SDKS

- Android SDK Guide
- iOS SDK Guide
- JavaScript SDK Guide
- Liveness API Documentation
- SDK response update 3.1.0.0

Android SDK Guide

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

Requirements and Compatibility

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

- **Android Studio**: The latest version is recommended for optimal compatibility.
- **Minimum SDK Version**: Android SDK version 24 (Nougat) or higher.
- **Target SDK Version**: Android SDK version 35 (Android 15) to ensure your app is compatible with the latest Android OS.
- **Compile SDK Version**: Android SDK version 36.

Installation

1. Add the library

Download the "hybridComponent_3_0_0_17.aar" library and add it to your project's `libs` folder. Ensure you configure your project's `build.gradle` file to include the library as a dependency:

> dependencies {
> implementation(name: 'hybridComponent_3_0_0_17', ext: 'aar')
> }

2. Import Required Libraries

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

> Java
>
> import mabel_tech.com.scanovate_sdk.ScanovateSDK;
>
> import mabel_tech.com.scanovate_demo.HybridComponent;
> import mabel_tech.com.scanovate_sdk.SdkResultHandler;
> import mabel_tech.com.scanovate_sdk.data.model.ComponentCloseResult;

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

Implement in app/build.gradle:

```
dependencies {
    implementation(files("libs/hybridComponent_3_0_0_17.aar"))

    // Dependencies required by the SDK
    implementation(platform("androidx.compose:compose-bom:2026.02.00"))
    implementation("androidx.compose.ui:ui")
    implementation("androidx.compose.material3:material3:1.5.0-alpha14")
    implementation("androidx.activity:activity-compose:1.12.4")
    implementation("androidx.navigation:navigation-compose:2.7.7")
    implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.8.0")
    implementation("androidx.lifecycle:lifecycle-runtime-compose:2.8.0")
    implementation("androidx.security:security-crypto:1.1.0-alpha06")
    implementation("com.squareup.retrofit2:retrofit:2.9.0")
    implementation("com.squareup.retrofit2:converter-gson:2.9.0")
    implementation("com.squareup.okhttp3:okhttp:4.12.0")
    implementation("com.google.code.gson:gson:2.10.1")
    implementation("com.google.accompanist:accompanist-permissions:0.34.0")
    implementation("com.google.android.gms:play-services-location:21.3.0")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-play-services:1.8.1")
    implementation("com.airbnb.android:lottie-compose:6.4.0")
}
```

> Permissions
> The SDK declares the necessary permissions in its own manifest. These are automatically merged upon compilation. Manual declaration is not required.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

Example Implementation

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

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

> The SDK offers two invocation methods: the new API (ScanovateSDK) and the legacy API (HybridComponent), which maintains full compatibility with version 3.0.0.x. Both offer the same functionality. See the Backward Compatibility section.

> Example
>
> // Example capture method implementation

```
ScanovateSDK.INSTANCE.start(
    this,                                   // Activity
    "1",                                    // documentType (String, número > 0)
    "es",                                   // language ("es" o "en")
    "miProyecto",                           // projectNameSdk
    "mi-api-key",                           // apiKeySdk
    1,                                      // productId (int > 0)
    "https://servidor.com/miProyecto/api/",         // urlSdk
    "https://tracer.com/api/EventTracer/",          // urlTracerBackendService (opcional, "" si no
aplica)
    "",                                     // processId (opcional)
    1,                                      // functionCapture: 1 = Liveness, 2 = CardCapture
    true,                                   // isFrontSide: true = frente, false = reverso
```

```
    "token",                                // token (opcional, "" si no aplica)
    "",                                     // uidDevice (opcional, el SDK genera uno si está vacío)
    new SdkResultHandler() {
        @Override
        public void onSuccess(ComponentCloseResult result) {
            // Captura exitosa
            int statusCode = result.getStatusCode();
            String message = result.getMessage();
            boolean isAlive = result.isAlive();
            String image = result.getImage();           // Base64
            String keyProcess = result.getKeyProcessLiveness();
            String uid = result.getUidDevice();
        }

        @Override
        public void onFailure(ComponentCloseResult result) {
            // Error o cancelación
            int statusCode = result.getStatusCode();
            String message = result.getMessage();
        }
    }
);
```

Parameters Explained

- **language**: Sets the language for the SDK's UI.
- **projectName**: Unique identifier for your project.
- **apiKey**: Authentication key provided by Scanovate.
- **productId**: Identifies the specific Scanovate product/service being used.
- **sdkUrl**: The base URL for making API calls to the Scanovate services.
- **Url_TracerBackendServices**: Url for the event reporting service is not required and is only an extra service. (**Optional**)
- **ImmersiveMode:** Mode to make the component consume all available space while hiding the system UI.
- **Process_ID**: Process identifier to perform the events mapped at the SDK level. (**Optional**)
- **functionCapture**: Specifies the operation mode of the SDK.
- **documentSide**: Determines which side of the document to capture.
- **additionalParameters**: Allows for passing any additional required parameters.

- **completionHandler**: Closure that handles the response or error from the SDK.

Process Transaction Results

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

State Codes Reference

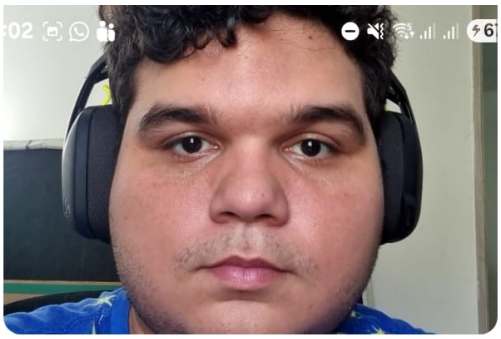
Be aware of the following state codes when processing responses:

- `200`: "SUCCESS"
- `201`: "THE_NUMBER_OF_CONFIGURED_ATTEMPTS_WAS_EXCEEDED_AND_NO_LIFE_WAS_FOUND_IN_THESE"
- `203`: "TIMEOUT"
- `302`: "INTERNAL_ERROR"
- `204`: "CANCELED_PROCED"
- `205`: "PERMISSIONS_DENIED"
- `401`: "TOKEN_ERROR"
- `404`: "INVALID_CREDENTIALS"
- `503`: "CONNECTION_ERROR"

This guide aims to streamline the integration process of the Scanovate Colombia SDK into your Android application, ensuring you can efficiently implement a robust identity validation system.

Demo Application

For a comprehensive example, including full source code demonstrating the integration and usage of the Scanovate Colombia SDK, visit our GitHub repository:



Datos del Proceso

KEY PROCESS LIVENESS
25493d268fea4b94adb3f1a8e5318b88

UID DEVICE
e7012802-0491-474b-8668-960fff2bbace

Información del Dispositivo

LATITUDE
10.9557524

LONGITUDE
-74.7983355

DEVICE MODEL
samsung SM-G998B

DEVICE BRAND
samsung

OS VERSION
Android 15 (API 35)

APP VERSION
3.0.0.17

NETWORK TYPE
WIFI

Resultado

STATUS CODE
200 — SUCCESS

MESSAGE
SUCCESS

IS ALIVE
true

FUNCTION CAPTURE
1 (Liveness)

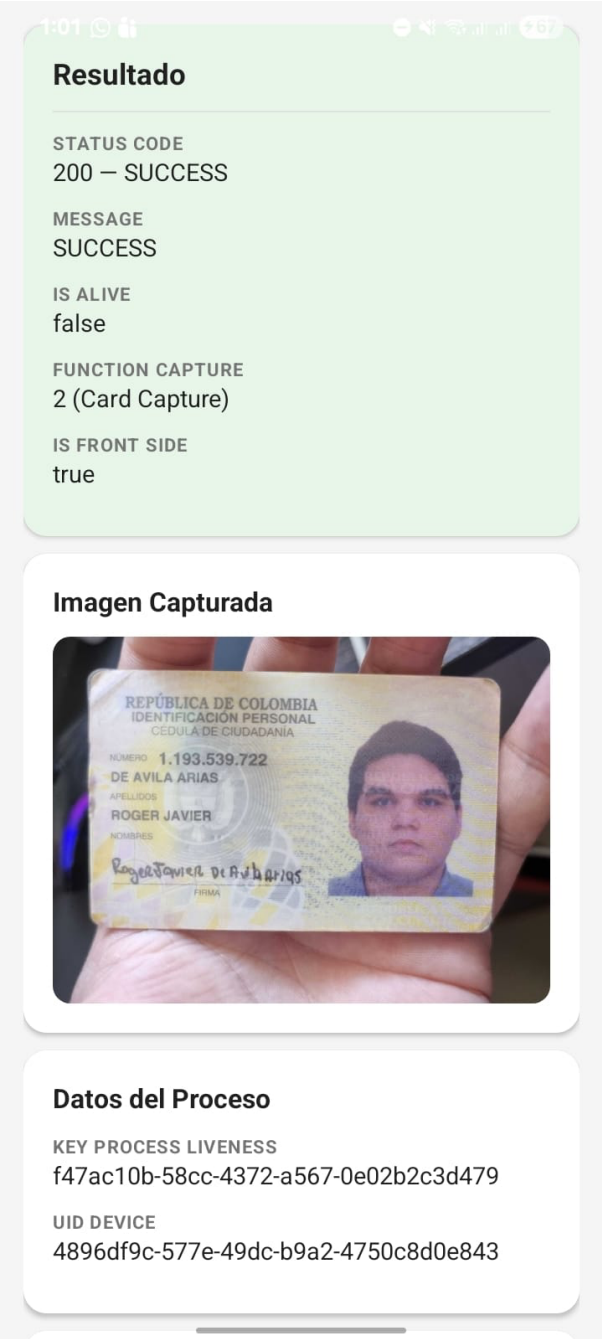
IS FRONT SIDE
false

Imagen Capturada

Datos del Proceso

KEY PROCESS LIVENESS
25493d268fea4b94adb3f1a8e5318b88

UID DEVICE
e7012802-0491-474b-8668-960fff2bbace



Scanovate Colombia SDK Demo App For Android

This demo app provides a hands-on example to help you understand how to integrate and utilize the SDK in your own applications.

iOS SDK Guide

This guide outlines the steps for integrating the SMSDK framework into your iOS application, enabling identity validation processes through facial biometric verification or document scanning.

Installation

1. Add the library

- Download the "SMSDK.xcframework" file.
- In your Xcode project, navigate to the target's general settings.
- Go to the "Frameworks, Libraries, and Embedded Content" section.
- Click the "+" button and add the "SMSDK.xcframework" to your project. Ensure it's set to "Embed & Sign".

2. Import Required Libraries

In the file where you plan to use the SDK, import the necessary libraries:

> swift
>
> ```
> import UIKit
> import AdoComponent
> ```

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

Minimum SDK Version for iOS

Update the minimum iOS version to iOS 11.0:

- Navigate to your target's Build Settings.
- Find the "Deployment" section.
- Set the "iOS Deployment Target" to iOS 11.0 or higher.

Example Implementation

To initiate the SMSDK framework, use the `initWith` method from the `SMManager` class. This method requires a delegate and an `SMParams` object containing the launch parameters. Implement the `SMDelegate` extension to handle the SDK's response.

> Intialization

```
let params = SMParams(productId: "1",
                      projectName: "lulobankqa",
                      apiKey: "db92efc69991",
                      urlSdk: "https://adocolumbia.ado-tech.com/lulobankqa/api/",
                      token: "",
                      function: 1, // 1 for Liveness, 2 for Document Scanning
                      isFrontSide: false, // true for front, false for back of the document
                      uidDevice: "",
                      language: "en") // "en" for English, "es" for Spanish

let smManagerVC = SMManager.initWith(delegate: self, params: params)
smManagerVC.modalPresentationStyle = .fullScreen
present(smManagerVC, animated: true, completion: nil)

// MARK: - SMDelegate
extension ViewController: SMDelegate {
    func completedWithResult(result: Bool, response: ResultsResponse?) {
        dismiss(animated: true) {
            // Handle the SDK response here
        }
    }
}
```

Parameters Explained

- **productId**: Identifier for the product being used.
- **projectName**: Your project identifier provided by the service.
- **apiKey**: Your API key for authentication with the service.
- **urlSdk**: The base URL for the SDK's services.
- **token**: Optional token for additional authentication (if required).
- **function**: Determines the operation mode (e.g., 1 for Liveness, 2 for Document Scanning).
- **isFrontSide**: Indicates which side of the document to capture.
- **uidDevice**: A unique identifier for the device.
- **language**: Specifies the language for the SDK interface.

Resources

Resource files, including animations provided by the client, can be found at the following path within your project:

> SMSDKTest/Resources/Animations

Ensure these resources are correctly integrated into your project for the SDK to function as intended.

State Codes Reference

Be aware of the following state codes when processing responses:

- `200`: "SUCCESS"
- `201`: "THE_NUMBER_OF_CONFIGURED_ATTEMPTS_WAS_EXCEEDED_AND_NO_LIFE_WAS_FOUND_IN_THESE"
- `203`: "TIMEOUT"
- `204`: "CANCELED_PROCED"
- `205`: "PERMISSIONS_DENIED"
- `401`: "TOKEN_ERROR"
- `404`: "INVALID_CREDENTIALS"
- `500`: "CONNECTION_ERROR"

Demo Application

For a comprehensive example, including full source code demonstrating the integration and usage of the Scanovate Colombia SDK, visit our GitHub repository:

Scanovate Colombia SDK Demo App For iOS

This demo app provides a hands-on example to help you understand how to integrate and utilize the SDK in your own applications.

JavaScript SDK Guide

From now on, the ComponentsManager.js file will no longer be loaded locally, and the use of ADO Tech’s official CDN is recommended for better version management, improved performance, and automatic updates.

The import is replaced:

```
<script type="text/javascript" src="Assets/scanovate_card_capture/script.js"></script>
<script type="text/javascript" src="Assets/ComponentsManager.js"></script>
```

with:

```
<script type="text/javascript" src="https://cdn-js.ado-tech.com/latest/ComponentsManager.js"></script>
```

Using latest will ensure that the most recent available version is always used, currently covering versions 1.0 through 3.0.0.1 Playground

IMPORTANT: ADO must be provided with a list of the domains from which the CDN will be consumed so they can be added to the allowlist and the service can be used.

Integrating ADO Technologies' JavaScript SDK into your web application enables you to leverage advanced identity verification features, such as Liveness Detection and Document Capture. This guide provides a structured approach to seamlessly incorporate these functionalities, enhancing the security and user experience of your platform.

Overview

The ADO Technologies JavaScript SDK offers a comprehensive suite of tools designed for real-time identity verification. By integrating this SDK, you can authenticate users by capturing their facial features and identification documents directly within your web application. This process is streamlined and user-friendly, ensuring a high level of accuracy in identity verification.

Requirements

Before starting the integration, ensure you have:

- Access to ADO Technologies' JavaScript SDK url.
- The API key and project name provided by ADO Technologies.

- A clear understanding of the specific features (e.g., Liveness Detection, Document Capture) you wish to implement.

Integration Steps

1. **Include SDK and Assets**: Incorporate the JavaScript SDK and related assets into your web project. This involves linking to the SDK's script files and CSS for styling.
2. **Configure SDK Parameters**: Set up the necessary parameters for the SDK, including the base URL, project name, API key, and product ID. These parameters are crucial for initializing the SDK and ensuring it functions correctly within your application.
3. **Implement User Interface**: Design and implement the user interface through which users will interact with the identity verification features. This includes input fields for configuration parameters and buttons to initiate the capture process.
4. **Capture Process**: Utilize the SDK's functions to capture facial images or documents based on the user's selection. This process should be intuitive, with clear instructions provided to the user.
5. **Handle Responses**: Implement logic to handle the SDK's responses, including success and error callbacks. Display the results appropriately within your application, ensuring users are informed of the outcome.
6. **Testing and Validation**: Thoroughly test the integration to ensure the identity verification process works as expected. Pay special attention to user experience, ensuring the process is smooth and intuitive.

Parameters

To initialize the ADO Technologies JavaScript SDK for identity verification within your web application, you'll need to configure several key parameters. These parameters are essential for tailoring the SDK's functionality to your specific needs and ensuring the verification process operates correctly. Below is an explanation of each parameter required for initialization:

1. **UrlBase**: The base URL of the ADO Technologies service. This URL is the entry point for all SDK requests and should be provided by ADO Technologies. It determines where the SDK sends its verification requests.
2. **ProjectName**: The name of your project as registered with ADO Technologies. This parameter helps the service identify which client is making the request, ensuring that the verification process is correctly attributed and logged.
3. **ApiKey**: A unique key provided by ADO Technologies that authenticates your application's requests. The API key is crucial for securing communication between your application and the ADO Technologies service, preventing unauthorized access.
4. **ProductId**: An identifier for the specific product or service you're using from ADO Technologies. This could relate to different types of verification services offered, such as Liveness Detection or Document Capture.

5. **functionCapture**: Determines the type of capture process to be initiated. This parameter allows you to specify whether you're performing Liveness Detection, Document Capture, or other supported verification processes. The options are typically represented as numerical values or specific strings defined by the SDK.
6. **IsFrontSide**: A boolean parameter indicating whether the document capture (if applicable) should focus on the front side of the identification document. This is relevant for services that require document images as part of the verification process.
7. **UidDevice**: A unique identifier for the device being used to perform the verification. This can be useful for logging, analytics, and ensuring that verification attempts are uniquely associated with a specific device.
8. **Token**: An optional parameter that may be required for additional authentication or session management purposes. If your verification process involves multiple steps or requires maintaining a session state, this token can be used to manage that state across requests.
9. **ProcessId**: An identifier for the specific verification process instance. This can be used to track the progress of a verification attempt or to retrieve results after the process has been completed (How to generate the process Id).

These parameters are typically set by assigning values to the corresponding input fields or variables within your web application's frontend code. Once configured, these parameters are passed to the SDK's initialization function, which prepares the SDK for the capture and verification process based on the provided configuration.

It's important to handle these parameters securely, especially those that could be sensitive, such as the `ApiKey` and `Token`. Ensure that your application's frontend and backend architecture support secure transmission and storage of these values.

Example Implementation

Below is an example HTML structure demonstrating how to set up the SDK in your web application. This example includes the SDK and asset links, configuration inputs, and the capture initiation button.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=0, minimal-ui">
    <title>Demo ADO Components</title>
    <script type="text/javascript" src="https://cdn-js.ado-tech.com/latest/ComponentsManager.js"></script>
```

```
    <link rel="stylesheet"
href="Assets/scanovate_card_capture/assets/main.css">
    <link rel="stylesheet"
href="Assets/scanovate_card_capture/assets/loader.css">
</head>
<body>
    <!-- Configuration and Capture UI omitted for brevity -->

    <script>
        function InitCapture() {
            // Capture initialization logic and callbacks
        }
    </script>
</body>
</html>
```

This structure is a starting point for integrating the SDK. Customize the configuration and UI according to your application's needs and the specific features you plan to use.

By following this guide, you can effectively integrate ADO Technologies' JavaScript SDK into your web application, enabling robust identity verification functionalities that enhance the security and user experience of your platform.

Liveness API Documentation

Introduction

The Liveness API provides access to biometric liveness detection results and reporting capabilities. This documentation focuses on two key endpoints: retrieving liveness results and generating reports.

Important Contact Information: For information about this API and other solutions in our catalog, please contact our financial area for evaluation at julian@ado-tech.com. All access keys, endpoint URLs, and other access elements will only be provided after reaching a formal agreement between both entities.

Important Note

The liveness detection process requires integration with components from https://docs.ado-tech.com/books/b-trust/chapter/sdks. These components have associated costs and service agreements that must be discussed with the finance department before implementation.

API Endpoints

1. Get Results

Retrieves the results of a previously executed liveness verification process.

Endpoint: `POST {base_url}/api/images/getResults`

Request Body:

```
{
   "idTransaction": "process_id",
   "user": "your_username",
   "password": "your_password",
   "apiKey": "your_api_key",
```

```
    "transactionNumber": "process_id"
}
```

Response: The API returns detailed liveness verification results, including:

```
{
    "referenceNumber": "a7112314-f8c6-40b9-a5de-ab91fa98e3bc",
    "score": 0.8818287,
    "quality": 0.8818287,
    "probability": 0.9878632,
    "threshold_probabillity": 0.6,
    "threshold_quality": 0.5,
    "isAlive": true,
    "isFraud": false,
    "image": "/ ",
    "videoBase64": " "
}
```

Key Response Fields:

- `referenceNumber`: Unique identifier for the verification result
- `score`: Overall liveness score
- `quality`: Image quality score
- `probability`: Probability that the subject is alive
- `threshold_probabillity`: Minimum probability threshold for positive verification
- `threshold_quality`: Minimum quality threshold for acceptable images
- `isAlive`: Boolean indicating liveness detection result
- `isFraud`: Boolean indicating potential fraud detection
- `image`: Base64 encoded image (abbreviated in example)
- `videoBase64`: Base64 encoded video if applicable (abbreviated in example)

Note: The key process liveness ID required for this endpoint is obtained from the B-Trust SDK components. Access to these components requires proper licensing and authorization.

2. Generate Report

Generates a comprehensive report of liveness verifications for a specific project and date range.

Endpoint: `POST {base_url}/api/images/Report`

Request Body:

```
{
   "IntialDate": "2024-01-01T00:00:00.000Z",
   "EndDate": "2024-02-01T23:59:59.999Z",
   "projectId": "your_project_id"
}
```

Important Considerations:

- It is recommended to request reports spanning only 1-2 months at a time for optimal performance
- The `projectId` must match the assigned project identifier from your service agreement
- Date formats must follow ISO 8601 standard (YYYY-MM-DDThh:mm:ss.sssZ)

Response: The API will return a comprehensive report of liveness verification transactions within the specified date range for the given project.

Additional Services

For more advanced biometric verification needs, the following solutions are available:

- **Compare Face**: Validates and compares facial images
- **Validar rostro-persona**: Verifies that a face belongs to a specific person
- **Validar rostro-documento**: Validates a face against identification document photos

These additional services can be integrated with the liveness detection process to create a complete identity verification solution. Each component returns useful data for integration with the liveness verification workflow.

Service Acquisition

Our catalog contains numerous additional routines and services for biometric verification and identity validation. For more information about all available services, pricing, and implementation:

- Please contact our financial area at: julian@ado-tech.com
- All access elements including API keys, endpoint URLs, and credentials will only be provided after a formal agreement is reached between both entities
- Integration support is available after service contracts are finalized

Integration Considerations

- Proper error handling is essential for all API calls
- Credentials must be securely stored and transmitted
- Results should be evaluated against your specific security threshold requirements
- Integration with the B-Trust SDK components requires proper licensing and configuration

By leveraging these endpoints, you can access liveness verification results and generate comprehensive reports for your biometric verification processes.

SDK response update 3.1.0.0

As part of the SDK update, starting now we will generate standardized response objects, including the full detail of fields and their possible values.

Additionally, we improved the tagging logic to correctly detect icons and related events. With these improvements, when a spoofing-related event is detected, a transaction will be generated and the event will be recorded with its corresponding details.

If spoofing is detected, an evidence image will also be sent to the customer-provided **webhook**. Moreover, from now on the transaction details (including the transaction number) will be delivered through the webhook to ensure an auditable record of the event.

> **Expected response:** Transaction details sent to the webhook.

> **SDK response level (PSA):** This information will be returned by the SDK as part of the standardized response object.


```
{
  "processType": "liveness",
  "statusCode": 1,
  "detail": {
    "status": "approved",
    "reason": {
      "name": "NOT_ALERT",
      "description": "The transaction was approved."
    },
    "uidDevice": "1da62007d8974a9fa71e18cba76f6fc1",
    "keyProcessLiveness": "69c265aeca9d4863b61b1fc6a8099f52",
    "processId": "3f2b9c1e-7a4d-4f2a-9c0c-6c4c2e9b8d12",
    "image": {
      "encoding": "base64",
      "mimeType": "image/jpeg",
      "data": "<BASE64>"
    }
  }
}
```

Field description

- **processType**: Type of process executed.
  - Possible values: `"liveness" | "cardcapture"`
  - Notes:
    - If `processType = "liveness"`, the field `keyProcessLiveness` **may be present**.
    - If `processType = "cardcapture"`, `keyProcessLiveness` **must NOT be returned**.
- **Side**: Indicates which side of the document was captured**.**
  - Possible values: `"front" | "back"`
  - Notes:
    - If `processType = "liveness"`, the field **Side must NOT be returned**.
    - If `processType = "cardcapture"`, **Side may be present**.
- **statusCode**: Numeric result code of the process.
  - `1` = approved
  - `2` = rejected
  - `3` = failed
- **detail**: Object containing the detailed outcome.
  - **status**: Text status of the result.
    - Possible values: `"approved" | "rejected" | "failed"`
  - **reason**: Human-readable reason for the result.
    - **name**: Short label describing the reason (e.g., `"No alerts"`).
    - **description**: Longer explanation of the reason (e.g., `"The transaction was approved."`).
  - **uidDevice**: Unique device identifier (string).
  - **keyProcessLiveness** *(only for liveness)*: Unique identifier for the liveness process (string).
    - Returned **only** when `processType = "liveness"`.
  - **processId**: Unique process identifier (GUID/UUID) generated per execution to correlate SDK responses and webhook events.
  - **image**: Image object associated with the process.
    - **encoding**: Encoding type used for the image payload.
      - Typical value: `"base64"`
    - **mimeType**: MIME type of the image data.
      - Examples: `"image/jpeg"`, `"image/png"`
    - **data**: Base64-encoded image content (string).
      - Usually provided **without** the `` prefix.

The component's possible responses are split by functionality: CardCapture (document capture) or **Liveness** (proof of life). Below is a practical, complete list of outcomes you should handle (success, pending, failures, cancellations, and technical errors).

Liveness

Status (code)	Nombre (name)	Descripción (description)	Objeto (reason)
NONE	NOT_ALERT	The transaction was approved with no alerts detected.	{"code":"NONE","name":"NOT_ALERT","description":"The transaction was approved with no alerts detected."}
NOT_ALIVE	Not alive detected	The liveness check indicates the subject is not alive.	{"code":"NOT_ALIVE","name":"Not alive detected","description":"The liveness check indicates the subject is not alive."}
CLOSED_EYES	Eyes closed detected	The subject's eyes were detected as closed during the capture.	{"code":"CLOSED_EYES","name":"Eyes closed detected","description":"The subject's eyes were detected as closed during the capture."}
SPOOFING	Spoofing detected	A potential spoofing attempt was detected during the liveness process.	{"code":"SPOOFING","name":"Spoofing detected","description":"A potential spoofing attempt was detected during the liveness process."}
PAD	Presentation attack detected (PAD)	A potential presentation attack was detected (PAD confidence triggered).	{"code":"PAD","name":"Presentation attack detected (PAD)","description":"A potential presentation attack was detected (PAD confidence triggered)."}
PERMISSION_DENIED	Camera permission denied	Camera access was denied by the user or the operating system.	{"code":"PERMISSION_DENIED","name":"Camera permission denied","description":"Camera access was denied by the user or the operating system."}
CAMERA_NOT_FOUND	Camera not found	No camera was detected on the device, or it is unavailable.	{"code":"CAMERA_NOT_FOUND","name":"Camera not found","description":"No camera was detected on the device, or it is unavailable."}
FACE_NOT_DETECTED	Face not detected	No face was detected in the camera frame.	{"code":"FACE_NOT_DETECTED","name":"Face not detected","description":"No face was detected in the camera frame."}
NETWORK_ERROR	Network error	A network issue prevented the process from completing.	{"code":"NETWORK_ERROR","name":"Network error","description":"A network issue prevented the process from completing."}
SDK_ERROR	SDK error	An internal SDK error occurred during the process.	{"code":"SDK_ERROR","name":"SDK error","description":"An internal SDK error occurred during the process."}
USER_CANCELED	User canceled	The user canceled the process before completion.	{"code":"USER_CANCELED","name":"User canceled","description":"The user canceled the process before completion."}

Example:

*As an example response, the JSON below represents a **rejected transaction (Status 2)**.*
*The `reason` field may come as an **object**, including the **rejection reason details** and its **description**.*
It also includes identifiers such as `uidDevice` and `keyProcessLiveness`, plus `processId` (the unique process UID).
*Evidence (e.g., an image) can be returned as **Base64**, including the **encoding type**.*


```
{
  "processType": "liveness",
  "statusCode": 2,
  "detail": {
    "status": "rejected",
    "reason": {
      "name": "SPOOFING",
      "description": "The liveness check indicates the subject is not alive."
    },
    "uidDevice": "1da62007d8974a9fa71e18cba76f6fc1",
    "keyProcessLiveness": "69c265aeca9d4863b61b1fc6a8099f52",
    "processId": "3f2b9c1e-7a4d-4f2a-9c0c-6c4c2e9b8d12",
    "image": {
      "encoding": "base64",
      "mimeType": "image/jpeg",
      "data": "<BASE64>"
    }
  }
}
```


CardCapture

Status (code)	Nombre (name)	Descripción (description)	Objeto (reason)
`NONE`	`NOT_ALERT`	The transaction was approved with no alerts detected.	`{"code":"NONE","name":"NOT_ALERT","description":"The transaction was approved with no alerts detected."}`
`PERMISSION_DENIED`	`Camera permission denied`	Camera access was denied by the user or the operating system.	`{"code":"PERMISSION_DENIED","name":"Camera permission denied","description":"Camera access was denied by the user or the operating system."}`

Status (code)	Nombre (name)	Descripción (description)	Objeto (reason)
`CAMERA_NOT_FOUND`	`Camera not found`	No camera was detected on the device, or it is unavailable.	`{"code":"CAMERA_NOT_FOUND","name":"Camera not found","description":"No camera was detected on the device, or it is unavailable."}`
`CAMERA_IN_USE`	`Camera in use`	The camera is currently being used by another application.	`{"code":"CAMERA_IN_USE","name":"Camera in use","description":"The camera is currently being used by another application."}`
`NOT_SUPPORTED`	`Device not supported`	The device does not meet the requirements for this capture process.	`{"code":"NOT_SUPPORTED","name":"Device not supported","description":"The device does not meet the requirements for this capture process."}`
`DOCUMENT_NOT_DETECTED`	`Document not detected`	No document was detected in the camera frame.	`{"code":"DOCUMENT_NOT_DETECTED","name":"Document not detected","description":"No document was detected in the camera frame."}`
`NETWORK_ERROR`	`Network error`	A network issue prevented the process from completing.	`{"code":"NETWORK_ERROR","name":"Network error","description":"A network issue prevented the process from completing."}`
`SDK_ERROR`	`SDK error`	An internal SDK error occurred during the process.	`{"code":"SDK_ERROR","name":"SDK error","description":"An internal SDK error occurred during the process."}`
`USER_CANCELED`	`User canceled`	The user canceled the process before completion.	`{"code":"USER_CANCELED","name":"User canceled","description":"The user canceled the process before completion."}`
`FOCUS_LOST`	`Focus lost / component interrupted`	The process was interrupted because the app lost focus or the component was closed, minimized, or moved to the background.	`{"code":"FOCUS_LOST","name":"Focus lost / component interrupted","description":"The process was interrupted because the app lost focus or the component was closed, minimized, or moved to the background."}`

Example:

For the **document capture** process, it will be mapped under `processType: "cardCapture"` and will use `statusCode: 3`, which groups any **cancellation scenario** that affects the execution of the flow, for example:

- **Process cancelled by the user.**
- **Component focus loss during the session** (e.g., the app goes to the background or the user navigates to another screen).

The response will include the `reason` (cancellation reason) and `status` (state), as well as the `processId` (unique process identifier) and its **linked ID** (if applicable). Additionally, the `imageDetail` object will be returned, where the result of the processed image will be provided.

```
{
  "processType": "cardcapture",
```

```
  "side":"front",
  "statusCode": 3,
  "detail": {
    "status": "failed",
    "reason": {
      "name": "USER_CANCELED",
      "description": "The user canceled the process before completion."
    },
    "uidDevice": "1da62007d8974a9fa71e18cba76f6fc1",
    "processId": "3f2b9c1e-7a4d-4f2a-9c0c-6c4c2e9b8d12",
    "image": {
      "encoding": "base64",
      "mimeType": "image/jpeg",
      "data": "<BASE64>"
    }
  }
}
```


This process will trigger an **event to the webhook**, where a record will be **registered** and linked by a **UID** (for example, `processId` as the unique process identifier). The relationship between the **transaction** and the **webhook event** will be kept for traceability.

> The payload sent will follow the structure below. This response **only applies to the webhook** when `reasonCode` is one of the following values: **`SPOOFING`**, **`PAD`**, **`CLOSED_EYES`**, **`NOT_ALIVE`**.


```
{
   "transactionId": "10743",
   "processId": "3f2b9c1e-7a4d-4f2a-9c0c-6c4c2e9b8d12",
   "transactionStatus": "rejected",
   "stages": [
      {
         "processType": "liveness",
         "status": “Spoofing”,
         "reason": {
            "name": "SPOOFING",
            "description": "A potential spoofing attempt was detected during the liveness process."
         },
```

```
        "uidDevice": "1da62007d8974a9fa71e18cba76f6fc1",
        "keyProcessLiveness": "69c265aeca9d4863b61b1fc6a8099f52",
        "processId": "3f2b9c1e-7a4d-4f2a-9c0c-6c4c2e9b8d12"
      }
   ]
}
```


webhook-risk-validation.yaml