bonify

Bonify integration guide

Governed bonus intelligence

Bonify recommends rewards using anonymous activity, deterministic policy and constrained model assistance. Your platform retains approval, delivery and financial execution.

Current endpoint

Production base URL: `https://bonify.pro/api/production`.

The interactive reference is `/swagger`; the generated OpenAPI document is `/api/openapi`. These describe the current implementation. Sandbox requires its own configured environment and must not be assumed to share production data.

Authentication

Create a workspace account at `/signup`. In Integration, create an API client and key. The complete key is shown once. Store it on your server; never ship it in browser code. The key binds requests to one operator. Sending a different operator identifier does not change ownership.

Keys start PENDING. Activate them with `POST /v1/auth/activate`, including `X-API-Key`, `Content-Type: application/json` and a unique `Idempotency-Key` header. The body is:

```
{

  "environment": "production",

  "verification": "BONIFY_API_KEY_ACTIVATION_V1"

}
```

A legacy `operator_code` assertion is optional; a mismatch is rejected. Read/write scopes remain enforced. External API keys cannot call `/v1/admin`.

Request behavior

Use a stable idempotency key when retrying the same write, and a new key for a new operation. Preserve request IDs for support and audit. Treat validation failures and revision conflicts as actionable responses; do not blindly retry them. Keep

response state and timestamps intact.

Main resource families

- Metadata, configuration and integration verification.
- Anonymous player/event ingestion and outcome reporting.
- Decision evaluation, review and explanations.
- Versioned operator rules, Model guidance and model-input policy.
- Reward catalogs, mappings, entitlements and controlled experiments.
- Feedback, audit and bounded activity/change streams.

Use the generated reference for exact payloads and endpoints. It is more precise than duplicated static examples.

State and authority

A suggestion is not a platform approval. Approval is not delivery, and delivery is not a realized outcome. Keep draft/published/archived configuration states separate. Bonify never grants a client authority to invent a payout, override safety or write a balance. Connected platforms remain authoritative for execution.

Streams

Use the documented cursor or Last-Event-ID contract. Reconnect without dropping the cursor, honor explicit resynchronization requirements and close subscriptions when the view or client is disposed. Never render raw prompts, credentials, personal player data or hidden model reasoning.

Private inference

Model transport is server-only and independent of external platform API keys. The Mac Studio preparation is described in BONIFY_AI_INTEGRATION.md.

Private inference and Mac Studio preparation

Current architecture

The API's `LocalInferenceCoordinator` supplies the active Ollama provider to in-process decision orchestration. It preserves the model digest pin, server-side candidate allowlist, one-request concurrency, durable decision lease, timeouts and deterministic fallback. The standalone intelligence service is not a replacement production Ollama gateway.

The existing Hostinger model remains active. Mac Studio support is prepared in code; no Mac endpoint or private network identity has been supplied or verified yet.

Supported private transport

`OLLAMA_BASE_URL` accepts loopback, RFC1918 IPv4, private `.internal` service names, Tailscale IPv4 `100.64.0.0/10`, loopback IPv6 and the Tailscale IPv6 prefix. HTTPS Tailscale hostnames must be qualified `.ts.net` names. Public endpoints, link-local metadata addresses, URL credentials, query strings and fragments are rejected.

Set `OLLAMA_RUNTIME_TARGET=vps` for the existing host or `mac-studio` for an explicitly configured remote endpoint. Mac mode rejects a loopback/default local endpoint. The target label is not evidence of connectivity.

An optional `OLLAMA_AUTH_TOKEN` is a server-only bearer credential for a gateway that actually enforces it. It is attached to tag checks and chat calls, never returned by health or exposed to the browser. Redirects are rejected, and raw gateway error bodies are not retained. Native Ollama does not itself add bearer authentication.

Prepare the Mac

1. Install Ollama and Tailscale using their official instructions.

2. Keep Ollama bound to `127.0.0.1:11434`; install the selected model and confirm its exact digest.

3. Use private Tailscale Serve for the loopback service. Do not expose port 11434 to the internet and do not use Funnel.

4. Restrict tailnet policy so only the Hostinger service can reach the Mac inference endpoint.

5. Verify access from the API container network, not only from the VPS shell.

A typical private proxy command is `tailscale serve --bg http://127.0.0.1:11434`. Verify the installed Tailscale CLI's current syntax before use.

Controlled cutover

Keep the prior environment securely for rollback. Configure the remote target, private base URL, model and matching digest in protected server environment storage. Restart the API, check authenticated Inference connection, and execute a bounded structured inference. Confirm gateway authentication where used and verify that a missing model, timeout or invalid response retains fallback behavior. Only then treat the remote model as available.

The UI distinguishes endpoint/model verification from recorded successful generation. It displays the real configured decision and insight cadence rather than a hardcoded interval.

References

- Tailscale Serve

- Tailscale addresses

- Ollama configuration