KYC Ecuador StartCompareFaces

Identity Validation Flow Integration Guide for Ecuador StarCompareFaces Routine

This guide offers a detailed approach to integrating a specialized identity validation flow tailored for Ecuadorian users. This process stands out by authenticating users through real-time validation of their facial features, comparing them against the official data provided by the Civilian Registry of Ecuador. By adhering to a proven framework used in classic verification flows, this integration is adapted to meet the unique requirements of users from Ecuador, ensuring a secure and efficient verification process.

Overview

The identity validation flow for Ecuador leverages advanced facial recognition technology to compare a user's live-captured photograph against identity data from the Civilian Registry of Ecuador. This comparison ensures that the person attempting to verify their identity matches the official records, thereby enhancing security and trust in digital platforms.

Key Steps for Integration

1. **User Consent and Instruction**: Begin by informing users about the process and obtaining their consent. Clearly explain the need for a facial photograph and how it will be used for verification purposes. Ensure users understand the importance of clear lighting and a neutral background for the photograph.
2. **Capture and Submission**: Implement a user-friendly interface that guides users through the photograph capture process. This interface should include real-time feedback to help users position their face correctly within the designated area. Once the photograph is

captured, it, along with any necessary identification information (e.g., unique identification number), is submitted for verification.
3. **Real-Time Verification**: Upon submission, the system processes the photograph and identification information, comparing them against the data provided by the Civilian Registry of Ecuador. This step utilizes facial recognition algorithms to ensure a match between the live-captured photograph and the official records.
4. **Verification Outcome**: The result of the verification process is communicated back to the user and the platform in real-time. A successful verification confirms the user's identity matches the official records, while any discrepancies are flagged for further review.

Implementation Considerations

- **Privacy and Data Protection**: Ensure the process complies with local and international data protection regulations. User data, especially biometric information, should be handled with the utmost care, ensuring privacy and security.
- **User Experience**: Design the verification process to be as intuitive and straightforward as possible. Minimize user effort and provide clear instructions and feedback throughout the process.
- **Technical Integration**: Depending on your platform's architecture, choose the appropriate method (GET or POST) for submitting the verification request. Ensure your system is capable of handling the response, whether it's a direct callback or a JSON object containing the verification outcome.
- **Testing and Quality Assurance**: Before launching the integration, conduct thorough testing to ensure accuracy in the verification process and a smooth user experience. Consider various user scenarios and edge cases to refine the process.

By following this guide, you can integrate a robust and efficient identity validation flow into your platform, specifically designed for Ecuadorian users. This process not only enhances security by leveraging real-time data from the Civilian Registry of Ecuador but also offers a seamless and user-friendly experience, building trust and confidence among your user base.

Step 1: Preparing for Integration

Before initiating the integration process, ensure you have the following:

- Access to the base URL for the identity verification service.
- An API key and project name provided by the service provider.
- Understanding of the specific parameters required for the Ecuadorian identity validation flow.

CURL Command Structure

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

For the facial validation of the StarCompare Faces routine, we use the StarCompare Faces service for creating the UID. This service will request the customer's photograph for validation, extracted from the Ecuadorian registry, along with data such as fingerprint code, NUIP, Documentype (3 for Ecuadorian ID), full name, and digital signature photograph in case we are the ones making the call to the Ecuadorian civil registry. We need that in the request, only the document number and fingerprint code are provided. If no photo is sent, our system will make a call to the civil registry to extract this information from the data obtained.

```
curl --location
'{URL_Base}/api/Integration/{ProjectName}/Validation/StartCompareFaces' \
--header 'apiKey: your_api_key' \
--header 'projectName: your_project_name' \
--header 'Content-Type: application/json' \
--data '{
  "ProductId": your_productid,
  "CustomerServicePhoto": base64 photo by ecuador registry,
  "SignaturePhoto": base64 photo signature for ecuador registry,
  "DactilarCode": customer's fingerprint code,
  "IdentificationNumber": customer document number,
  "Name": client's full name,
  "DocumentType": type of document (3 For Ecuadorian cedula)
}'
```

Parameters Explained

- **{URL_Base}**: The base URL of the identity verification service. This should be replaced with the actual URL provided to you.
- **{ProjectName}**: The name of your project as registered with the identity verification service. Replace `{ProjectName}` with your specific project name.

Code Response Description

200: "UID" JSON formatted object with transaction information.

400: The provided data does not correspond to the expected criteria.

401: Authorization process was unsuccessful. Validate the project code and/or API Key.

404: The specified product code and/or project does not exist.

500: An error has occurred, validate the delivered ID number for more details.

Step 2: Constructing the Request

The identity validation process can be initiated by using the GET methods .

For the GET Method:

Construct a URL with the required parameters appended as query strings. The basic structure is as follows:

> URL_Base/compare-faces?callback=https://www.google.com/&uid=UID

Parameters:

- `callback`: The URL to which the user will be redirected after the verification process is completed.
- `uid`: unique identifier, assigned to each user for facial validation of the transaction created to be validated.

Step 3: Handling the User Experience

1. **User Consent**: Inform the user about the minimum conditions required for capturing the facial photograph with Liveness detection. The browser will request permission to access the device's camera and location.
2. **Capture Process**: After granting permission, the user will be prompted to capture their photograph by clicking on "capturar fotografía". They must keep their face within the on-screen oval until the internal clock completes.
3. **Data Entry**: On the Identification Data screen, users must enter their unique identification number and individual fingerprint code to proceed with the identity validation by pressing "Continuar".
4. **Completion**: Upon completion, users will see a summary screen indicating that the transaction has finished successfully.

Step 4: Receiving the Response

After the user completes the process, your application will receive a JSON object at the specified callback URL. The JSON structure includes the transaction's outcome and relevant data, such as the `id`, `codeId`, and `ThresHoldCompareFaces`.

Step 5: Retrieving Transaction Results

The Validation method is a crucial part of the identity verification process, allowing you to retrieve detailed information about the transaction and the outcome of the validation. This method is particularly useful for post-verification steps, such as auditing, compliance checks, or further user verification processes. Below, we detail how to use the Validation method with a `curl` command, which is designed to fetch the results of a specific transaction using a GET request.

Overview

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

CURL Command Structure

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

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

Parameters Explained

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

Headers

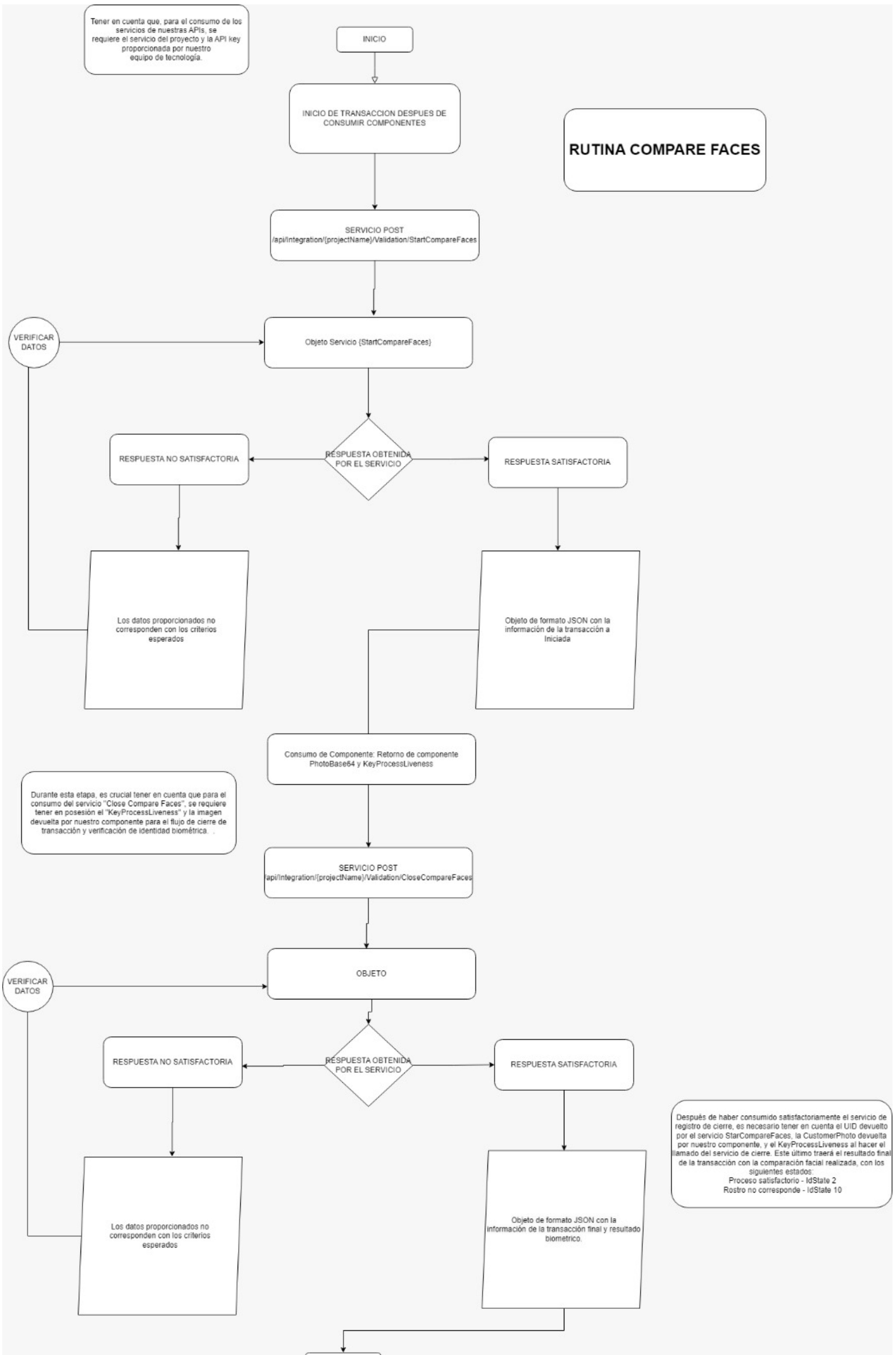
- **accept**: Indicates the expected media type of the response, which is `application/json` for JSON-formatted data.
- **apiKey**: Your API key for authentication with the identity verification service. Replace `your_api_key` with the actual API key assigned to your project.
- **returnDocuments**: A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

Usage Tips

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

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

Routine Flow Chart


RUTINA COMPARE FACES
INICIO
INICIO DE TRANSACCION DESPUES DE CONSUMIR COMPONENTES
SERVICIO POST /api/Integration/{projectName}/Validation/StartCompareFaces
VERIFICAR DATOS
Objeto Servicio {StartCompareFaces}
RESPUESTA OBTENIDA POR EL SERVICIO
RESPUESTA NO SATISFACTORIA
RESPUESTA SATISFACTORIA
Los datos proporcionados no corresponden con los criterios esperados
Objeto de formato JSON con la información de la transacción a Iniciada
Consumo de Componente: Retorno de componente PhotoBase64 y KeyProcessLiveness
SERVICIO POST /api/Integration/{projectName}/Validation/CloseCompareFaces
VERIFICAR DATOS
OBJETO
RESPUESTA OBTENIDA POR EL SERVICIO
RESPUESTA NO SATISFACTORIA
RESPUESTA SATISFACTORIA
Los datos proporcionados no corresponden con los criterios esperados
Objeto de formato JSON con la información de la transacción final y resultado biometrico.
FIN

Tener en cuenta que, para el consumo de los servicios de nuestras APIs, se requiere el servicio del proyecto y la API key proporcionada por nuestro equipo de tecnología.

Durante esta etapa, es crucial tener en cuenta que para el consumo del servicio "Close Compare Faces", se requiere tener en posesión el "KeyProcessLiveness" y la imagen devuelta por nuestro componente para el flujo de cierre de transacción y verificación de identidad biométrica. .

Después de haber consumido satisfactoriamente el servicio de registro de cierre, es necesario tener en cuenta el UID devuelto por el servicio StarCompareFaces, la CustomerPhoto devuelta por nuestro componente, y el KeyProcessLiveness al hacer el llamado del servicio de cierre. Este último traerá el resultado final de la transacción con la comparación facial realizada, con los siguientes estados:
Proceso satisfactorio - IdState 2
Rostro no corresponde - IdState 10

Revision #11
Created 23 May 2024 19:58:18 by Admin
Updated 29 May 2024 14:25:09 by Admin