

# Single-use link

## Introduction

This document provides comprehensive guidance for integrating with the B-Trust identity verification service. The service enables secure identity verification through a combination of document authentication and facial recognition.

## Requirements and Compatibility

Before proceeding with integration, please ensure you have the following resources and knowledge:

- Access to the base URL for the identity verification service
- API key and project name provided by the service provider
- The product ID associated with the service you intend to utilize
- Working knowledge of HTTP GET and POST methods
- Authentication credentials for web services
- Endpoint for callback registration and webhook configuration
- Development environment capable of handling REST API calls
- Understanding of JSON request and response structures

## Authentication

### Login Service

To access the B-Trust API services, you must first authenticate using the login endpoint. This will provide the access token required for all subsequent requests.

**Endpoint:** `https://api-fintechcart.ado-tech.com/api/v1/auth/login`

**Method:** POST

**Headers:**

x-accountId: AdoQa

Content-Type: application/json

### Request Body:

```
{
  "username": "your-username@example.com",
  "password": "your-password"
}
```

### Example Response:

```
{
  "access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXLTJpRcWVjdHdZOGtDN05VdFZsTzBUSlJaTzhsOFRkRkZQ
SXZzcmJzIn0...",
  "expires_in": 3600,
  "refresh_expires_in": 0,
  "token_type": "Bearer",
  "not-before-policy": 0,
  "scope": "email profile"
}
```

The `access_token` obtained from this response must be included in the Authorization header for all subsequent API requests, using the format `Bearer {access_token}`.

# Identity Verification Flow Services

## Create Flow Request

This endpoint allows you to create a new identity verification request, initiating the verification flow process.

**Endpoint:** `https://api-fintechart.ado-tech.com/api/v1/flowmanager/flowrequest/create`

**Method:** POST

**Headers:**

Authorization: Bearer {access\_token}

x-accountid: AdoQa

Content-Type: application/json

### Request Body Parameters:

Parameter	Type	Description
documentType	String	Type of identification document (e.g., "1" for national ID)
documentNumber	String	The identification number on the document
flowType	String	The type of verification flow to initiate (e.g., "1" for enrollment)
riskAmount	Number	The monetary value associated with the transaction for risk assessment
callbackUrl	String	URL where the user will be redirected after verification

### Example Request Body:

```
{
  "documentType": "1",
  "documentNumber": "1001818723",
  "flowType": "1",
  "riskAmount": 1230000,
  "callbackUrl": "https://chat.openai.com/"
}
```

### Example Response:

```
{
  "code": 6871,
  "typeDocument": 1,
  "document": "1001818723",
  "url": "https://kyc-qa.ado-tech.com/AdoQa/f7fb4984a8a347699e1c72cc5",
  "key": "f7fb4984a8a347699e1c72cc5",
  "flowType": "1",
  "state": 1,
  "createFor": "oscar.castañeda@ado-tech.com",
  "updateFor": "oscar.castañeda@ado-tech.com",
  "valiteKey": "2025-05-09T09:23:19.0795159Z",
}
```

```
"amountRisk": 1230000,
"customerId": 2,
"callbackUrl": "https://chat.openai.com/",
"createDate": "2025-05-08T09:18:19.0795885Z",
"project": 142,
"customer": {
  "code": 2,
  "idAccount": "AdoQa",
  "urlAdo": "https://adocolombia-qa.ado-tech.com/ADODemo",
  "apiKey": "db92efc69991",
  "projectNameAdo": "ADODemo",
  "urlClientFlow": "https://kyc-qa.ado-tech.com/AdoQa",
  "adoProduct": 1,
  "adoRiskId": 1,
  "styleLogo": "https://scanovate.com/wp-content/uploads/2019/07/scanovate_logo.gif",
  "styleColorPrimary": "#2851e6",
  "styleColorSecondary": "#000",
  "styleBackgroundColorBody": "#fff",
  "styleBackgroundColorContainer": "#fff",
  "styleBackgorundColorPrimaryButton": "#0076ff",
  "styleColorPrimaryTextButton": "#fff",
  "styleBackgroundColorSecondaryButton": "#ecee0",
  "styleColorSecondaryTextButton": "#8593a2"
}
```

## Response Fields:

Field	Description
code	Internal reference code for the request
typeDocument	Type of identification document
document	The identification number
url	The URL to redirect the user for verification
key	Unique key for this verification request
flowType	Type of verification flow
state	Current state of the request (1 = created)
createFor	Email of user who created the request
updateFor	Email of user who last updated the request

Field	Description
valiteKey	Expiration datetime of the verification key
amountRisk	Monetary value for risk assessment
customerId	Customer ID in the system
callBackUrl	URL where user will be redirected after verification
createDate	Creation datetime of the request
project	Project ID in the system
customer	Object containing customer configuration details

# Retrieve Flow Request

This endpoint allows you to retrieve information about an existing verification request.

**Endpoint:** `https://api-fintechart.ado-tech.com/api/v1/flowmanager/flowrequest/byId`

**Method:** GET

**Headers:**

```
Authorization: Bearer {access_token}
x-accountid: AdoQa
```

**Query Parameters:**

Parameter	Description
key	The unique key of the verification request

**Example Request:**

```
GET https://api-fintechart.ado-tech.com/api/v1/flowmanager/flowrequest/byId?key=b74bfc9040924f06a419dacc2
```

**Example Response:**

```
{
  "success": true,
  "message": "get successfull",
  "flowRequestData": {
    "documentType": 1,
```

```
"documentNumber": "1234097206",
"flowUrl": "https://kyc-qa.ado-tech.com/AdoQa",
"flowKey": "b74bfc9040924f06a419dacc2",
"flowType": "1",
"state": "created",
"createdBy": "oscar.castañeda@ado-tech.com",
"updateBy": "oscar.castañeda@ado-tech.com",
"createDate": "2025-02-18T10:05:45.131812Z",
"riskAmount": 1230000,
"customerId": 2,
"callbackUrl": "https://chat.openai.com/"
}
}
```

## Response Fields:

Field	Description
success	Boolean indicating if the request was successful
message	Message describing the result of the operation
flowRequestData	Object containing the verification request data
documentType	Type of identification document
documentNumber	The identification number on the document
flowUrl	Base URL for the verification flow
flowKey	Unique key for this verification request
flowType	Type of verification flow
state	Current state of the request
createdBy	Email of user who created the request
updateBy	Email of user who last updated the request
createDate	Creation datetime of the request
riskAmount	Monetary value for risk assessment
customerId	Customer ID in the system
callbackUrl	URL where user will be redirected after verification

# Webhook Integration

Webhooks allow your system to receive real-time notifications when a verification process is completed. This section details how to set up and handle webhook callbacks.

# Webhook Authentication

Before receiving webhook notifications, you must authenticate to obtain a token.

**Endpoint:** {example\_host}/auth/realms/{example\_realm}/protocol/openid-connect/token

**Method:** POST

### Headers:

Content-Type: application/x-www-form-urlencoded

### Request Body Parameters (form-urlencoded):

Parameter	Description
client_id	Your client ID for webhook authentication
client_secret	Your client secret for webhook authentication
grant_type	Authentication method (use "client_credentials")

### Example Request (CURL):

```
curl -X POST \  
  '{example_host}/auth/realms/{example_realm}/protocol/openid-connect/token' \  
  -H 'Content-Type: application/x-www-form-urlencoded' \  
  -d 'client_id={example_client}&client_secret={example_secret}&grant_type=client_credentials'
```

**Example Response:**

```
{
  "access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUiiwia2kiOiA6IChzTjZFTlprcWVjdHdZOgtdN05VdFZsTzBUSlJaTzhsOFRkRkZQ
SXZzcmJzIn0...",
  "expires_in": 299,
  "refresh_expires_in": 0,
  "token_type": "Bearer",
  "not-before-policy": 0,
  "scope": "email profile"
}
```

# Receiving Verification Process Data

Your webhook endpoint should be prepared to receive notifications when a verification process is completed.

**Webhook Endpoint:** `{example_host}/{example_data_call_back}`

**Method:** POST

**Headers:**

```
Authorization: Bearer {access_token}
Content-Type: application/json
```

**Example Webhook Payload:**

```
{
  "Uid": "b2b731bc-785c-410e-80a4-27cb82215956",
  "key": "c511dd3154264283aa226fbe9",
  "StartingDate": "2023-09-07T10:55:26.603",
  "CreationDate": "2023-09-07T10:55:47.99",
  "CreationIP": "186.82.84.1",
  "DocumentType": 1,
  "IdNumber": "1001818723",
  "FirstName": "CARLOS",
  "SecondName": "HABID",
  "FirstSurname": "VERGEL",
  "SecondSurname": "BARRAZA",
  "Gender": "M",
  "BirthDate": "2002-08-30T00:00:00",
  "PlaceBirth": "BARRANQUILLA (ATLANTICO)",
  "ExpeditionCity": null,
  "ExpeditionDepartment": null,
  "BirthCity": null,
  "BirthDepartment": null,
  "TransactionType": 1,
  "TransactionTypeName": "Enroll",
  "IssueDate": "2020-09-03T00:00:00",
  "TransactionId": "125",
  "ProductId": "1",
  "ComparationFacesSuccessful": false,
```



```
"FaceFound": false,
"FaceDocumentFrontFound": false,
"BarcodeFound": false,
"ResultComparationFaces": 0.0,
"ComparationFacesAproved": false,
"Extras": {
  "IdState": "4",
  "StateName": "Documento auténtico, sin cotejo facial"
},
"NumberPhone": null,
"CodFingerprint": null,
"ResultQRCode": null,
"DactilarCode": null,
"ReponseControlList": null,
"Images": [],
"SignedDocuments": [],
"Scores": [
  {
    "Id": 4,
    "UserName": null,
    "StateName": "Documento auténtico, sin cotejo facial",
    "StartingDate": "0001-01-01T00:00:00",
    "Observation": null
  }
],
"Response_ANI": null,
"Parameters": null
}
```

### Webhook Response:

Your webhook endpoint should respond with a 200 OK status to acknowledge receipt of the data. You may include additional information in your response as needed.

### Webhook Payload Fields:

Field	Description
Uid	Unique identifier for this verification process
key	Key that matches the flow request key
StartingDate	Date and time when the verification process started

Field	Description
CreationDate	Date and time when the verification record was created
CreationIP	IP address from which the verification was initiated
DocumentType	Type of identification document
IdNumber	Identification number from the document
FirstName	First name of the verified individual
SecondName	Second name of the verified individual
FirstSurname	First surname/last name of the verified individual
SecondSurname	Second surname/last name of the verified individual
Gender	Gender of the verified individual
BirthDate	Date of birth of the verified individual
PlaceBirth	Place of birth of the verified individual
TransactionType	Type of transaction (1 = Enroll)
TransactionTypeName	Name of the transaction type
IssueDate	Date when the identification document was issued
TransactionId	Unique identifier for the transaction
ProductId	Identifier of the product used for verification
ComparisonFacesSuccessful	Boolean indicating if facial comparison was successful
FaceFound	Boolean indicating if a face was detected
FaceDocumentFrontFound	Boolean indicating if a face was found on the front of the document
BarcodeFound	Boolean indicating if a barcode was detected and read
ResultComparisonFaces	Numerical score of facial comparison
ComparisonFacesApproved	Boolean indicating if the facial comparison met approval threshold
Extras	Object containing additional verification data
Scores	Array of assessment scores for the verification

# User Redirection

After creating a verification request, you should redirect the user to the URL provided in the response:

```
https://kyc-qa.ado-tech.com/AdoQa/{key}
```

This URL contains the unique key for the verification request and enables the user to complete the identity verification process through a secure web interface.

## Redirection Methods

You can implement user redirection using various approaches:

### HTML Link:

```
<a href="https://kyc-qa.ado-tech.com/AdoQa/f7fb4984a8a347699e1c72cc5">Complete Identity Verification</a>
```

### JavaScript Redirection:

```
window.location.href = 'https://kyc-qa.ado-tech.com/AdoQa/f7fb4984a8a347699e1c72cc5';
```

### Server-Side Redirection (Example in Node.js):

```
res.redirect('https://kyc-qa.ado-tech.com/AdoQa/f7fb4984a8a347699e1c72cc5');
```

## Handling the Callback

The `callbackUrl` parameter specified when creating a flow request is crucial as it defines where the user will be redirected after completing the verification process. Your application should be prepared to handle this callback:

1. **Capture URL Parameters:** Set up your callback endpoint to capture query parameters that may contain status information.
2. **Verification Status Check:** After receiving a callback, use the "Retrieve Flow Request" endpoint to get the current status and details of the verification process.
3. **User Experience:** Display appropriate feedback to the user based on the verification result (success, pending, failure).
4. **Process Results:** Update your application's user records and proceed with the appropriate business logic based on the verification outcome.

### Example Callback Handler (Pseudocode):

```
// Callback endpoint handler
app.get('/verification-callback', async (req, res) => {
  try {
    // Extract verification key from query parameters or session
```

```
const verificationKey = req.query.key || req.session.verificationKey;

// Retrieve verification status using the API
const verificationStatus = await checkVerificationStatus(verificationKey);

// Process verification result
if (verificationStatus.success) {
  // Handle successful verification
  // Update user profile, grant access, etc.
  res.render('verification-success', { user: verificationStatus.userData });
} else {
  // Handle failed verification
  res.render('verification-failed', { reason: verificationStatus.message });
}
} catch (error) {
  // Handle errors
  console.error('Verification callback error:', error);
  res.render('error', { message: 'Unable to process verification' });
}
});
```

# Advanced Integration Considerations

## Security Best Practices

### Token Management:

- Store authentication tokens securely
- Implement token refresh mechanisms
- Never expose tokens in client-side code

### Data Encryption:

- Use HTTPS for all API communications
- Consider encrypting sensitive data before transmission
- Implement secure storage for verification results

### Error Handling:

- Implement robust error handling for API failures
- Provide friendly user feedback for verification issues
- Log errors for troubleshooting and security monitoring

## Performance Optimization

### Caching Strategy:

- Cache verification status when appropriate
- Implement efficient state management to reduce API calls

### Connection Pooling:

- Reuse HTTP connections when making multiple API calls
- Configure appropriate timeout settings

## Customization Options

The B-Trust system allows extensive customization of the verification experience:

**Branding:** The customer object in the response contains various styling parameters that define the look and feel of the verification interface:

- styleLogo: URL to your company logo
- styleColorPrimary: Primary color for UI elements
- styleColorSecondary: Secondary color for UI elements
- styleBackgroundColorBody: Background color for the page body
- styleBackgroundColorContainer: Background color for containers
- styleBackgroundcolorPrimaryButton: Background color for primary buttons
- styleColorPrimaryTextButton: Text color for primary buttons
- styleBackgroundColorSecondaryButton: Background color for secondary buttons
- styleColorSecondaryTextButton: Text color for secondary buttons

**Risk Assessment:** The riskAmount parameter allows adjustment of the verification process according to the transaction value and associated risk level.

**Flow Types:** Different flowType values enable various verification workflows tailored to specific use cases:

- Type "1": Standard enrollment process
- Other types: Contact your service provider for additional flow options

# Error Handling and Troubleshooting

## Common Error Scenarios

### Authentication Failures:

- Ensure credentials are correct
- Check token expiration
- Verify account permissions

### Invalid Parameters:

- Validate all input parameters before sending
- Check document type compatibility
- Ensure document numbers match expected formats

### Callback Issues:

- Confirm callback URL is publicly accessible
- Ensure URL encoding is handled properly
- Check for firewall or security restrictions

## Debugging Tips

**Logging:** Implement comprehensive logging for all API interactions to facilitate troubleshooting.

**Testing Environment:** Utilize the QA environment ( <https://kyc-qa.ado-tech.com> ) for testing before moving to production.

**Postman Collections:** Use the provided Postman collection for manual testing and exploration of the API.

## Webhook Implementation Summary

1. Create an endpoint in your application to receive webhook notifications.
2. Authenticate with the webhook service to obtain a token.
3. Process incoming verification data and update your application's user records.
4. Respond with appropriate status codes to acknowledge receipt of the data.

Remember that the webhook will send the complete verification result payload, including personal information, document details, and verification scores. Your webhook implementation should handle this data securely and in compliance with applicable data protection regulations.

---

Revision #5

Created 8 May 2025 14:22:35 by roger de avila

Updated 8 May 2025 15:01:37 by roger de avila