Clarity Identity Verification (IDV)
API Specification - v0.5

Version: 0.5 | **Status:** DRAFT | **Last Updated:** January 29th, 2026

---

1. Overview

1.1 Service Description

Clarity provides identity verification services that validate user identities across the full employee lifecycle, from pre-hire to post-hire:

- `IDENTITY_DOCUMENT`: Government-issued ID document verification.
- `BIOMETRIC_FACIAL`: Facial recognition against ID photo.
- `LIVENESS_CHECK`: Live person detection (anti-spoofing).
- `DEEPFAKE_DETECTION`: Analysis of AI manipulation in face or document.
- `BIOMETRIC_COMPARISON`: Mule and proxy interview detection via facial comparison.

Clarity establishes a biometric **Ground Truth** for every user during their initial interaction (e.g., CV submission or first interview). Every subsequent verification in the employee lifecycle, be it onboarding, password resets, or bulk transitions, is biometrically compared against this anchor to detect identity drift or "bait-and-switch" attacks.

1.2 Integration Architecture

Clarity operates as an **asynchronous identity verification service**. While the system provides a `202 Accepted` acknowledgement, high-assurance results (especially those requiring human-in-the-loop expert review) are delivered via webhooks or polling.

1. Your system initiates verification requests
2. Clarity processes verification asynchronously
3. Results delivered via various channels such as:
    a. Webhook callbacks
    b. Polling endpoints
    c. Messaging channels (email/slack etc)

1.3 Base URLs

```
Production: https://api.getclarity.ai/v1
Sandbox: https://sandbox-api.getclarity.ai/v1
```

---

2. Authentication

All API requests require **Bearer Token authentication**.

Required Headers:

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

API Key Management:

- Clarity will provide an API Key for your organization
- Store API keys in secure environment variables or secret managers
- Rotate keys every 90 days as a security best practice

---

3. Core API Endpoints

3.1 Initiate Verification

`POST /verifications`

Starts an asynchronous verification process. Includes new identifiers for historical context and lifecycle tracking.

Request:

```JSON
{
  "userId": "user_12345",
  "internalUniqueId": "RH_EMP_998877",
  "useCaseId": "credential_change",
  "givenName": "John",
  "familyName": "Doe",
  "verificationTypes": ["IDENTITY_DOCUMENT", "BIOMETRIC_FACIAL",
"LIVENESS_CHECK", "BIOMETRIC_COMPARISON"],
  "webhookUrl": "https://robinhood.com/webhooks/getclarity"
}
```

Field Descriptions:

- `internalUniqueId`: Your system's unique identifier (e.g., Employee ID) to track the identity chain end-to-end.
- `useCaseId`**:** Identifier for the specific lifecycle event (e.g., **`“onboarding”`, `“step-up_auth”`)**
- `verificationTypes`: Array of verification types to perform
- `redirectUrl`: URL to redirect user after completing verification
- `webhookUrl`: Endpoint to receive verification result notifications
- `expirationMinutes`: Time limit for user to complete verification (15-30 minutes recommended)

Response (202 Accepted):

```JSON
{
  "verificationId": "ver_7h8j9k0l1m2n3o4p",
  "status": "PENDING",
  "verificationUrl":
"https://verify.getclarity.ai/v/ver_7h8j9k0l1m2n3o4p",
  "expiresAt": "2026-01-28T15:30:00Z",
  "createdAt": "2026-01-28T15:00:00Z"
}
```

3.2 Check Verification Status

`GET /verifications/{verificationId}`

Polls the current status of a verification request.

Response (200 OK):

```JSON
{
  "verificationId": "ver_7h8j9k0l1m2n3o4p",
  "userId": "user_12345",
  "status": "PROCESSING",
  "verificationTypes": [
    {
      "type": "IDENTITY_DOCUMENT",
      "status": "COMPLETED",
      "result": "PASS"
    },
    {
      "type": "BIOMETRIC_FACIAL",
      "status": "PROCESSING",
      "result": null
    },
    {
      "type": "LIVENESS_CHECK",
      "status": "PENDING",
      "result": null
    }
  ],
  "overallResult": null,
  "updatedAt": "2026-01-28T15:05:00Z"
}
```

Status Values:

- `PENDING`: Awaiting user action
- `PROCESSING`: Currently being verified
- `HUMAN_EXPERT_REVIEW`: Currently being reviewed by a human expert

- `COMPLETED`: Verification finished
- `FAILED`: Technical failure
- `CANCELLED`: Manually cancelled

Result Values:

- `PASS`: Verification passed successfully with low risk
- `FAIL`: Verification failed with high risk indications

3.4 Initiate Batch Verification

POST `/verifications/batch` Supports bulk identity checks required for high-volume verification needs.

Request Example:

```json
{
  "batchName": "Acquisition_Project_X",
  "useCaseId": "m_and_a_onboarding",
  "verifications": [
    { "userId": "user_A1", "internalUniqueId": "RH_001",
"givenName": "Alice" },
    { "userId": "user_B2", "internalUniqueId": "RH_002",
"givenName": "Bob" }
  ],
  "webhookUrl": "https://robinhood.com/webhooks/getclarity-batch"
}
```

3.5 Check Batch Status

GET `/verifications/batch/{batchId}` Retrieves the current progress and results of a batch request.

Response (200 OK):

```JSON
{
  "batchId": "batch_abc123",
  "status": "PROCESSING",
  "totalCount": 50,
  "completedCount": 12,
  "results": [
    { "verificationId": "ver_7h8j9k0l...", "status": "COMPLETED",
"result": "PASS" }
  ]
}
```

3.6 Get User Verification History

GET `/users/{internalUniqueId}/history` Returns every verification event associated with a specific Employee ID, showing how they compared to the "Ground Truth" over time.

Response (200 OK):

```JSON
{
  "internalUniqueId": "RH_EMP_998877",
  "groundTruthEstablished": "2026-01-15T10:00:00Z",
  "verificationHistory": [
    {
      "verificationId": "ver_998",
      "timestamp": "2026-01-29T14:00:00Z",
      "useCaseId": "credential_change",
```

```
      "status": "COMPLETED",
      "result": "PASS",
      "comparisonToGroundTruth": "MATCH"
    },
    {
      "verificationId": "ver_442",
      "timestamp": "2026-01-15T10:00:00Z",
      "useCaseId": "onboarding",
      "status": "COMPLETED",
      "result": "PASS",
      "comparisonToGroundTruth": "ORIGINAL_TEMPLATE"
    }
  ]
}
```


4. Error Handling

4.1 HTTP Status Codes

Code	Meaning	Action
200	Success	Process response data
201	Created	Resource created successfully
202	Accepted	Request is queued; await webhook or poll status
400	Bad Request	Check request format and parameters
401	Unauthorized	Verify API key is correct
403	Forbidden	Check API key permissions
404	Not Found	Verify resource ID is correct
500	Server Error	Retry with exponential backoff
503	Service Unavailable	Clarity service temporarily down, retry later

4.2 Error Response Format

```JSON
{
  "error": {
    "code": "LIVENESS_FAILED",
    "message": "Liveness Check Failed",
    "details": {
      "reason": "The system detected a ''spoof'' attempt, such as a physical mask, a screen photo, or a deepfake injection.",
      "field": "livenessCheck"
    },
    "requestId": "req_abc123xyz"
  }
}
```

4.3 Actionable Failure Codes

I. Document Quality Failures

These codes are returned when a document is detected but the image quality prevents reliable verification.

1. **`DETECT_BLUR`**
    - **Reason**: The image is too blurry for the OCR engine to accurately extract data.
    - **Actionable Feedback**: Instruct the user to retake the photo in better lighting while keeping the camera steady.
2. **`DETECT_CUTOFF`**
    - **Reason**: Parts of the document (e.g., edges or MRZ) are outside the camera frame.
    - **Actionable Feedback**: Prompt the user to ensure the entire document is visible and fits within the on-screen guides.
3. **`DOCUMENT_DETECTION`**
    - **Reason**: No recognizable document was found in the submitted image.
    - **Actionable Feedback**: Ask the user to verify they are using a supported document type and that it is placed on a flat, contrasting surface.
4. **`DETECT_GLARE`**
    - **Reason**: A strong reflection or glare is obscuring critical security features or text.
    - **Actionable Feedback**: Suggest the user move away from direct light sources or change the angle of the document to minimize reflections.

II. Document Authenticity & Validity Failures

These codes indicate issues with the legal status or physical integrity of the ID document.

5. `EXPIRED_DOCUMENT`
    - **Reason**: The document's expiry date has passed and it is no longer legally valid.
    - **Actionable Feedback**: Request the user to provide a current, valid form of identification.
6. `UNSUPPORTED_DOCUMENT`
    - **Reason**: The document type is valid but not currently supported for this specific automated flow.
    - **Actionable Feedback**: Provide the user with a list of accepted document types (e.g., Passport, Driver's License).
7. `PICTURE_FACE_INTEGRITY`
    - **Reason**: Signs of physical tampering were detected on the printed photo of the ID.
    - **Actionable Feedback**: This is a high-risk failure; do not offer an automated retry. Escalate to the security team for manual review.
8. `DIGITAL_TAMPERING`
    - **Reason**: Metadata or pixel analysis suggests the image was digitally edited or manipulated.
    - **Actionable Feedbac**k: Reject the session. This is typically an indicator of a bad actor attempt.

III. Biometric & Facial Similarity Failures

These codes relate to the "Ground Truth" comparison and the liveness of the person performing the check.

9. `FACE_MISMATCH`
    - **Reason**: The live person does not match the photo on the ID or the established "Ground Truth" template.
    - **Actionable Feedback**: Flag as a potential "proxy interview" or "bait-and-switch" case. Escalate to Insider Threat.
10. `LIVENESS_FAILED`
    - **Reason**: The system detected a "spoof" attempt, such as a physical mask, a screen photo, or a deepfake injection.
    - **Actionable Feedback**: If high confidence, reject. If inconclusive, request a guided "active liveness" check (e.g., following a dot on screen).
11. `FACE_NOT_FOUND`
    - **Reason**: No face was detected in the selfie or video capture.
    - **Actionable Feedback**: Remind the user to center their face in the frame and remove any obstructions like heavy masks or hats.

IV. System & Permission Failures

These codes address technical barriers to completing the IDV process.

12. `USER_LOCKED_OUT`
    - **Reason**: The user has exceeded the maximum number of failed submission attempts for this `verificationId`.
    - **Actionable Feedback**: Instruct the user to contact the Robinhood internal helpdesk to have their verification count reset.
13. `INSUFFICIENT_PERMISSION`
    - **Reason**: The user's device or browser denied access to the camera.
    - **Actionable Feedback**: Provide a link to instructions on how to enable camera permissions in device settings.

Document Version: 0.5
Last Updated: January 29th, 2026