

Liveness Detection API

Base URL: <https://api.taskinogo.com> · Version v1

Overview

The Taskinogo Liveness Detection API verifies that a **real, live person** is in front of the camera (anti-spoofing against printed photos, screen replays, and masks) and that they performed the requested **challenge actions** (head movements, blink). It runs as a synchronous, server-to-server REST API.

On a successful check you receive a clear verdict, a confidence score, a 512-dimension face embedding (ArcFace), and the sharpest frontal frame. Media is processed in memory and discarded — we never store biometric data (see Privacy & data).

Calls are **server-to-server**. Keep your API key on your backend — never embed it in a mobile or web client. Your app captures the camera frames on the device and sends them through your own server to this API.

Authentication

Every request must include your API key in the `X-AI-API-Key` header. You receive your key by email after requesting access. Treat it like a password.

```
header
```

```
X-AI-API-Key: tsk_your_api_key_here
```

Liveness must be enabled on your key. If it is not, the API returns `403 LIVENESS_DISABLED` — contact us to enable it.

Quickstart

A liveness check is always two calls: create a session, then verify the capture.

```
shell
```

```
# 1) Create a session – returns the challenge order and input limits
curl -X POST https://api.taskinogo.com/v1/liveness/session \
  -H "X-AI-API-Key: $TASKINOGO_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{}'
```

```
# 2) Capture the selfie sequence, then verify it
curl -X POST https://api.taskinogo.com/v1/liveness/verify \
  -H "X-AI-API-Key: $TASKINOGO_API_KEY" \
  -H "Content-Type: application/json" \
  -d @verify_payload.json
```

How it works

01

Session

Create a session. The response returns a **random challenge order** and the input limits to honor.

02

Capture

Guide the user through the challenge and capture frames (or a short video), tagging each frame with its phase.

03

Verify

Send the capture to `/verify` and receive a synchronous real-or-spoof verdict.

A session is single-use and short-lived (default 5 minutes). Create a new one for every check.

Create a session

POST /v1/liveness/session

The body is **optional**. Send {} to use your key's defaults, or override the challenge and angles for this session. Any field you omit keeps its default.

request

```
curl -X POST https://api.taskinogo.com/v1/liveness/session \  
-H "X-AI-API-Key: $TASKINOGO_API_KEY" \  
-H "Content-Type: application/json" \  
-d '{  
  "actions": ["turn_left", "blink", "turn_right"],  
  "yaw_deg": 25,  
  "pitch_deg": 18,  
  "blink_ear": 0.20  
'
```

Optional body fields

FIELD	TYPE	ALLOWED	DEFAULT
<code>actions</code>	string[]	Subset of <code>turn_left</code> , <code>turn_right</code> , <code>turn_up</code> , <code>turn_down</code> , <code>blink</code> . Min 2, order preserved.	Your key's full set, shuffled
<code>yaw_deg</code>	number	15-40	25
<code>pitch_deg</code>	number	10-30	20
<code>blink_ear</code>	number	0.12-0.28	0.20

Values outside the allowed range return 400. Response:

200 application/json

```
{  
  "session_id": "live_a1b2c3d4e5f6...",  
  "challenge": ["turn_left", "blink", "turn_right"],  
  "expires_at": "2026-05-25T14:35:00.000Z",  
  "config": {  
    "min_images": 8,  
    "max_images": 20,  
    "max_video_size_mb": 25,  
    "accepted_modes": ["images", "video"]  
  }  
}
```

Capture in the exact order of `challenge`, and respect `config.min_images / max_images` and `max_video_size_mb`.

Verify

`POST /v1/liveness/verify`

Send the captured sequence for the session. Returns a synchronous verdict. Supports two input modes.

Images mode (recommended)

Your app extracts frames during the challenge and tags each with a phase. Send **8–20 frames** (see config). Each frame is a JPEG, base64-encoded (no data-URL prefix needed).

FRAME FIELD	TYPE	DESCRIPTION
<code>index</code>	number	Sequence index, starting at 0.
<code>timestamp_ms</code>	number	Milliseconds since capture start.
<code>phase</code>	string	The action this frame belongs to (see Challenges). Must start with the action name.
<code>image_b64</code>	string	Base64 JPEG (quality 80–90, ≥720p recommended).

```
request - mode: images
```

```
{
  "session_id": "live_a1b2c3d4...",
  "mode": "images",
  "frames": [
    { "index": 0, "timestamp_ms": 0, "phase": "center", "image_b64": "/9j/4AAQ..." },
    { "index": 1, "timestamp_ms": 200, "phase": "center", "image_b64": "/9j/4AAQ..." },
    { "index": 2, "timestamp_ms": 600, "phase": "turn_left", "image_b64": "/9j/4AAQ..." },
    { "index": 3, "timestamp_ms": 800, "phase": "turn_left", "image_b64": "/9j/4AAQ..." },
    { "index": 4, "timestamp_ms": 1200, "phase": "blink", "image_b64": "/9j/4AAQ..." },
    { "index": 5, "timestamp_ms": 1400, "phase": "blink", "image_b64": "/9j/4AAQ..." },
    { "index": 6, "timestamp_ms": 1800, "phase": "turn_right", "image_b64": "/9j/4AAQ..." },
    { "index": 7, "timestamp_ms": 2000, "phase": "turn_right", "image_b64": "/9j/4AAQ..." }
  ]
}
```

Video mode

If your client can't segment frames, record a short clip (~5s, mp4 or webm) while the user follows the prompts and send it base64-encoded. The API extracts and auto-segments frames. Keep it under `config.max_video_size_mb`.

```
request - mode: video
```

```
{
  "session_id": "live_a1b2c3d4...",
  "mode": "video",
  "video_b64": "GkXfo59ChoEBQveBAULygQRC..." // base64 of an mp4/webm clip
}
```

Challenges & angles

Five challenge actions are supported. Tag each captured frame's phase with the action it belongs to.

ACTION	PHASE TAG	PASSES WHEN
<code>turn_left</code>	<code>turn_left</code>	Yaw \leq <code>-yaw_deg</code> (head physically left)
<code>turn_right</code>	<code>turn_right</code>	Yaw \geq <code>+yaw_deg</code>
<code>turn_up</code>	<code>turn_up</code>	Pitch \leq <code>-pitch_deg</code>
<code>turn_down</code>	<code>turn_down</code>	Pitch \geq <code>+pitch_deg</code>
<code>blink</code>	<code>blink</code>	Eye-aspect-ratio dips below <code>blink_ear</code> then reopens

Phase rule: a frame counts toward an action if its phase starts with that action's name — so `turn_left`, `turn_left_start`, `turn_left_end` all count for `turn_left`. Capture 2–3 frames per action (and 2 neutral center frames at the start) so the peak of the movement is included.

Sign convention (selfie / front camera): turning the head physically *left* yields a *negative* yaw; turning *right* yields positive. Looking *up* is negative pitch; looking *down* is positive. For `blink` mode is order-independent, but at least 3 frames with eyes visible are needed.

Response & verdict

A successful check (`verified: true`) returns the verdict, scores, the face embedding, and the best frame:

```
200 - verified
```

```
{
  "verified": true,
  "confidence": 0.94,
  "anti_spoof_avg_score": 0.99,
  "same_person_score": 0.87,
  "challenge_passed": true,
  "challenge_details": {
    "passed": true,
    "completed_actions": ["turn_left", "blink", "turn_right"]
  },
  "embedding": [0.0123, -0.0456, "... 512 floats ..."],
  "best_frame_b64": "/9j/4AAQSkZJRg...",
  "best_frame_index": 6,
  "reason_codes": ["liveness_passed", "challenge_completed", "no_spoof_detected"],
  "processing_time_ms": 312,
  "frames_analyzed": 8
}
```

A rejected check (`verified: false`) returns the reason and details, and **no** embedding or best frame:

```
200 - rejected
```

```
{
  "verified": false,
  "anti_spoof_avg_score": 0.41,
  "same_person_score": 0.78,
  "reason_codes": ["spoof_detected"],
  "rejection_details": { "anti_spoof_avg_score": 0.41, "threshold": 0.85 },
  "processing_time_ms": 180
}
```

FIELD	MEANING
<code>verified</code>	Overall verdict — real (true) or spoof/failed (false).
<code>confidence</code>	0–1 combined confidence (anti-spoof + identity + quality).
<code>anti_spoof_avg_score</code>	0–1 average anti-spoof score across frames (higher = more likely real).
<code>same_person_score</code>	0–1 consistency that all frames are the same person.

FIELD	MEANING
<code>embedding</code>	512-float ArcFace embedding (only when verified). It is yours; we don't store it.
<code>best_frame_b64</code>	Sharpest frontal frame as base64 JPEG (only when verified).
<code>processing_time_ms</code>	Server processing time in milliseconds.

Reason codes

CODE	MEANING
<code>liveness_passed</code>	All checks passed.
<code>challenge_completed</code>	Every requested action was detected.
<code>no_spoof_detected</code>	Anti-spoof threshold cleared.
<code>spoof_detected</code>	Below anti-spoof threshold (photo, screen replay, mask).
<code>challenge_failed</code>	One or more actions not detected (see <code>rejection_details</code>).
<code>different_persons_detected</code>	Frames inconsistent — more than one face or a swap.
<code>insufficient_face_detections</code>	A face was visible in fewer than 70% of frames.
<code>invalid_input</code>	Missing or malformed frames / video.
<code>internal_error</code>	Unexpected processing error — safe to retry with a new session.

Errors

Errors return a JSON body { "error": "...", "code": "..." } with these HTTP statuses:

HTTP	CODE	WHAT TO DO
400	MISSING_FIELDS, INVALID_INPUT, INVALID_FRAME_COUNT, INVALID_FRAME_FORMAT	Fix the payload / frame count / field types.
401	—	Missing or invalid API key.
403	LIVENESS_DISABLED, SESSION_FORBIDDEN	Key lacks liveness, or session belongs to another key.
404	SESSION_NOT_FOUND	Create a new session.
409	SESSION_USED	Session already used — create a new one.
410	SESSION_EXPIRED	Session expired (default TTL 5 min) — create a new one.
413	VIDEO_TOO_LARGE	Reduce the clip below max_video_size_mb.
429	RATE_LIMIT, CONCURRENCY_LIMIT	Back off and retry.
502	PROCESSING_ERROR	Retry with a new session.

Retry rule: 429 → exponential backoff. 5xx → new session and retry. 4xx (except 429) → fix the request, don't retry.

Rate limits & sessions

- **Single-use sessions.** Each session verifies once; a second attempt returns 409 SESSION_USED.
- **TTL.** Sessions expire after 5 minutes by default — verify promptly.
- **Rate & concurrency limits** apply per API key; exceeding them returns 429.

Session status (optional)

Poll a session's status (no biometric data) with:

```
GET /v1/liveness/session/:id
```

```
curl https://api.taskinogo.com/v1/liveness/session/live_a1b2c3d4... \  
-H "X-AI-API-Key: $TASKINOGO_API_KEY"
```

A public health check is available at GET /v1/liveness/health (no auth).

Code examples

Node.js

node.js

```
const BASE = "https://api.taskinogo.com";
const KEY = process.env.TASKINOGO_API_KEY;
const headers = { "X-AI-API-Key": KEY, "Content-Type": "application/json" };

async function runLiveness(captureFrames) {
  // 1) Create a session and read the challenge order.
  const session = await fetch(`${BASE}/v1/liveness/session`, {
    method: "POST",
    headers,
    body: JSON.stringify({ actions: ["turn_left", "turn_right", "blink"] }),
  }).then((r) => r.json());

  // 2) Capture frames in the order of session.challenge (your camera code).
  const frames = await captureFrames(session.challenge);

  // 3) Verify.
  const result = await fetch(`${BASE}/v1/liveness/verify`, {
    method: "POST",
    headers,
    body: JSON.stringify({ session_id: session.session_id, mode: "images", frames }),
  }).then((r) => r.json());

  return result; // { verified, confidence, embedding, ... }
}
```

Python

python

```
import os, requests

BASE = "https://api.taskinogo.com"
H = {"X-AI-API-Key": os.environ["TASKINOGO_API_KEY"], "Content-Type": "application/json"}

# 1) Create a session.
session = requests.post(f"{BASE}/v1/liveness/session", headers=H,
                       json={"actions": ["turn_left", "turn_right", "blink"]}).json()

# 2) Capture frames in the order of session["challenge"], then verify.
result = requests.post(f"{BASE}/v1/liveness/verify", headers=H, json={
    "session_id": session["session_id"],
```

```
"mode": "images",
"frames": frames, # list of {index, timestamp_ms, phase, image_b64}
}).json()

print(result["verified"], result.get("confidence"))
```

Best practices

- Capture 2–3 frames per action plus 2 neutral center frames, ~200–300 ms apart.
- Give the user a clear on-screen prompt and a moment to complete each movement before sampling its frames.
- Use $\geq 720p$, JPEG quality 80–90, even lighting, one face in frame.
- On weak networks prefer images mode with cropped frames over a full video.
- Keep the key on your server; call this API from your backend, never directly from the device.
- Treat `verified` as the gate; use `confidence` only for additional risk logic, not as the sole control for high-risk decisions.

Privacy & data

Submitted images and video are processed in memory and discarded immediately after scoring. We do **not** store biometric templates or embeddings — the embedding and best frame are returned to you and never persisted on our side. We keep only a request audit log (timestamp, key, mode, verdict, scores, timing — no media). You remain the data controller for the faces you submit; ensure you have a lawful basis and any required consent.

Need a key or have a question? [Contact us](#).
