showInstruction : "Y",`
				/***supported parameter `Y/N` ***/
				`captureEnable : "F",`
				/***supported parameter `Y/N` ***/
				`fieldName : "" type : "DEFAULT"`

				/***supported parameter SANS_SERIF, SERIF, MONOSPACE, DEFAULT_BOLD ***/ style : "NORMAL" /***supported parameter NORMAL, BOLD, ITALIC, BOLD_ITALIC ***/ backButtonColor : "FFAD36" backButtonColorAlpha : "1" retryButtonColor : "FFAD36" retryButtonColorAlpha : "1" confirmButtonColor : "FFAD36" confirmButtonColorAlpha : "1" textLabelColor : "FFFFFF" textLabelAlpha : "1" instructionButtonColor : "FFFFFF" instructionButtonColorAlpha : "1" instructionButtonTXTColor : "487D95" instructionButtonTXTColorAlpha : "1" headerTextLabelColor : "FFFFFF" headerTextLabelAlpha : "1" headerType : "DEFAULT" /***supported parameter SANS_SERIF, SERIF, MONOSPACE, DEFAULT_BOLD ***/ headerStyle: "NORMAL" /***supported parameter NORMAL, BOLD, ITALIC, BOLD_ITALIC ***/ headerTextLabelSize:”” textLabelSize=”” }

Below method is released in SDK version 6.2.6.2 and it is recommended to use this method going forward.

```
ImageProcessingSDK.autoCapture(Activity activityContext, ImageType
imagesTypes, JSONObject idCaptureConfig, JSONObject additionalJSON,
HashMap<String, String> snippetNameMap)
```

Table 13: Image Capture Parameters (6.2.6.2 and above)

Parameter	Type	Default	Range	Description
`activityContext`	Context			Instance of your activity
`imageType`	Enum			`ImageType.FRONT` or `ImageType.BACK`

`idCaptureConfig`	JSONObject			/***We can use below config JSON to pass all the feature related configuration parameter, if no value is sent it will use default value, All the value should be sent as string***/
				Note:
				For customizing UI use "`customizeUserInterface`" method with JSON string.
				`{`
				`id_capture_portrait : "Y",`
				/***supported parameter Y/N ***/
				`id_light_threshold : "70",`
				`id_min_focus : "12",`
				`id_max_focus : "35",`
				`id_glare_percentage : "5",`
				`id_enable_capture_button_time : "60",`
				`id_max_image_size : "500",`
				`id_image_height : "1170",`
				`id_image_width : "800",`
				`id_capture_enable : "N",`
				/***supported parameter `Y/N` ***/
				`id_generic_doc_fieldname : "",`
				/***Pass the below parameter to enable barcode scanning from given ID type***/
				`id_country : "", //3 character ISO country code`
				`id_state : "", //2 character ISO state code`
				`id_type : ""  //Refer to ID Type table`
				`}`

5.2 Processing Images with Parameters and Additional Data

This function is used to process the image(s) and retrieve data. Also, additional data is sent in JSON format. You can also include parameter settings to request a manual review of images by the IDmission back office review team, or to bypass certain features that are built into the solution. This function is used for the following Service ID: 20.

```
ImageProcessingSDK.processImage(activityContext, countryCode, stateCode,
idType,additionalJSON);
ImageProcessingSDK.processImage(activityContext, countryCode, stateCode,
idType,additionalJSON, isFinalSubmit);
ImageProcessingSDK.processImage(activityContext, countryCode, stateCode,
idType,additionalJSON, isFinalSubmit, clearFormKey);
ImageProcessingSDK.processImage(Activity activityContext, String
countryCode, String stateCode, IdType idType, JSONObject additionalJSON,
JSONObject empJSON, boolen isFinalSubmit, boolen clearFormKey)
```

The captured image data will be sent to the server for processing.

Table 14: Processing Images Parameters

Parameter	Type	Description
activityContext	Context	Instance of your activity
idType	Enum	Refer to ID Type table
countryCode	String	3-character ISO country code
stateCode	String	2-character ISO state code
additionalJSON	JSONObject	JSON Object may contain following elements, those with a double // are mandatory. Please refer to the Additional JSON Element Dictionary section of this document for more details on each of the subsequent elements: { "Service_ID": "",//Mandatory "Manual_Review_Required" : "" "Bypass_Age_Validation" : "" "Bypass_Name_Matching" : "" "Deduplication_Required" : "" "Need_Immediate_Response" : "" "Capture_Secondary_ID" : "" "Unique_Customer_Number": "", "Old_Client_Customer_Number": "", "Customer_Name": "", "Customer_Type": "", "Customer_Phone": "", "Customer_Email": "", "Unique_Merchant_Number": "", "Unique_Employee_Number": "", "Unique_Employee_Code": "", "Gender": "", "Address_Line_1": "", "Address_Line_2": "", "Country": "", "State": "",

		"City": "", "Postal_Code": "", "Customer_Attribute": "", "AddressLine1": "", "AddressLine2": "", "Customer_Gender": "", "AgentCountry": "", "Encrypted_Data": "", "PreviousFormId": "", } NOTE: When performing a service that includes customer enrollment, the `Unique_Customer_Number` is mandatory.

empJSON		JSON Object may contain following elements, those with a double `//` are mandatory. Please refer to the EmpJSON Element Dictionary section of this document for more details on each of the subsequent elements:
		
		{
		"Employee_Code":"", /Conditional
		"Employee_Type":"", /Conditional
		"Login_ID": "",
		"Employee_Email": "",/Conditional
		"Employee_Name":"",/Conditional
		"Employee_Mobile_Number":"",
		"Employee_ID_Number":"",
		" Employee_Gender ":"",
		"Employee_Email ":"",
		" Employee_Gender ":"",
		" Employee_AddressLine1 ":"",
		" Employee_AddressLine2 ":"",
		" Employee_City":"",
		"Employee_State":"",
		"Employee_Country":"",
		}
		
		NOTE: When performing a service that includes employee enrollment the `Employee_Code`, `Employee_Type`, `Employee_Email`, and `Employee_Name` elements are all mandatory.

`isFinalSubmit`	boolean	Set `TRUE` to execute submit request.
`clearFormKey`	boolean	Clear form key once operation is executed successfully

Table 15: ID Types

ID Type Key	ID Type Value	ID Back Required
IdType.PASSPORT	PP	N
IdType.NATIONAL_ID	NID	Y
IdType.RESIDENCE_CARD	RID	Y
IdType.DRIVERS_LICENSE	DL	Y
IdType.PHOTO_ID	PID	Y
IdType.VOTER_ID_CARD	VID	Y
IdType.TAX_ID_CARD	TID	Y
IdType.WORK_VISA_PERMIT	WV	Y
IdType.STUDENT_VISA_PERMIT	SV	Y
IdType.MILITARY_POLICE_GOVERNMENT_ID	GID	Y
IdType.BOAT_SHIP_ID_CARD	BID	Y
IdType.OTHERS	OTH	Y
IdType.GLOBAL_ENTRY_CARD	GE	Y
IdType.PASSPORT_CARD	PPC	Y

By default, all of the ID Types from various countries that IDmission accepts are made available to you through this SDK. If you want to limit the ID Types you accept, you can do so by limiting the selections within your initiating application. The list of all supported ID types is available on the IDmission website. If you find that there is an ID Type that is not supported, please contact our support team to request one to be added.

5.3 Live Face Detection

This set of functions is used to detect live face and capture an image of the face. These functions are used for the following Service IDs: 10, 50, 55, 105, 185, 190, 305.

```
@Deprecated
ImageProcessingSDK.setImageProcessingResponseListener(activityContext);

ImageProcessingSDK.setSDKResponseListener(activityContext);

@Deprecated
imageProcessingSDK.detectFace(activityContext, lightThreshold,
focusThreshold, detectionThreshold, launchFrontCamera);

@Deprecated
imageProcessingSDK.detectFace(MainActivity.this, lightThresholdVal,
focusThresholdVal, detectionThreshold,maxImageSize, launchFrontCamera);

@Deprecated
imageProcessingSDK.detectFace(MainActivity.this, lightThresholdVal,
focusThresholdVal, detectionThreshold,maxImageSize, launchFrontCamera,
showPreview);
```

```
@Deprecated
ImageProcessingSDK.detectFace(Activity activityContext, int lightThreshold,
int focusThreshold, int faceDetectionThreshold, int maxImageSize, boolean
launchFrontCamera, boolean showPreview, ColorCode faceOutlineHexColor,
ColorCode detectedFaceOutlineHexColor, ColorCode outsideFaceOutlineColor);

@Deprecated
ImageProcessingSDK.detectFace(Activity activityContext, int lightThreshold,
int focusThreshold, int faceDetectionThreshold, int maxImageSize, boolean
launchFrontCamera, boolean showPreview, ColorCode faceOutlineHexColor,
ColorCode detectedFaceOutlineHexColor, ColorCode outsideFaceOutlineColor,
Bitmap titleBitmap);

@Deprecated
ImageProcessingSDK.detectFace(Activity activityContext, int lightThreshold,
int focusThreshold, int faceDetectionThreshold, int maxImageSize, boolean
launchFrontCamera, boolean showPreview, ColorCode faceOutlineHexColor,
ColorCode detectedFaceOutlineHexColor, ColorCode outsideFaceOutlineColor,
Bitmap titleBitmap, boolean showInstruction, int imageresourceid);

@Deprecated
ImageProcessingSDK.detectFace(Activity activityContext, int lightThreshold,
int focusThreshold, int faceDetectionThreshold, int maxImageSize, boolean
launchFrontCamera, boolean showPreview, ColorCode faceOutlineHexColor,
ColorCode detectedFaceOutlineHexColor, ColorCode outsideFaceOutlineColor,
Bitmap titleBitmap, boolean showInstruction, int imageresourceid, boolean
toogleCamera);

@Deprecated
ImageProcessingSDK.detectFace(Activity activityContext,Bitmap titleImage,
int imageresourceid, JSONObject commonConfigJson);
```

Table 16: Live Face Parameters

Parameter	Type	Default	Range	Description
`lightThreshold`	Integer	60	50-100	Light Threshold
`focusThreshold`	Integer	15	10-50	Focus Threshold
`detectionThreshold`	Integer	20	10-50	Detection Threshold
`maxImageSize`	Integer	500	256-2048	Maximum image size in kb
`launchFrontCamera`	boolean	true	true-false	For launching front camera by default
`showPreview`	boolean	true	true-false	Enable/Disable preview screen after face-detection

ColorCode faceOutlineHexColor : Face outline color detectedFaceOutlineHexColor : Detected face color outsideFaceOutlineColor : Color outside face outline	Object		Hex ColorCode, Transparency (1-100)	HexColorCode - Hexcolor string with RRGGBB format without # Transparency: int value (1-100), where 1 is no transparency and 100 is max transparency.
titleBitmap	Bitmap	null		Send a bitmap image to display on screen
showIntruction	boolean	true	true/false	To show instruction screen
imageresourceid	Resource file ID	Default instruction image		Custom instruction image
toggleCamera	boolean	false	true/false	Enable toggle camera button
commonConfigJSON	JSONObject			/***We can use below config JSON to pass all the configuration parameter, if no value is sent it will use default value, All the value should be sent as string***/ { lightThreshold : "60", focusThreshold : "15", faceDetectionThreshold : "20", maxImageSize : "500", launchFrontCamera : "Y", showPreview : "Y", faceOutlineHexColor : "FFAD36", faceOutlineHexColorAlpha : "1", detectedFaceOutlineHexColor : "6EB24C", detectedFaceOutlineHexColorAlpha : "1", outsideFaceOutlineColor : "487D95",

				outsideFaceOutlineColorAlpha : "1", detectedFaceOutsideColor : "487D95", detectedFaceOutsideColorAlpha : "1", showInstruction : "Y", toggleCamera : "N", type : "DEFAULT" /***supported parameter SANS_SERIF, SERIF, MONOSPACE, DEFAULT_BOLD ***/ style : "NORMAL" /***supported parameter NORMAL, BOLD, ITALIC, BOLD_ITALIC ***/ backButtonColor : "FFAD36" backButtonColorAlpha : "1" retryButtonColor : "FFAD36" retryButtonColorAlpha : "1" confirmButtonColor : "FFAD36" confirmButtonColorAlpha : "1" textLabelColor : "FFFFFF" textLabelAlpha : "1" instructionButtonColor : "FFFFFF" instructionButtonColorAlpha : "1" instructionButtonTXTColor : "487D95" instructionButtonTXTColorAlpha : "1" faceContours : "Low", /***supported parameter All, Low, Medium, Zero ***/ headerTextLabelColor : "FFFFFF" headerTextLabelAlpha : "1" headerType : "DEFAULT"

				/***supported parameter SANS_SERIF, SERIF, MONOSPACE, DEFAULT_BOLD ***/ headerStyle: "NORMAL" /***supported parameter NORMAL, BOLD, ITALIC, BOLD_ITALIC ***/ headerTextLabelSize:” ” textLabelSize=”” faceTitleOnTop =”N” faceHintMessageOnTop= ”Y” "fd_show_custom_ui":"N", "fd_face_outline_image_id":"", "fd_outside_face_outline_image_id":"", "fd_face_outline_progress_images":"****Comma separated images resource id****", "fd_face_outline_progress_images_delay":"500", "fd_face_turn_arrow_list":"****comma separated 4 images resource ids(Left arrow to Bottom arrow)****", "fd_toggle_camera_button_icon":"", } Custom UI images should be transparent **.PNG** images with particular dimension. Please check below ZIP for sample images. https://drive.google.com/file/d/1qd2DrIIDkNSAeQLjcibVP3Xlrj3-27e5/view?usp=sharing

Below method is released in SDK version 6.2.6.2 and it is recommended to use this method going forward.

```
ImageProcessingSDK.detectFace(Activity activityContext, JSONObject
commonConfigJson);
```

Table 17: Live Face Parameters

Parameter	Type	Description

`commonConfigJSON`	JSONObject	/***We can use below config JSON to pass all the feature related configuration parameter, if no value is sent it will use default value, All the value should be sent as string***/
		
		Note: For customizing UI use “`customizeUserInterface`” method with JSON string.
		
		`{`
		`fd_light_threshold : "60",`
		`fd_focus_threshold : "15",`
		`fd_detection_threshold : "20",`
		`fd_max_image_size : "500",`
		`fd_launch_front_camera : "Y",`
		`fd_toggle_camera : "N",`
		`fd_enable_passive_face_detection : "Y"`
		`}`

5.4 Process Image and Facial Biometric Verification with Additional Data

This is an overloaded API to process the image and match the face image. Also, additional data will be sent in JSON format. These functions are used for the following Service IDs: 10, 50, 55, 155, 160, 165, 505, 510, 515.

```
ImageProcessingSDK.processImageAndMatchFace(activityContext, countryCode,
stateCode, idType, faceImageType, additionalJSON);
ImageProcessingSDK.processImageAndMatchFace(activityContext, countryCode,
stateCode, idType, faceImageType, additionalJSON, isFinalSubmit);
ImageProcessingSDK.processImageAndMatchFace(activityContext, countryCode,
stateCode, idType, faceImageType, additionalJSON, isFinalSubmit,
clearFormKey);
ImageProcessingSDK.processImageAndMatchFace(Activity activityContext, String
countryCode, String stateCode, IdType idType, String faceImageType,
JSONObject additionalJSON, JSONObject empJSON, Boolean isFinalSubmit,
Boolean clearFormKey);
```

The captured image data will be sent to the server for processing.

Table 18: Process Image and Facial Biometric Verification Parameters

Parameter	Type	Description
activityContext	Context	Instance of your activity
idType	Enum	Refer to ID Type table mentioned above
countryCode	String	3-character ISO country code
stateCode	String	Refer to States table mentioned above
faceImageType	String	Keyname: FACE PROCESSED_FACE OVAL_FACE

additionalJSON	JSONObject	JSON Object may contain following elements, those with a double // are mandatory. Please refer to the Additional JSON Element Dictionary at the end of this document for details regarding each of these elements:
		{
		"Service_ID": "",//Mandatory
		"Manual_Review_Required" : ""
		"Bypass_Age_Validation" : ""
		"Bypass_Name_Matching" : ""
		"Deduplication_Required" : ""
		"Need_Immediate_Response" :
		""
		"Capture_Secondary_ID" : ""
		"Unique_Customer_Number": "",
		/Conditional
		"Old_Client_Customer_Number":
		"", /Conditional
		"Customer_Name": "",
		"Customer_Type": "",
		"Customer_Phone": "",
		"Customer_Email": "",
		"Unique_Merchant_Number":
		"",
		"Unique_Employee_Number": "",
		"Unique_Employee_Code": "",
		"Gender": "",
		"Address_Line_1": "",
		"Address_Line_2": "",
		"Country": "",
		"State": "",
		"City": "",
		"Postal_Code": "",
		"Customer_Attribute": "",
		"AddressLine1": "",
		"AddressLine2": "",
		"Customer_Gender": "",
		"AgentCountry": "",
		"Encrypted_Data": "",
		"PreviousFormId": "",
		}

		NOTE: When performing a service that includes customer enrollment, the `Unique_Customer_Number` is mandatory.

`empJSON`		JSON Object may contain following elements, those with a double // are mandatory. Please refer to the empJSON Element Dictionary at the end of this document for details regarding each of these elements
		`{`
		`"Employee_Code":"",`
		`"Employee_Type":"",`
		`"Login_ID": "",`
		`"Employee_Email": "",`
		`"Employee_Name":"",`
		`"Employee_Mobile_Number":"",`
		`"Employee_ID_Number":"",`
		`"Employee_Gender ":"",`
		`"Employee_Email ":"",`
		`"Employee_Gender ":"",`
		`"Employee_AddressLine1 ":"",`
		`"Employee_AddressLine2 ":"",`
		`"Employee_City":"",`
		`"Employee_State":"",`
		`"Employee_Country":"",`
		`}`
		NOTE: When performing a service that includes employee enrollment the `Employee_Code`, `Employee_Type`, `Employee_Email`, and `Employee_Name` elements are all mandatory.

`isFinalSubmit`	boolean	Set TRUE to execute submit request.
`clearFormKey`	boolean	Clear form key once operation is executed successfully

Auto Image Capture and Live Face Capture must be performed prior to this call. The captured image data will be sent to server for processing.

5.5 Customer Enrollment with Biometrics and Additional Data

To enroll a customer with their biometrics. Also, additional data will be sent in JSON format. This function is used for Service ID: 175.

```
enrollFingerprint(Activity activityContext, JSONObject additionalJSON,
boolean clearFormKey)
```

Table 19: Process Image and Facial Biometric Verification Parameters

Parameter	Type	Description

`additionalJSON`	JSONObject	JSON Object may contain following elements, those with a double `//` are mandatory. Please refer to the Additional JSON Element Dictionary section of this document for more details on each of the subsequent elements:
		`{`
		`"Service_ID": "",//`
		`"Unique_Customer_Number": "", //`
		`"Customer_Name": "",`
		`"Customer_Type": "",`
		`"Customer_Phone": "",`
		`"Customer_Email": "",`
		`"Gender": "",`
		`"Address_Line_1": "",`
		`"Address_Line_2": "",`
		`"Country": "",`
		`"State": "",`
		`"City": "",`
		`"Postal_Code": "",`
		`}`

`isFinalSubmit`	boolean	Set `TRUE` to execute submit request.
`clearFormKey`	boolean	Clear form key once operation is executed successfully

5.6 Employee Enrollment with Biometrics and Additional Data

To enroll an employee with their biometrics (Voice, face, fp). Also, additional data will be sent in JSON format. This function is used for Service ID: 180.

```
createEmployee(Activity activityContext, JSONObject employeeJSON, JSONObject
additionalJSON, boolean isFinalSubmit, boolean clearFormKey)
```

Table 20: Employee Enrollment with Biometrics Parameters

Parameter	Type	Description

additionalJSON	JSONObject	JSON Object may contain following elements, those with a double Asterix** are mandatory
		{
		"Service_ID": "",**//
		"Unique_Employee_Code": "",
		}

empJSON	JSONObject	JSON Object may contain following elements, those with a double // are mandatory. Please refer to the EmpJSON Element Dictionary section of this document for more details on each of the subsequent elements:
		{
		"Employee_Code":"", //
		"Employee_Type":"", //
		"Login_ID": "",
		"Employee_Email": "", //
		"Employee_Name":"", //
		"Employee_Mobile_Number":"",
		"Employee_ID_Number":"",
		" Employee_Gender ":"",
		"Employee_Email ":"",
		" Employee_Gender ":"",
		" Employee_AddressLine1 ":"",
		" Employee_AddressLine2 ":"",
		" Employee_City":"",
		"Employee_State":"",
		"Employee_Country":"",
		}

isFinalSubmit	boolean	Set TRUE to execute submit request.
clearFormKey	boolean	Clear form key once operation is executed successfully

5.7 Customer Update

Used to update a previously enrolled customer. This function is used for Service ID: 70.

```
updateCustomer(getActivity(), faceImgType, additionalData)
```

Table 21: Customer Update Parameters

Parameter	Type	Description
faceImageType	String	Keyname:w FACE PROCESSED_FACE OVAL_FACE

additionalJSON	JSONObject	JSON Object may contain following elements, those with a double // are mandatory. Please refer to the Additional JSON Element Dictionary section of this document for more details on each of the subsequent elements:
		{
		"Service_ID": "",//Mandatory
		"Unique_Customer_Number": "",
		/Conditional
		"Old_Client_Customer_Number": "",
		//
		"Customer_Name": "",
		"Customer_Type": "",
		"Customer_Phone": "",
		"Customer_Email": "",
		"Gender": "",
		"Address_Line_1": "",
		"Address_Line_2": "",
		"Country": "",
		"State": "",
		"City": "",
		"Postal_Code": "",
		}

isFinalSubmit	boolean	Set TRUE to execute submit request.
clearFormKey	boolean	Clear form key once operation is executed successfully

5.8 Employee Update

Used to update a previously enrolled employee. This function is used for Service ID: 75

```
updateEmployee(Activity activityContext, String faceImageType, JSONObject
employeeJSON, JSONObject additionalJSON, boolean isFinalSubmit, boolean
clearFormKey)
```

Table 22: Employee Update Parameters

Parameter	Type	Description

`faceImageType`	String	`Keyname:`
		`FACE`
		`PROCESSED_FACE`
		`OVAL_FACE`

`additionalJSON`	JSONObject	JSON Object may contain following elements, those with a double // are mandatory. Please refer to the Additional JSON Element Dictionary section of this document for more details on each of the subsequent elements:
		
		`{`
		`"Service_ID": "",**//`
		
		`}`

`empJSON`		JSON Object may contain following elements, those with a double // are mandatory. Please refer to the EmpJSON Element Dictionary section of this document for details regarding each of these elements
		
		`{`
		`"Employee_Code":"", //Mandatory`
		`"Login_ID": "",`
		`"Employee_Email": "",`
		`"Employee_Organization":"",`
		`"Employee_Name":"",`
		`"Employee_MobileNumber":"",`
		`"Employee_ID_Number":"",`
		`"Employee_Country":"",`
		`"Employee_DateOfBirth ":"",`
		`"Employee_Email ":"",`
		`"Employee_Gender ":"",`
		`"Employee_AddressLine1 ":"",`
		`"Employee_AddressLine2 ":"",`
		`"Employee_State":"",`
		`"Employee_Nationality":"",`
		`}`

`isFinalSubmit`	boolean	Set `TRUE` to execute submit request.
`clearFormKey`	boolean	Clear form key once operation is executed successfully

5.9 Customer Verification

Used to verify a customer using biometrics. This function is used for Service ID: 105. (For verifying customer face image or FP, there must be a capture first).

```
ImageProcessingSDK.verifyCustomer(activityContext, biometricType);

ImageProcessingSDK.verifyCustomer(activityContext, biometricType,
additionalJSON);
```

Table 23: Customer Verification Parameters

Parameter	Type	Range	Description
activityContext	Context		Context
biometricType	String	FACE FP VOICE 4F	Include the ranges that are required to be verified.

additionalJSON	JSONObject		JSON Object may contain following elements, those with a double // are mandatory. Please refer to the Additional JSON Element Dictionary section of this document for more details on each of the subsequent elements:
			{
			"Service_ID": "",//Mandatory
			"Unique_Customer_Number": "", //
			"Old_Client_Customer_Number": "",
			"Customer_Name": "",
			"Customer_Type": "",
			"Customer_Phone": "",
			"Customer_Email": "",
			"Gender": "",
			"Address_Line_1": "",
			"Address_Line_2": "",
			"Country": "",
			"State": "",
			"City": "",
			"Postal_Code": "",
			}

isFinalSubmit	Boolean		Set TRUE to execute submit request
clearFormKey	Boolean		Clear form key once operation is executed successfully

5.10 Employee Verification

Used to verify an employee using biometrics. This function is used for Service ID: 305.

```
imageProcessingSDK.verifyEmployee(Activity activityContext, String empCode,
JSONObject additionalJSON,boolean isFinalSubmit);
```

Table 24: Employee Verification Parameters

Parameter	Type	Default	Range	Description
`activityContext`	Context			Instance of your activity
`employeeCode`	String	null		Unique code of employee

`additionalJSON`	JSONObject			JSON Object must contain the following elements
				`{`
				`"Service_ID": "",//`
				`"Unique_Employee_Code":`
				`"",//`
				`}`

`biometricType`	String		`FACE`
			`FP`
			`VOICE`
			`4F`
				Include the ranges that are required to be verified.

`isFinalSubmit`	boolean			Set `TRUE` to execute submit request
`clearFormKey`	Boolean			Clear form key once operation is executed successfully

5.11 Customer Search

Used to search a customer with or without biometrics. This function is used for service ID 185 (biometric) and 186 (non-biometric).

```
imageProcessingSDK.genericApiCall(Activity activityContext, JSONObject
commonConfigJson);
```

Table 25: Customer Search Parameters

Parameter	Type	Description
`activityContext`	Context	Instance of your activity

`commonConfigJson`	JSONObject	`{`
		`“country_id”: “”,`
		`“state_id”: “”,`
		`“id_type”: “”,`
		`“country_id_secondary”: “”,`
		`“state_id_secondary”: “”,`
		`“id_type_secondary”: “”,`
		`“additionalDataJSON”: {`
		`"Service_ID": "",//Mandatory`
		`"Unique_Customer_Number": "", //`

```
              "Old_Client_Customer_Number": "",
              "Customer_Name": "",
              "Customer_Type": "",
              "Customer_Phone": "",
               "Customer_Email": "",
       "Gender": "",
                "Address_Line_1": "",
                "Address_Line_2": "",
                "Country": "",
                "State": "",
                "City": "",
                "Postal_Code": "",
},
"clear_form_key": "",
"emailId": "",
"mobileNo": "",
"notificationType": "",
"receivedOTP": "",
"addressJson": "",
"empCode": "",
"employeeDataJSON": {
            "Employee_Code":"", //Mandatory
            "Login_ID": "",
            "Employee_Email": "",
            "Employee_Organization":"",
            "Employee_Name":"",
            "Employee_MobileNumber":"",
            "Employee_ID_Number":"",
            "Employee_Country":"",
            "Employee_DateOfBirth ":"",
            "Employee_Email ":"",
            "Employee_Gender ":"",
            "Employee_AddressLine1 ":"",
            "Employee_AddressLine2 ":"",
            "Employee_State":"",
            "Employee_Nationality":"",
}

}
```

NOTE:
Please refer to the Additional JSON Element Dictionary section of this document for additional information on the preceding parameters.

Employee JSON is required when service ID supports employee related operations. If you are not supporting employee related operations, you may skip the Employee JSON node.

5.12 Employee Search

Used to search an employee with or without biometrics. This function is used for service ID 190 (biometric) and 191 (non-biometric).

Table 26: Employee Search Parameters

Parameter	Type	Description
activityContext	Context	Instance of your activity

commonConfigJson	JSONObject	{
		“country_id”: “”,
		“state_id”: “”,
		“id_type”: “”,
		“additionalDataJSON”: {
		"Service_ID": "",//Mandatory
		"Unique_Customer_Number": "", //
		"Old_Client_Customer_Number": "",
		"Customer_Name": "",
		"Customer_Type": "",
		"Customer_Phone": "",
		"Customer_Email": "",
		"Gender": "",
		"Address_Line_1": "",
		"Address_Line_2": "",
		"Country": "",
		"State": "",
		"City": "",
		"Postal_Code": "",
		},
		“clear_form_key”: “”,
		“emailId”: “”,
		“mobileNo”: “”,
		“notificationType”: “”,
		“receivedOTP”: “”,
		“addressJson”: “”,
		“empCode”: “”,
		“employeeDataJSON”: {
		"Employee_Code":"", //Mandatory
		"Login_ID": "",
		"Employee_Email": "",
		"Employee_Organization":"",
		"Employee_Name":"",
		"Employee_MobileNumber":"",
		"Employee_ID_Number":"",
		"Employee_Country":"",
		"Employee_DateOfBirth ":"",
		"Employee_Email ":"",
		"Employee_Gender ":"",
		"Employee_AddressLine1 ":"",
		"Employee_AddressLine2 ":"",
		"Employee_State":"",
		"Employee_Nationality":"",
		}
		}

		NOTE: Please refer to the Additional JSON Element Dictionary section of this document for additional information on the preceding parameters. Employee JSON is required when service ID supports employee related operations. If you are not supporting employee related operations, you may skip the Employee JSON node.

5.13 Generic API Call

Below call is the generic implementation of all the above calls. Instead of calling separate function for different service ID we can use the below call and pass the required data with proper service ID.

```
imageProcessingSDK.genericApiCall(Activity activityContext, JSONObject
commonConfigJson);
```

Table 27: Generic API Parameters

Parameter	Type	Description
`activityContext`	Context	Instance of your activity

`commonConfigJson`	JSONObject	`{`
		`“country_id”: “”,`
		`“state_id”: “”,`
		`“id_type”: “”,`
		`“additionalDataJSON”: {`
		`"Service_ID": "",**//Mandatory`
		`},`
		`“clear_form_key”: “”,`
		`“emailId”: “”,`
		`“mobileNo”: “”,`
		`“notificationType”: “”,`
		`“receivedOTP”: “”,`
		`“addressJson”: “”,`
		`“empCode”: “”,`
		`“employeeDataJSON”: {`
		`"Service_ID": "",//Mandatory`
		`"Unique_Customer_Number": "", //`
		`"Old_Client_Customer_Number": "",`
		`"Customer_Name": "",`
		`"Customer_Type": "",`
		`"Customer_Phone": "",`
		`"Customer_Email": "",`
		`"Gender": "",`
		`"Address_Line_1": "",`
		`"Address_Line_2": "",`
		`"Country": "",`
		`"State": "",`
		`"City": "",`
		`"Postal_Code": "",`
		`}`

		} **NOTE:** Please refer to the Additional JSON Element Dictionary section of this document for additional information on the preceding parameters. Employee JSON is required when service ID supports employee related operations. If you are not supporting employee related operations, you may skip the Employee JSON node.

5.14 ID Auto Extraction

The ID Auto Extraction functionality provides an option to capture the barcode, MRZ or QR code from an ID document, automatically extract the data from it, and use it to prefill data into an application, in offline mode. This is done with Service ID 620.

First, we need to call the below function with given parameters.

```
public void getAutoFillFieldInformation(Activity activityContext, JSONObject
idCaptureConfig);
```

idCaptureConfig	"id_country”:””, "id_state”:””, "id_type”:””

It will return if barcode or MRZ is available for the selected criteria on the below callback method.

```
public void onAutoFillFieldInformationAvailable(Map<String, String> result,
Response response);
```

result	FRONT : “BARCODE_MRZ” //BARCODE or MRZ BACK : “BARCODE_MRZ” //BARCODE or MRZ

After that we have to call the below method with given parameters.

```
public void autoFill(Activity activityContext, JSONObject idCaptureConfig,
JSONObject additionalJSON);
```

idCaptureConfig	"Id_country”:””, //same as above call "Id_state”:””, //same as above call "Id_type”:””, //same as above call "Id_side”:””, //Side which is received on above callback(FRONT/BACK)

After scanning the ID, it will return a response on the below call-back method.

```
public void onAutoFillResultAvailable(Map<String, String> resultMap,
Response response);
```

5.15 Offline Liveness Detection

The offline liveness detection is used to collect a series of selfie images from an individual to determine if they are a live person, not a spoof.

For using this service, pass a value of 660 in the Service ID in additional data information while calling “detectFace” function.

6 Additional SDK Features

The features listed in the section below are utilized in combination with the services listed above. For example, when executing Service ID 50 which is ID Validation + Face Matching with Customer Enrollment, an application could utilize all of the following features for a more robust enrollment:

- Image Capture with Image Processing
- Live Face Capture with Facial Biometric Matching
- Inclusion of Customer Name, Customer Phone, Customer Email, Customer Address details in Additional JSON
- Card Capture
- Voice Recording
- Video Recording
- Video Conferencing
- Fingerprint Capture
- GPS Capture
- Signature Capture
- Slant ID Capture
- Generic Document Capture (for capturing a proof of address document like a bank statement)

6.1 Card Capture

To capture a debit/credit card, use the following call.

```
ImageProcessingSDK.detectCard(activityContext);
ImageProcessingSDK.detectCard(activityContext, JSONObject additionalJSON);
```

Parameter	Type	Description
activityContext	Context	Instance of your activity

6.2 Voice Recording

To record voice, use the following call.

```
ImageProcessingSDK.startVoiceRecording(Activity activityContext, final int
recordingTime)

ImageProcessingSDK.startVoiceRecording(Activity activityContext, final int
recordingTime, boolean recordWithUI)

ImageProcessingSDK.startVoiceRecording(Activity activityContext, final int
recordingTime, boolean recordWithUI, JSONObject additionalJSON)
```

Table 28: Voice Recording Parameters

Parameter	Type	Default	Description
activityContext	Context		Instance of your Activity
recordingTime	Integer	10	Voice Recording time
recordWithUI	Boolean	0	Voice Recording voice in foreground or background

6.3 4F Fingerprint Capture

To capture four fingerprints using the device camera, use the following call.

```
ImageProcessingSDK.captureFourFingerprint(Activity activityContext,
JSONObject fingerCaptureConfig)
```

Table 29: Fingerprint Capture Parameters

Parameter	Type	Description
activityContext	Context	Instance of your Activity
fingerCaptureConfig	JSONObject	/***We can use below config JSON to pass all the feature related configuration parameter, if no value is sent it will use default value, All the value should be sent as string***/ Note: For customizing UI use “customizeUserInterface” method with JSON string. { "cfc_indexfinger_threshold": "60", "cfc_middlefinger_threshold": "60", "cfc_ringfinger_threshold": "60", "cfc_babyfinger_threshold": "30", "cfc_indexfinger_min_threshold": "20", "cfc_middlefinger_min_threshold": "20", "cfc_ringfinger_min_threshold": "20",

		"cfc_babyfinger_min_threshold": "15", "cfc_image_width": "450", "cfc_aggressiveness_factor": "1", "cfc_zoom_camera": "20", "cfc_ridge_width": "45", "cfc_finger_length_percentage": "40", "cfc_sharp_threshold": "20", "cfc_process_indexfinger": "Y", "cfc_process_middlefinger": "Y", "cfc_process_ringfinger": "Y", "cfc_process_babyfinger": "Y", "cfc_capture_left_hand": "Y" }

6.4 Standard Fingerprint Capture

To capture fingerprint, use the following call.

```
ImageProcessingSDK.captureFingerprint(Activity activityContext,
FingerprintDeviceType fingerprintDeviceType, FingerType fingerType, int
minNFIQValue, int deviceTimeout, int minImageSize)
```

Table 30: Standard Fingerprint Parameters

Parameter	Type	Default	Description
activityContext	Context		Instance of your Activity
fingerprintDeviceType	FingerprintDeviceType Class with two parameter Futronic, IB_Watson, MORPHO	Futronic	Fingerprint device to be selected.
fingerType	FingerType class		Finger to be selected
minNFIQValue	Int	3	Minimum nfiq value
deviceTimeOut	Int	50	Minimum time out value
minImageSize	Int	20	Minimum image size

6.5 Clear Fingerprint

To clear fingerprint, use the following call.

```
ImageProcessingSDK.clearFingerprint(FingerType fingerType)
```

Parameter	Type	Default	Description
fingerType	FingerType class	RTHUMB	Finger to be selected

6.6 Video Recording

To complete the video recording operation, use the following call.

```
ImageProcessingSDK.startVideoRecording(Activity activityContext,int
videoRecordingTime,JSONObject additionalJSON,String textOnVideoScreen)
```

Table 31: Video Recording Parameters

Parameter	Type	Default	Description
`activityContext`	Context		Instance of your Activity
`videoRecordingTime`	Int	`10`	Time limit to record a video
`textOnVideoScreen`	String		String data to be seen on video recording screen

6.7 Video Conferencing

To complete the video conferencing operation, use the following call.

```
ImageProcessingSDK.startVideoConferencing(Activity activityContext,
JSONObject videoConferencingData)
```

Table 32: Video Conferencing Parameters

Parameter	Type	Description
`activityContext`	Context	Instance of your Activity
`videoConferencingData`	JSONObject	Blank `JSONObject` for now

6.8 GPS Capture

To capture GPS coordinates, use the following call.

```
ImageProcessingSDK.completeOperation(activityContext)
```

Parameter	Type	Description
`activityContext`	Context	Instance of your Activity

6.9 QR / Barcode Capture

To capture QR Code or Barcode, use the following call.

```
ImageProcessingSDK.scanBarcode(Activity activityContext,JSONObject
additionalJSON)
```

Parameter	Type	Description
`activityContext`	Context	Instance of your Activity

6.10 Signature Capture

To capture a signature, use the following code.

```
ImageProcessingSDK.captureSignature(Activity activityContext)
ImageProcessingSDK.captureSignature(Activity activityContext, JSONObject
captureSignatureConfig)
```

Table 33: Signature Capture Parameters

Parameter	Type	Description
activityContext	Context	Instance of your Activity

captureSignatureConfig	JSONObject	{
		signature_capture_background: “N”
		}

6.11 Slant ID Capture

To capture an ID at a slant to observe different dimensions, use the following call.

```
autoCapture(Activity activityContext, ImageType imagesTypes, JSONObject
idCaptureConfig, JSONObject additionalJSON, HashMap<String, String>
snippetNameMap)
```

Table 34: Slant ID Capture Parameters

Parameter	Type	Description
activityContext	Context	Instance of your activity
imageType	Enum	ImageType.SLANTED_IMAGE_CAPTURE
idCaptureConfig	JSONObject	/***We can use below config JSON to pass all the feature related configuration parameter, if no value is sent it will use default value, All the value should be sent as string***/ Note: For customizing UI use “customizeUserInterface” method with JSON string. { id_capture_portrait : "Y", /***supported parameter Y/N ***/ id_light_threshold : "70", id_min_focus : "12", id_max_focus : "35", id_glare_percentage : "5", id_enable_capture_button_time : "60", id_max_image_size : "500", id_image_height : "1170", id_image_width : "800", id_capture_enable : "N",

		/***supported parameter Y/N ***/ id_generic_doc_fieldname : "" }

additionalJSON	JSONObject	JSON Object may contain following elements, those with a double // are mandatory. Please refer to the Additional JSON Element Dictionary section of this document for more details on each of the subsequent elements:
		{
		"Service_ID": "",//Mandatory
		"Manual_Review_Required" : ""
		"Bypass_Age_Validation" : ""
		"Bypass_Name_Matching" : ""
		"Deduplication_Required" : ""
		"Need_Immediate_Response" : ""
		"Capture_Secondary_ID" : ""
		"Unique_Customer_Number": "",
		"Old_Client_Customer_Number": "",
		"Customer_Name": "",
		"Customer_Type": "",
		"Customer_Phone": "",
		"Customer_Email": "",
		"Unique_Merchant_Number": "",
		"Unique_Employee_Number": "",
		"Unique_Employee_Code": "",
		"Gender": "",
		"Address_Line_1": "",
		"Address_Line_2": "",
		"Country": "",
		"State": "",
		"City": "",
		"Postal_Code": "",
		"Customer_Attribute": "",
		"AddressLine1": "",
		"AddressLine2": "",
		"Customer_Gender": "",
		"AgentCountry": "",
		"Encrypted_Data": "",
		"PreviousFormId": "",
		}
		
		NOTE: When performing a service that includes customer enrollment, the Unique_Customer_Number is mandatory.

snippetNameMap	HashMap	Related to snippetCapture field.

6.12 Generic Document Capture

To capture a generic document, use the following call.

```
imageProcessingSDK.captureGenericDocument(MainActivity.this,
capturePortraitChcBoxPOA.isChecked(), minimumLightThreshold,
minFocusScoreThreshold,focusScoreThreshold, glarePercentageThreshold,
enableCaptureButtonTime, maxImageSize,
highResolutionImageHeight, highResolutionImageWidth, addJSON,
new ColorCode(idImageOutlineColor, idImageOutlineAlpha),
new ColorCode(detectedIdImageOutlineColor, detectedIdImageOutlineAlpha),
new ColorCode(idImageOutsideOutlineColor, idImageOutsideOutlineAlpha),
new ColorCode(detectedIdImageOutsideOutlineColor,
detectedIdImageOutsideOutlineAlpha), titleBitmap,
showInstructionPOA.isChecked(), 0,fieldName);
```

Table 35: Generic Document Capture Parameters

Parameter	Type	Default	Range	Description
activityContext	Context			Instance of your activity
capturePortrait	boolean			For capturing image in portrait mode
ColorCode idOutlineColor : Color for face outline idOutsideOutlineColor : Color for area around outline outsideFaceOutlineColor : Color outside face outline	Object		Hex ColorCode, Transparency (1-100)	HexColorCode - Hexcolor string with RRGGBB format without # Transparency: int value (1-100), where 1 is no transparency and 100 is max transparency.
enableCaptureButtonTime	integer	15	5-200	No. of seconds after which manual capture is enabled
FieldName	String			Generate document according to field name
glarePercentage	integer	1	0-50	Maximum glare in percentage allowed
idCaptureConfig	JSONObject			/***We can use below config JSON to pass all the feature related configuration parameter, if no value is sent it will use default value, All the value should be sent as string***/

				Note:
				For customizing UI use “customizeUserInterface” method with JSON string.
				{
				id_capture_portrait : "Y",
				/***supported parameter Y/N ***/
				id_light_threshold : "70",
				id_min_focus : "12",
				id_max_focus : "35",
				id_glare_percentage : "5",
				id_enable_capture_button_time : "60",
				id_max_image_size : "500",
				id_image_height : "1170",
				id_image_width : "800",
				id_show_instruction : "Y",
				/***supported parameter Y/N ***/
				id_capture_enable : "F",
				/***supported parameter Y/N ***/
				id_generic_doc_fieldname : ""
				}

imageHeight	integer	830	640-1920	Image Height in pixels
imageWidth	integer	1170	480-1280	Image Width in Pixels
lightThreshold	integer	60	50-100	Minimum light required
maxFocusThreshold	integer	25	10-50	Minimum focus required
maxImageSize	integer	500	256-2048	Maximum image size in kb
maxImageSize	integer	500	256-2048	Maximum image size in kb
minFocusThreshold	integer	15	5-30	Bare minimum(fallback) focus required

```
imageProcessingSDK.captureGenericDocument(Activity activityContext,
JSONObject idCaptureConfig, JSONObject additionalJSON);
```

6.13 Delete Image

To delete captured image data, use the following call.

```
ImageProcessingSDK.deleteData();
```

7 Callback Instructions

To receive callbacks, the class should implement `ImageProcessingResponseListener` interface.

```
public class YourActivity implements ImageProcessingResponseListener
```

OR you can implement an individual listener

```
InitializationResultListener
AutoFillFieldInformationListener
AutoFillResultListener
AutoImageCaptureListener
BankStatementResultListener
BirthCertificateResultListener
CardCaptureListener
CreateEmployeeListener
CustomerVerificationResultListener
CustomizeUserInterfaceResultListener
DownloadXsltResultListener
ExecuteCustomProductListener
FaceDetectionListener
FaceMatchingResultListener
FingerprintCaptureListener
FingerprintEnrolmentListener
FingerprintVerificationListener
FourFingerCaptureListener
GenerateOTPListener
GenerateTokenListener
GenericApiCallResponseListener
GenericDocumentResultListener
GPSCoordinateListener
IDValidationAndVideoMatchingListener
ImageProcessingAndFaceMatchingResultListener
ImageProcessingResultListener
OperationResultListener
ProofOfAddressResultListener
ScanBarcodeListener
SignatureCaptureListener
SnippetCaptureResultListener
UpdateCustomerListener
UpdateEmployeeListener
VerifyAddressListener
VerifyEmployeeListener
VerifyOTPListener
VerifyTokenListener
VideoConferencingListener
VideoRecordingListener
VideoRecordingListener
```

Sequence diagrams describing how these callbacks are used in conjunction with the integration calls above, can be found in Section 5.

7.1 Customizing the User Interface

```
public void onCustomizeUserInterfaceResultAvailable(Map<String, String>
resultMap, Response response);
```

7.2 Automatic Image Capture

The callback will contain a Response object, which will consist of `Response.statusCode` and `Response.statusMessage`.

```
public void onAutoImageCaptureResultAvailable(Map<String, String>
resultMap,Response response);
```

Table 36: Automatic Image Capture Result Parameters

Result Parameter	Description
FRONT	For front image call, Result image will be available in `resultMap` with key `FRONT`.
BACK	For back image call, Result image will be available in `resultMap` with key `BACK`.

7.3 Processing the Image

```
public void onImageProcessingResultAvailable(Map<String, String> resultMap,
Response response);
```

Below is a sample of some of the fields that will be returned in the response. The results will vary based on the type of ID submitted (e.g. MRZ present or not, barcode, etc.)

Table 37: Processing the Image Result Parameters

Result Parameter	Description	Result Parameter	Description
`Status_Code`	Response status Code	`DocumentType`	ID type from the matched template
`Status_Message`	Response status message	`IssueDate`	Issued date
`Valid_Issue_Date`	Valid issue date flag	`IssuingCountry`	Country code of issuing country
`Address`	Full Address	`Low_Resolution`	Indicates if the image is high enough quality

Age_Over_18	Age over 18 flag	MRZ_Data	Machine readable zone data
Date_of_Birth	Date of birth	MRZ_Detected	Machine readable zone data detected on front/back flag
Exception	Image processing Java error message	MRZErrorMessages	MRZ check digit error
Expiration_Date	Date ID Expires	MRZValid	MRZ valid flag
Face_Detected	Face detection flag	Name	Full name from the ID
First_Name	First name from the ID	Last_Name	Last name from the ID
Middle_Name	Middle name from the ID	MatchingTemplateFound	Indicates if a matching ID template was found for the submitted ID
EnoughFeatureFound	Indicates if the total number of features meet thresholds	EnoughFeaturesFoundOnID_BACK	Indicates if the number of features meet thresholds for back
EnoughFeaturesFoundOnID_FRONT	Indicates if the number of features meet thresholds for front	Nationality	Nationality on the ID
FormID	Unique ID assigned to the form	ColorCode	Indicates if the image is color or greyscale
ID_Country	3-character ISO country code	Security_Feature_Match	Security feature matched
ID_Not_Expired	ID Expired flag	ID_State	ISO 2-digit state
ID_Number	ID/Document Number	State	Response state from the server based on the Svc ID
IDNumber2	Secondary ID number	Address Line 1	AddressLine1
Address Line 2	AddressLine2	Form_Key	Unique ID assigned by you or our server for the request
Request_Id	Unique ID assigned by our server for the request	Form State Code	This includes the 'Form State Code' based on 'Form State' which has been populated in response. For example, (If the Form State is

			'Approved' then Form State Code will be 00) All available Form State Codes are mentioned below in Appendix B. (For details description of sate codes please go through point number 10.)

7.4 Live Face Detection

```
public void onFaceDetectionResultAvailable(Map<String, String>
resultMap,Response response);
```

Table 38: Live Face Detection Result Parameters

Result Parameter	Description
`FACE`	Face image will be available in `resultMap` with key `FACE`.
`PROCESSED_FACE`	Processed face image will be available in `resultMap` with key `PROCESSED_FACE`.
`OVAL_FACE`	Processed face image with oval will be available in `resultMap` with key `OVAL_FACE`.

Please refer to the Automatic Image Capture callback table mentioned above for Response object description.

7.5 Process Image and Facial Biometric Verification

```
public void onImageProcessingAndFaceMatchingResultAvailable(Map<String,
String> resultMap, Response response);
```

Below is a sample of some of the fields that will be returned in the response. The results will vary based on the type of ID submitted (e.g. MRZ present or not, barcode, etc.)

Table 39: Process Image and Facial Biometric Result Parameters

Result Parameter	Description	Result Parameter	Description
`Status_Code`	Response status Code	`DocumentType`	ID type from the matched template
`Status_Message`	Response status message	`IssueDate`	Issued date

Valid_Issue_Date	Valid issue date flag	IssuingCountry	Country code of issuing country
Address	Full Address	Low_Resolution	Indicates if the image is high enough quality
Age_Over_18	Age over 18 flag	MRZ_Data	Machine readable zone data
Date_of_Birth	Date of birth	MRZ_Detected	Machine readable zone data detected on front/back flag
Exception	Image processing Java error message	MRZErrorMessages	MRZ check digit error
Expiration_Date	Date ID Expires	MRZValid	MRZ valid flag
Face_Detected	Face detection flag	Name	Full name from the ID
First_Name	First name from the ID	Last_Name	Last name from the ID
Middle_Name	Middle name from the ID	MatchingTemplateFound	Indicates if a matching ID template was found for the submitted ID
EnoughFeatureFound	Indicates if the total number of features meet thresholds	EnoughFeaturesFoundOnID_BACK	Indicates if the number of features meet thresholds for back
EnoughFeaturesFoundOnID_FRONT	Indicates if the number of features meet thresholds for front	Nationality	Nationality on the ID
FormID	Unique ID assigned to the form	ColorCode	Indicates if the image is color or greyscale
ID_Country	3 character ISO country code	Security_Feature_Match	Security feature matched
ID_Not_Expired	ID Expired flag	ID_State	ISO 2 digit state
ID_Number	ID/Document Number	State	Response state from the server based on the Svc ID
IDNumber2	Secondary ID number	Address Line 1	AddressLine1
Address Line 2	AddressLine2	Form_Key	Unique ID assigned by you or our server for the request
Request_Id	Unique ID assigned by our server for the request	FaceVerificationStatus	Face verification result

`Form State Code`	This includes the 'Form State Code' based on 'Form State' which has been populated in response. For example, (If the Form State is 'Approved' then Form State Code will be 00) This includes the 'Form State Code' based on 'Form State' which has been populated in response. For example, (If the Form State is 'Approved' then Form State Code will be 00) For all available Form State please reference the ID Validation / Form Status and State Codes section of this document.		

7.6 Customer Enrollment

```
public void onFingerprintEnrolmentFinished(Map<String,String > resultMap,
Response response)
```

Table 40: Customer Enrollment Result Parameters

Result Parameter	Description
`Status`	Response status Code
`Status_Message`	Response status message
`Request_Id`	Unique ID assigned by the server
`FormKey`	Unique ID assigned by the application or our server if not provided
`FormID`	Form identifier
`State`	Status of the request

7.7 Employee Enrollment

```
public void onCreateEmployeeFinished(Map<String,String> resultMap, Response
responses);
```

Table 41: Customer Enrollment Result Parameters

Result Parameter	Description
`Status`	Response status Code
`Status_Message`	Response status message
`Request_Id`	Unique ID assigned by the server
`FormKey`	Unique ID assigned by the application or our server if not provided
`FormID`	Form identifier
`State`	Status of the request

7.8 Customer Update

```
public void onUpdateCustomerFinished(Map<String, String> resultMap, Response
response);
```

Table 42: Customer Update Result Parameters

Result Parameter	Description
`Status`	Response status Code
`Status_Message`	Response status message
`Request_Id`	Unique ID assigned by the server
`FormKey`	Unique ID assigned by the application or our server if not provided
`FormID`	Form identifier
`State`	Status of the request

7.9 Employee Update

```
public void onUpdateEmployeeFinished(Map<String,String> resultMap, Response
responses);
```

Table 43: Employee Update Result Parameters

Result Parameter	Description
`Status`	Response status Code
`Status_Message`	Response status message
`Request_Id`	Unique ID assigned by the server
`FormKey`	Unique ID assigned by the application or our server if not provided
`FormID`	Form identifier
`State`	Status of the request

7.10 Customer Verification

```
public void onCustomerVerificationResultAvailable(Map<String, String>
resultMap, Response response);
```

Table 44: Customer Verification Result Parameters

Result Parameter	Description
`Status`	Response status Code
`Status_Message`	Response status message
`Request_Id`	Unique ID assigned by the server
`FormKey`	Unique ID assigned by the application or our server if not provided
`FormID`	Form identifier
`State`	Status of the request based on verification status
`CustomerVerificationStatus`	Biometric Verification result NOTE: Scores may be returned but are information only, the result should always come from this field.

7.11 Employee Verification

```
public void onVerifyEmployeeFinished(Map<String,String> resultMap, Response
responses);
```

Table 45: Employee Verification Result Parameters

Result Parameter	Description
Status	Response status Code
Status_Message	Response status message
Request_Id	Unique ID assigned by the server
FormKey	Unique ID assigned by the application or our server if not provided
FormID	Form identifier
State	Status of the request based on verification status
Employee_Verify_Status	Biometric Verification result NOTE: Scores may be returned but are information only, the result should always come from this field.

7.12 Customer Search

```
public void genericApiCallResponse(Map<String,String> resultMap, Response
responses);
```

Table 46: Customer Search Result Parameters

Result Parameter	Description
Status	Response status Code
Status_Message	Response status message
Request_Id	Unique ID assigned by the server
FormKey	Unique ID assigned by the application or our server if not provided
FormID	Form identifier
State	Status of the request based on Search status

7.13 Employee Search

```
public void genericApiCallResponse(Map<String,String> resultMap, Response
responses);
```

Table 47: Employee Search Result Parameters

Result Parameter	Description
`Status`	Response status Code
`Status_Message`	Response status message
`Request_Id`	Unique ID assigned by the server
`FormKey`	Unique ID assigned by the application or our server if not provided
`FormID`	Form identifier
`State`	Status of the request based on Search status

7.14 Generic API Callback

```
public void genericApiCallResponse(Map<String,String> resultMap, Response
responses);
```

Table 48: Generic API Result Parameters

Result Parameter	Description
`Status`	Response status Code
`Status_Message`	Response status message
`Request_Id`	Unique ID assigned by the server
`FormKey`	Unique ID assigned by the application or our server if not provided
`FormID`	Form identifier
`State`	Status of the request

7.15 Card Capture

```
public void onCardDetectionResultAvailable (Map<String, String> resultMap,
Response response);
```

Table 49: Card Capture Result Parameters

Result Parameter	Description
`CardType`	Card type
`CardNumber`	Card number
`CardExpiryDate`	Card expiry date
`CARD`	Redacted card image

Please refer to the Automatic Image Capture callback table mentioned above for Response object description.

7.16 Voice Recording

```
public void onVoiceRecordingFinished(Map<String, String> resultMap, Response
response);
```

Result Parameter	Description
`VOICE_RECORDING`	Recorded Voice file path will be available in `resultMap` with key `FILEPATH`.

7.17 4F Fingerprint Capture

```
public void onFourFingerCaptureFinished(Map<String,String > resultMap,
Response response);
```

Result Parameter	Description

`CAPTURE_4F_FINGERPRINT`	Fingerprint images will be available in `resultMap` with keys:
	`LRFINGER`
	`LMFINGER`
	`LIFINGER`
	`LLFINGER`
	`RRFINGER`
	`RMFINGER`
	`RIFINGER`
	`RLFINGER`

7.18 Standard Fingerprint Capture

```
public void onFingerprintCaptureFinished(Map<String,String > resultMap,
Response response);
```

Result Parameter	Description
CAPTURE_FINGERPRINT	Finger image, Finger type, Finger Count will be available in resultMap with keys FINGEPRINT, FINGER_TYPE, FINGER_COUNT.

7.19 Video Recording

```
public void onVideoRecordingFinished(Map<String,String > resultMap, Response
response);
```

Result Parameter	Description
VIDEO_RECORDING	Recorded Video file path will be available in resultMap with key FILEPATH.

7.20 Video Conferencing

```
public void onVideoConferencingFinished(Map<String,String > resultMap,
Response response);
```

Result Parameter	Description
Response	Please check “Response Status Code” for detail

7.21 GPS Capture

```
public void onGPSCoordinateAvailable(Map<String, String> resultMap, Response
response);
```

Result Parameter	Description
GPS_CAPTURE	GPS status message is returned in response

7.22 QR / Barcode Capture

```
public void onScanBarcodeFinished(Map<String, String> resultMap, Response
response);
```

Result Parameter	Description
SCAN_BARCODE_DATA	Barcode/QR code image, Type will be available in resultMap with key SCAN_BARCODE_DATA

7.23 Signature Capture

```
public void onCaptureSignatureFinished(Map<String, String> resultMap,
Response response);
```

Result Parameter	Description
`CAPTURE_SIGNATURE_DATA`	Signature image will be available in `resultMap` with key `CAPTURE_SIGNATURE_DATA`

7.24 Generic Document Capture

```
public void onCaptureGenericDocumentResultAvailable(Map<String, String>
resultMap, Response response);
```

Result Parameter	Description
`GENERIC_DOCUMENT`	Generic Document image will be available in `resultMap` with key `GENERIC_DOCUMENT`.

7.25 Initialization Response

```
public void onInitializationResultAvailable(Map<String, String> resultMap,
Response response);
```

Table 50: Initialization Response Result Parameters

Result Parameter	Description
`Status`	Response status Code
`Status_Message`	Response status message

8 ID Validation / Form Status and State Codes

All process image callbacks include the `formStatus`, which shows the results of the ID Validation and additional processing if completed. The table below describes each `formStatus` that can be returned, the recommended actions to be taken by your application and the state codes that can be returned.

Table 51: Form Status Returns and State Codes

Form Status	Description	Recommended Actions	State Code
Approved	• All checks in our system have passed • Data was extracted with the level of accuracy within our defined thresholds based on image quality.	• No action required • ID is approved	00
Barcode Unreadable	If a barcode is expected to be present, and the barcode data is not detected by our system.	The application using this SDK should prompt the user to recapture the images where the barcode is present in better lighting without glare so the ID can be processed properly.	50
Expired ID	If an ID has an expiration date that is expired as of today's date.	The application using this SDK should prompt the user to provide an ID that is not expired.	40
Face Not Detected	• A face is *not* detected on an image. • Typically, this occurs when the face is not present on the ID • In some cases, is due to poor image quality or image glare.	The application using this SDK should prompt the user to provide a clear image of an ID that has a photo included.	31
Facial Biometric Match Failed	If a customer photo is captured and does not biometrically match the photo in the ID document, the ability to review the images is in the back-office.	The application using this SDK should prompt the user to capture a new selfie image.	30
ID Back Image not found	• The back of an ID is expected to be present • The back image was not sent *or* • A blank image was provided	The application using this SDK should prompt the user to provide a clear image of the back of the ID.	10

ID Front and Back Mismatch	If the data extracted from the back of the ID image does not belong to the presented front of the ID image.	The application using the SDK should prompt the user to provide new front and back images because inconsistences were found. **Note**: If this occurs more than once, the application could decide to push this to a manual review queue because it could be an issue with OCR extraction accuracy.	60
ID Front Image Not Found	If the front of an ID is not uploaded or a blank image was provided.	The application using this SDK should prompt the user to provide a clear image of the front of the ID.	11
ID Not Accepted	If the uploaded ID is configured invalid for the selected country.	The application using this SDK should prompt the user to provide an ID that is in the list of accepted ID Types. **Note**: The application should prompt the user to select an ID type from a dropdown list, which reduces the likeliness of this result.	61
ID Type Mismatch	If the ID Type submitted doesn't match the ID Type extracted from the ID Image.	The application using this SDK should prompt the user to select the ID Type that matches the Type of ID that was uploaded/captured.	62
Invalid ID Number	The ID Number on the ID *does not* match the pattern defined in the ID Template.	The application using this SDK should prompt the user to provide a new ID with a valid ID number.	63
Key Component Not Visible	A key component that is expected on the ID is missing such as the date of birth, expiration date, photo, etc.	• The application using this SDK should prompt the user that: • The ID they have provided is not valid or the image provided may be blurred or unreadable in some portions • To recapture or capture an image of a different ID.	12

Low Resolution	• The front or back image that is uploaded has been detected as low resolution *or* • The ID image is *not* a color/grayscale image based on resolution thresholds defined by IDmission.	The application using this SDK should prompt the user to recapture the images in better lighting without glare so the ID can be processed properly.	13
MRZ and ID Data Mismatch	If the data extracted from the MRZ does not match the data extracted from the ID image itself.	The application using the SDK should prompt the user to provide new image(s) because inconsistences were found. **Note**: If this occurs more than once, the application could decide to push this to a manual review queue because it could be an issue with OCR extraction accuracy.	64
MRZ Invalid	If the MRZ Validation flag (Machine Readable Zone) results in determining that the MRZ Data was not correct according to the standards.	The application using this SDK should prompt the user to upload a new image and ensure there is no glare or damage to any of the data on the image.	65
Name Match Failed	If the name submitted for the Customer or the Employee doesn't meet the matching thresholds against the name extracted from the ID Document.	The application using this SDK should prompt the user to provide a valid ID as this was deemed invalid.	66
No Image Found	If the ID front and back images are black or white without any ID Image.	The application using this SDK should prompt the user to manually re-enter the name.	14
Pending Review	If no other conditions are met that push an ID into the listed states, it will go into Pending Review.	The application using this SDK should prompt the user that the ID could not be validated, and they should upload a new ID.	N/A
Sample ID	If an ID has been decisioned by the manual review team as a sample.	The application using this SDK should prompt the user that: • The ID they have provided is not valid • To capture a different ID image.	67

Security Feature Invalid	If one or more security features defined for the ID template are detected as not matching with the image provided.	The application using the SDK could also decide to handle this directly and alert the user that the ID is invalid and to capture new images.	68
Suspected Sample	If an ID has been detected as a sample (logic has been configured to recognize sample ID numbers, names, watermarks and other suspected sample data.)	The application using this SDK should prompt the user that the ID they have provided is not valid and to capture an image of a different ID.	69
Suspected Tampering	• Photo editing software (e.g. Microsoft Paint, Adobe Photoshop) was detected. • Dates have been falsely created or modified in the ID document image.	The application using this SDK should prompt the user that the ID they have provided is not valid and to capture an image of a different ID.	70
Tampered ID	If an ID has been decisioned by the manual review team as being tampered with / edited in some way.	The application using this SDK should prompt the user that the ID they have provided is not valid and to capture an image of a different ID.	71
Template Not Found	The correct template cannot be matched against those configured for the client. **Note**: The system will not find a template to associate with the image for correct data extraction and validation.	The application using the SDK should prompt the user to capture a different ID.	15
Under the age of 18	If the date of birth (extracted from the barcode/MRZ) calculates the individual as under the age of 18.	The date of birth indicates that the user is under the age of 18 **Note**: The SDK can determine if this was already indicated in the application. If it was not indicated, the SDK should prompt for correct ID.	72

9 Pending Manual Review ID Validation Results

If Manual Review is requested and the form moves into a 'Review…' `formStatus`, this is the list of values that the `formStatus` will show until manual review is completed. These are not the final form states, this is just a temporary `formStatus` that will be present until review is completed.

Table 52: Pending Manual Review Form Status Returns

Form Status	Description	Recommended Actions
Review Barcode	If a barcode is expected to be present, and the barcode data is not detected by our system.	Manual review of the ID in the Back Office to verify the information on the front of the ID against what is shown on the back.
Review Expired ID	If an ID has an expiration date that is expired as of today's date.	Manual review of the ID in the Back Office to determine if the ID is truly expired or not.
Review Face Not Detected	If a face is not detected on an image. The main reason for this is that the face was not present on the ID, however in some cases this could be due to poor image quality, or if there is a glare throughout the image.	Manual review of the ID in the Back Office to determine if a face is actually present on the ID.
Review Facial Biometric Match Not Verified	If a customer photo is captured and does not biometrically match the photo in the ID document, the ability to review the images is in the back office.	Manual review of the ID in the Back Office to determine if the Customer Photo is a match to the photo on the ID. If the person appears to be the same, it would be Approved, if not, it would be marked as 'Facial Biometric Match Failed'.
Review ID Back Image not found	If the flags that determine a back image is not found are questionable, then the ability to review the ID back image is available in the back office.	Manual review of the ID back image in the Back Office to determine if its present and valid.

Review ID Front and Back Mismatch	If the back of an ID is expected to be present and if the data extracted from the front of the ID data does not match the data extracted from the barcode, then the ability to review the data is available in the Back Office.	Manual review of the ID in the Back Office because the data from the front does not match the data extracted from the barcode. The back office user will need to be able to see the ID Number field that was extracted from the front of the ID vs what was extracted from the barcode. They can manually compare this to the image to see if it's actually a mismatch or if it was an issue with OCR extraction.
Review ID Front Image Not Found	If the front of an ID is not uploaded or a blank image was provided.	Manual review of the ID front image in the Back Office to see if its present and valid.
Review ID Type Mismatch	If the ID Type submitted doesn’t match the ID Type extracted from the ID Image.	Manual Review of the ID in the Back Office to determine if the template match is correct based on the ID Type submitted.
Review Key Component Not Visible	A key component that is expected on the ID is missing such as the Date of birth, expiration date, photo, etc.	Manual Review of the ID in the Back Office to determine if the components are present but just weren’t extracted.
Review Low Resolution	If either the front or back image that is uploaded has been detected as low resolution or the ID image is not a color/gray scale image. This is based on resolution thresholds defined in our system.	Manual review of the ID in the Back Office to determine if the ID images have a clear enough resolution to be reviewed and validated.
Review MRZ and ID Data Mismatch	If the data extracted from the MRZ does not match the data extracted from the ID image itself.	Manual review of the ID in the Back office because the data from the front does not match the data extracted from the MRZ. The back-office user will need to be able to see the ID Number field that was extracted from the front of the ID vs what was extracted from the MRZ. They can manually compare this to the image to see if it's actually a mismatch or if it was an issue with OCR extraction.

Review MRZ Invalid	If the MRZ Validation flag (Machine Readable Zone) is questionable, and MRZ Data is expected to be present, then the ability to review the MRZ is available in the back office. An MRZ parsing tool is provided in the event the MRZ extracted might have a few characters transposed.	Manual Review of the ID in the Back Office. The MRZ String should be provided and an MRZ parsing tool should be present for the back-office user to utilize. In most cases, this is due to a glare on the MRZ so the data was extracted improperly. It can be manually fixed and tested to ensure the MRZ is truly valid.
Review Name Match Failed	If the name submitted for the Customer or the Employee doesn't meet the matching thresholds against the name extracted from the ID Document.	Manual Review in the Back Office of the name submitted against the name extracted from the ID.
Review Security Feature Invalid	If one or more security features defined for the ID template are detected as not matching with the image provided.	Manual Review in the Back Office. This occurs when the ID Template the image is matched against indicates the placement of the security feature is off or is not present.
Review No Image Found	If the ID front and back images are blank or white without any ID Image.	Manual Review in the Back Office. This occurs when no ID is recognized in the photo.
Review Template Not Found	If the image provided does not match any ID templates that are configured for a client, it can be reviewed by a back office team to determine if the image was just of poor quality and if it should be accepted OR if it should be rejected OR if the template needs to be setup and configured for that client.	Manual Review in the Back Office. This is generally due to poor image quality, so a manual check is required to see if it's truly a valid and accepted ID type for the region / customer.
Review Under the age of 18	If the date of birth (extracted from the barcode/MRZ) calculates the individual as under the age of 18.	Manual review of the ID in the Back Office to determine if the date of birth extracted has resulted in a proper calculated age.

10 Post Processing POST Update API Request Details

The below JSON format will be sent via an API once processing is complete by IDmission. A URL must be provided that this JSON data will be posted to. These URLs need to be defined in the dashboard for the Sandbox and Production environments through **My Account** > **Account Settings**.

JSON POST Update API Example:

```
{

"Form_Status":"Approved",
"Form_Id":16725444,
"Product_Id":1523,
"Form_Key":"IDMTPIA_1553497978827_10:07:B6:43:3A:65","
"ID_Type":"DL",
"ID_State":"AZ"
"ID_Country":"USA"
"Name":"JANE D DOE",
"First_Name" : "JANE",
"Middle_Name" : "D",
"Last_Name" : "DOE",
"Address" : "123 E TEST ST MESA AZ 85212",
"Address1":"123 E TEST ST",
"Address2":"MESA AZ 85212"
"Country":"USA",
"ID_Number":"B1111111",
"ID_Number_Match_Result":"Matched"
"IDNumber1",
"IDNumber2",
"IDNumber3",
"Valid_ID_Number" : "Y",
"Date_of_Birth":"01/01/1980"
"Date_of_Birth_Formatted" : "01/01/1980",
"Expiration_Date":"01/01/2025",
"Expiration_Date_Formatted":"01/01/2025"
"Issue_Date":"03/01/2012",
"Issuing_Country":"USA",
"Face_Detected" : "Y",
"MRZValid" : "NA",
"MRZData" : "MRZString",
"FaceVerificationStatus ":"Verified",
"Manual_Review_Executed":"Y",
}
```

Table 53: Post API Element Dictionary

Field Name	Field Type	Response Description
Header Node		
Form_Status	Text	This is the final result of the ID Validation and Face Matching processes. For a full list of values, please refer to Appendix C.
Form_Id	Text	Form ID that was generated by IDmission and provided in the original API response.
Product_Id	Text	Product ID for the environment, echo back from the request.
Form_Key	Text	Form_Key from the original request.
Form_Data Node NOTE: These are all of the values that are extracted from the ID images themselves or are result fields from the ID Validation process. If a value is not present for a particular field for a request, it will not be included in this JSON response.		
ID_Type	Text	In most cases, this is an echo back from the request, unless it was updated by the ID Verification Team because it was incorrect.
ID_Country	Text	In most cases, this is an echo back from the request, unless it was updated by the ID Verification Team because it was incorrect.
ID_State	Text	In most cases, this is an echo back from the request, unless it was added/updated by the ID Verification Team because it was not provided or incorrect.
Name	Text	Full Name extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present.
First_Name	Text	First Name extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present.
Middle_Name	Text	Middle Name extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present.
Last_Name	Text	Last Name extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present
Last_Name_2	Text	If there is a secondary last name, it will display as extracted from either the barcode, MRZ, or front of the ID when MRZ/barcode are not present.
Address	Text	Full Address extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present.
Address1	Text	Address Line 1 (usually the Street Number and Street Name) extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not presen.

Address2	Text	Address Line 2 (e.g. Suite Number, Unit) extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present.
Country	Text	City parsed from a barcode or MRZ, only when a one is present. In most cases the address will be in AddressLine1 / 2.
ID_Number	Text	ID Number extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present.
IDNumber1	Text	Additional IDNumber 1, extracted from barcode/MRZ data, or if neither are present, from the front of the ID. This will only be present if this additional ID Number component is defined in the matching ID template.
IDNumber2	Text	Additional IDNumber 2, extracted from barcode/MRZ data, or if neither are present, from the front of the ID. This will only be present if this additional ID Number 2 component is defined in the matching ID template.
IDNumber3	Text	Additional IDNumber 3, extracted from barcode/MRZ data, or if neither are present, from the front of the ID. This will only be present if this additional ID Number 3 component is defined in the matching ID template.
Valid_ID_Number	Text	Flag that indicates of the format of the ID number matches what is defined in the matching ID template. • Flag = Y if the ID Number extracts correctly and matches the format. • Flag = N if the ID Number does not match the format. • Flag = NA If there is no specific format defined • Flag = D if it cannot be clearly determined (D = Doubtful)
Date_of_Birth	Text	Date of Birth extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present. This is the text that is extracted directly, not always a formatted date.
Date_of_Birth_Formatted	Date	Date of Birth extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present. If the date cannot be formatted based on extraction (e.g. Odd format, not present in the MRZ and not corrected by manual review) then this field would be blank. Format YYYY/MM/DD
Expiration_Date	Text	Expiration Date extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present. This is the text that is extracted directly, not always a formatted date.

Expiration_Date_Formatted	Date	Expiration Date extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present. If the date cannot be formatted based on extraction (e.g. Odd format, not present in the MRZ and not corrected by manual review) then this field would be blank. Format YYYY/MM/DD
Issue_Date	Text	Issue Date extracted from either barcode, MRZ, or front of the ID when MRZ/barcode are not present. In most cases this is not present in a machine readable portion of the ID so we often see many variations of formats. This will be sent as a Text field.
IssuingCountry	Text	This is the 3-digit ISO country code in which the identity document was issued, either extracted from the barcode/MRZ or as defined in the template.
Face_Detected	Text	This flag indicates whether or not a clear/visible face is found on the identity document. If a 'Photo' component is captured for the specific ID template matched, and a face is clearly detected, this flag will = Y. If a face is expected and is NOT clearly detected, this flag will = N. If there is no 'Photo' component captured for the template, this flag will be blank.
MRZ_Data	Text	This is the raw data extracted from the MRZ. There is a parsing tool (in the back office portal) where you can copy and paste the extracted MRZ string, compare it to the image, and correct any errors that may have resulted during OCR. This way you can determine the validity of the MRZ without having to ask the customer to resubmit their identity documents.
MRZValid	Text	This flag indicates the validity of the MRZ data, this will = Y if it is valid (passes all of the check digits), otherwise this will = N. When flag = N, there are usually MRZ Error Messages associated with. When there is not an MRZ detected (MRZ Detected = N), this flag will be blank. NA when there is a template matched, but no MRZ expected.
FaceVerificationStatus	Text	This is the result of the Face Match – 'Verified' or 'Not Verified' will be the values. A verified face must have a score that equates to a 62% or higher match.
Manual_Review_Executed	Text	If a request was submitted with the 'Manual_Review_Required' parameter = Y, and a manual review was actually executed, this field will = Y. If not, it will = N or Blank.

10.1 Post API Expected Acknowledgement Response

The system who is accepting this POST API from the IDmission server should respond with a simple '200' HTTP successful acknowledgement response. Example shown below:

```
HTTP status code = 200

response json formatted as
{
"status_code" : 0,
"status_message" : "Success"
}
```

11 Additional JSON Element Dictionary

Table 54: Additional JSON Elements

Additional Data Element Name	Description	Valid Values
Service_ID	This is the ID of the service that you wish to execute. *ALWAYS MANDATORY FOR ALL REQUESTS	Refer to Section 2.1
Manual_Review_Required	If the ID validation that is performed on the ID images submitted doesn't result in an auto-decision, a manual review can be requested by our back office review team. This does incur an additional processing fee per review.	• Y – Review required if auto-decision results in a 'Review' form state. • N – No manual review will be performed • Blank – default, no manual review will be performed
Bypass_Age_Validation	By default, for ID image validation, our server requires the age (calculated from today's date – date of birth) be 18 or older. To bypass this validation, this parameter is used.	• Y – Bypass age validation for 18 and older. • N – Age validation will be performed; age must be 18 or older. • Blank – default, age validation will be performed.

Bypass_Name_Matching	By default, when a customer or employee name is submitted through additional data, the name submitted will be matched against the name extracted from the ID images. To bypass this name matching feature, this parameter is used.	- Y – Bypass name matching. - N – Name matching is performed. - Blank – default, Name matching is performed.
Deduplication_Required	Deduplication check is used for verifying if the input person data is matched with existing records or not.	- Y –It will check whether person data matches with existing record if matched it will return matched person records in response. - N –It will not check for existing record matching.
Need_Immediate_Response	This flag is used to get response Asynchronously or Synchronously.	- Y – Response will be received immediately, and actual process will execute Asynchronously. - N – Process will execute synchronously, and response will be returned once process complete.

`Capture_Secondary_ID`	This flag is used to capture secondary ID Documents and supports ID Validation, Face Matching.	• `Y`= When Capture Secondary ID =`Y` and information is captured, need to execute ID Validation and Face match calls and apply the business rules to move forms into respective states. • `N`=When Capture Secondary ID=`N` then exiting flow will work. No need to capture secondary ID documents.
`Unique_Customer_Number`	Unique ID assigned by the client to associate with a customer. This will be required for verification services.	Alphanum
`Old_Client_Customer_Number`	Unique ID assigned by the client to associate with a customer. This will be required for update services.	Alphanum
`Customer_Name`	Name of the Customer that will be stored with the request.	String
`Customer_Type`	Type of Customer that will be stored with the request	String
`Customer_Phone`	Customer's phone number that will be stored with the request.	Numeric
`Customer_Email`	Customer's email that will be stored with the request.	Must be a valid email format with @ and extension.
`Unique_Merchant_Number`	Unique ID for a store or location that an employee or customer is to be associated with.	String
`Unique_Employee_Code`	Unique ID assigned by the client to associate with an employee. This will be required for verification services.	String
`Gender`	Customer's gender that will be stored with the request	M, F

Address_Line_1	Line 1 of the Customer’s Address that will be stored with the request.	String
Address_Line_2	Line 2 of the Customer’s Address that will be stored with the request.	String
Country	3-digit ISO Country Code of the Customers Address that will be stored with the request.	3-digit ISO Country Code
State	State of the Customer’s Address that will be stored with the request.	String (State Name is used here, not a code value).
City	City of the Customer’s Address that will be stored with the request.	String
Postal_Code	Postal Code of the Customer’s Address that will be stored with the request.	String

12 EmpJSON Element Dictionary

Table 55: EmpJSON Elements

Element	Description	Valid Values
Employee_Code	Unique ID assigned by the client to associate with the employee. This will be required for verification services.	String
Employee_Type	Associated role for the employee that will align with access rights.	Admin Agent Manager Operator
Login_ID	Login ID to associate with an Employee.	String (if not provided this will be copied from the Employee_Code)
Employee_Email	Email address associated with the employee.	Must be a valid email format with @ and extension.
Employee_Name	Name to be associated with the employee.	String
Employee_Mobile_Number	Phone number to be associated with the employee.	Numeric

`Employee_ID_Number`	ID Number to be associated with the employee.	String
`Employee_Gender`	Gender of the employee.	M, F
`Employee_AddressLine1`	Line 1 of the Employee's Address that will be stored with the request.	String
`Employee_AddressLine2`	Line 2 of the Employee's Address that will be stored with the request.	String
`Employee_City`	City of the Employee's Address that will be stored with the request.	
`Employee_State`	State of the Employee's Address that will be stored with the request.	String (State Name is used here, not a code value).
`Employee_ZipCode`	Zip or Postal Code of the Employee's Address that will be stored with the request.	
`Employee_Country`	3-digit ISO Country Code of the Employee's Address that will be stored with the request.	3-digit ISO Country Code

13 Troubleshooting

This section includes general SDK troubleshooting information.

13.1 initializeAppItSDK function

For the parameter `initializeAppItSDK`, send the activity context (for Android only).

13.2 AppItFramework

To setup `AppItFramework`, perform the following steps:

1. Add **AppItFramework.framework** into your project
2. Embed and Sign it on the **General** tab
3. If you are running the latest XCODE version continue to step 4. If you are using a previous version, skip to step 5
4. Select the **Screenshot** checkbox
5. (Users running previous XCODE versions only) then you need to follow below steps:
   a. Drag and drop AppItFramework framework into your project.
   b. Select your project
   c. Navigate to the **General** tab
   d. Add AppItFramework framework entry in Embedded Binaries
   e. Remove duplicate entry from Linked Framework and Libraries

6. Add podfile and insert the following dependencies in the podfile:
```
pod 'WebRTC
pod 'CocoaLumberjack'
pod 'SocketRocket'
pod 'ZXingObjC', '~> 3.2.2'
```

13.3 Image Processing Initialization Calls

The `ImageProcessingSDK.initialize()` call is a synchronous call in Android. When initialization is successful, it will not throw an exception (which means it is not providing a callback method).

`ImageProcessingSDK.initializeAsync` is an asynchronous call also used for initialization. This call results in callback methods on `onInitializationResultAvailable`.

14 Document Version History

Table 56: Revision details

Reason of change	Revised By	Date	Version
Updated properties, cover page, footers and introduction.	Amanda	13-August-2020	7.2
Updated the Form Status Returns and State Codes table (Table 59), hyperlinks, and format	Radhika	16-Jul-2020	7.2
Removal of Initialize SDK function (section 3.3), updated: table numbers, table of contents (TOC), doc date and footers.	Amanda	02-Jul-2020	7.2
Document reformatting and stylization	Amanda	25-Jun-2020	7.2
Updated initialization call	Dipen	25-May-2020	6.7
Added “fd_enable_passive_face_detection” config parameter in detectFace function	Dipen	30-Apr-2020	6.6
Added ID Auto Extraction, offline liveness detection changes, voice recording changes, callback method optional changes and added form ID in additional data screen. Also updated formatting for headings for table of contents	Anagha/Angela	18-Mar-2020	6.5
Dependency changes for passive face detection update.	Dipen	22-Jan-2020	6.4
-Id label shadow and button color. -Capture secondary ID documentation. -Signature background configuration. -Barcode scanning support in ID capture feature.	Dipen	09-Jan-2020	6.3
Updated dependencies for SDK version 6.4.4.2 or above(For AndroidX support)	Monu	25-Nov-2019	6.2

Added form state description in process image and process image and face match points and added form state codes table.	Anagha	13-Nov-2019	6.1
Updated 4F UI configuration json. Added country, state, idtype parameter in front/back ID capture to enable barcode extraction.	Dipen	10-Oct-2019	6.0
Minor revisions to text explanations for revisions 5.6 through 5.8, final review prior to release on Oct 6.	Angie	16-Sep-2019	5.9
Added callback functions for Generic API, Customer Search and Employee Search	Anagha	09-Sep-2019	5.8
Slant Id Capture	Anagha	09-Sep-2019	5.7
Main functions for Employee search and Customer search.	Anagha	09-Sep-2019	5.6
Generic API call	Dipen	28-Jul-2019	5.5
Video conferencing dependency and calls	Dipen	26-Jul-2019	5.4
Added instruction and preview screen parameter in common UI config call.	Dipen	24-Jun-2019	5.3
Added documentation for every parameter in "customizeUserInterface" call.	Dipen	18-Jun-2019	5.2
-Async initialization call. -Customize UI configuration method("customizeUserInterface") -Custom overlay for face detection -4F Fingerprint	Dipen	10-Jun-2019	5.1
Revision of entire document to align with the eVolv Service IDs. Removed irrelevant sections that do not apply to eVolv.	Angela	24-Mar-2019	5.0
Added new API with UI Config JSON for ID and Face detection	Dipen	29-Jan-2019	4.7
Face contour detection changes	Pranjal	28-Nov-2018	4.6
Generate token changes	Dipen	22-Jul-2018	4.5
Added face detection API with toggle camera option	Dipen	18-Jul-2018	4.4
Employee verification and POA verification	Anagha	13-Jul-2018	4.3
Signature capture, QR/Barcode, Capture Generic document capture	Monu	18-June-2018	4.2
Final revisions from feedback for eVolv branding and details	Angie	15-June-2018	4.1
Rebranded to eVolv and added sequence diagrams for standard use cases, renumbered sections.	Angie	13-June-2018	4.0
Removed threshold parameters from intialization call and sent those parameter in individual call.	Dipen	01/March/2017	3.9
Added new API call and update callback method description.	Dipen	25/February/2017	3.8
Updated face-detection API call	Dipen	16/February/2017	3.7

Updated default values and callback method parameter	Dipen	09/February/2017	3.6
Updated default values	Pranjal	30/January/2017	3.5
Added Overloaded method details	Suraj	27/January/2017	3.4
Added card capture call, callback. Updated parameters and callback values.	Pranjal	25/January/2017	3.3
Added Voice recording, Capture 4F Fingerprint, Capture Finger, Clear Fingerprint, Clear All Fingerprint, Fingerprint Enrolment, Fingerprint Verification ,Video Recording, Get GPS Location.	Sanket	11-May-2018	3.24
Id capture and face detection API update for showing instruction screen	Dipen	12-Apr-2018	3.23
Face Detection : Returning white oval image and allowing it to process. Customize background color of ID capture screen and face detection screen on detection.	Dipen	5-Apr-2018	3.22
Added “setImages” API to allow user to pass images to SDK so that it can be processed in SDK flow	Dipen	23-Mar-2018	3.21
Added overloaded API for “Customize User Interface”	Dipen	30-Jan-2018	3.20
Updated parameters and callback values	Pranjal	30/November/2016	3.2
Added “Customize User Interface”	Dipen	26-Jan-2018	3.19
Added API documentation for passing ColorCode -Face Detection -Image Capture(Front, Back, POA, Bank Statement, Birth Certificate)	Dipen	11-Jan-2018	3.18
Documented API with clear-form-key parameter -Process Image -Match Face -Process Image and Match Face	Dipen	05-Dec-2017	3.17
Updated Integration>Dependencies with new dependency project.	Dipen		3.16
Added overloaded method with final-submit flag for following calls: -Process Image -Match Face -Process Image and Match Face	Dipen	24-Oct-2017	3.15
Added new face-detection API description to enable/disable preview screen.	Dipen	05/Oct/2017	3.14
Added customer name node (Customer_Name) in additional data.	Dipen	05/May/2017	3.13
Added overloaded initialization API	Dipen	04/May/2017	3.12

Added customer verification API and callback, added following API call with additional data : -Card Detection -Capture POA -Capture Bank Statement -Capture Birth Certificate	Dipen	11/March/2017	3.11
Added capture bank-statement and capture birth-certificate API and callback.	Dipen	10/March/2017	3.10
Additional changes for making configurable product	Suryaprakash	10/November/2016	3.1
Changes for configurable product.	Pranjal	24/October/2016	3.0
Added details for input and callback parameters.	Pranjal	27/July/2016	2.9
Added face capture and verification call details	Pranjal	11/July/2016	2.8
Added form status table	Pranjal	31/May/2016	2.7
Updated parameters, URL, types and default values.	Pranjal	30/May/2016	2.6
Formatting updates	Angie	26/May/2016	2.5
Updated dependencies and callback parameters.	Pranjal	25/May/2016	2.4
Modified setup and integration section. Added callback section.	Pranjal	24/May/2016	2.3
Applied same font and color to all code snippets. Modified function parameters.	Pranjal	20/May/2016	2.2
Modified and added to minimum requirements, setup and integration. Removed certain sections.	Pranjal	19/May/2016	2.1
Full revision, inclusion of various instructions, response data, and samples to go along with the sample SDKs.	Pranjal	16/May/2016	2.0
Updates for integration	Mohan / Suraj	11/May/2016	1.1
Initial draft	Mohan	01/May/2016	1.0

Copyright

This publication and the information herein are furnished AS IS, is subject to change without notice, and should not be construed as a commitment by IDmission LLC. IDmission LLC assumes no responsibility or liability for any errors or inaccuracies, makes no warranty of any kind (express, implied or statutory) with respect to this publication and expressly disclaims any and all warranties of merchantability, fitness for particular purposes and non-infringement of third-party rights.