# Metrisque API

Partner reference · v1 · DRAFT for review, September 2026

01

## What this API is

Metrisque measures how AI assistants read, file, and recommend a brand: whether the models know it, which category shelf its pages land on, whether its content matches real buyer questions, and which brands the assistants actually pick. The API exposes the same measurements the Metrisque interface runs, through the same internal pipeline, the API is never a fork of the product. Every number a partner receives comes from a stored, re-queryable run.

It is built for agencies running measurements across a client portfolio, and for brands wiring AI-visibility readings into their own dashboards and workflows.

02

## Access and authentication

The API is paid-only. Keys are issued on approval: request access from Settings → API in the app (or ask your Metrisque contact), and your key arrives by email. A key is shown once at creation, store it like a password.

Every request carries the key in a header:

| HEADER |
| --- |
| `x-api-key: <your key>` |

Base URL:

| BASE URL |
| --- |
| `https://metrisque.com/api/v1` |

Rate limit: 60 requests per minute per key. Exceeding it returns HTTP 429 with error code `RATE_LIMIT`.

**03**

# Credits, the only meter

Every charge is a credit. Your key is attached to a plan that sets a monthly credit allowance; each measuring call holds credits before it runs and settles the charge only when it returns a result. Three consequences worth designing around:

- A failed run costs nothing, the hold is released.
- A repeat of a measurement already paid for serves the stored result and costs nothing.
- Polling an async job is always free, polls can never bill.

There is no separate request counter or model-call cap: if you hold credits, you can spend them. `GET /usage` returns your live balance and a per-surface breakdown.

**04**

# Conventions

**Envelope.** Every JSON response is wrapped as:

ENVELOPE

```
{ "v": "1", "request_id": "...", ...payload }
```

Quote the `request_id` in any support conversation, it identifies the exact call in our logs.

**Errors.** Errors return a JSON body with a stable error code and a human message, e.g. `BAD_INPUT`, `RATE_LIMIT`, `UNAUTHORIZED`. HTTP status matches the class (400 / 401 / 429 / 500).

**Ranks are 1-indexed.** Everywhere a position appears in the public API, 1 is first.

**Deprecations are non-breaking.** A renamed endpoint keeps its old path alive for existing keys; deprecated responses carry a `Deprecation` header and a note naming the replacement.

05

# Endpoints

| ENDPOINT | WHAT IT MEASURES | COST CLASS |
|---|---|---|
| `POST /memory` | Brand memory probe: do the models know this brand, and are its declared facts right | Credits · model calls |
| `POST /picks` | Per-model picks for a buyer query, with brand, title, and rank | Credits · model calls |
| `POST /consideration` | Per-model consideration ladders (up to 15 names) with positions and picked flags | Credits · model calls |
| `POST /meaning` | Where a page or text lands against a buyer question | Credits · embedding only |
| `POST /placement` | Store/product placement audit against the declared shelf | Credits · embedding only |
| `POST /attribution` | Per-phrase deltas for a text or URL: which words carry the fit, which cost it | Credits · embedding only |
| `POST /content` | Measure a text against a question: where it lands, what helps, what costs | Credits · embedding only |
| `POST /audit` | Whole-business Category Fit as an async job | Credits, settled on completion |
| `GET /audit/{report_id}` | Status and result of an audit job | Free |
| `POST /questions` | Create a saved buyer question | Credits |
| `GET /questions · GET/PATCH/DELETE /questions/{id}` | List, read, relabel, delete saved questions | Free |
| `GET /usage` | Credit balance and per-surface request breakdown | Free |

## `POST /memory`

Probes the three assistants' memory of a brand and checks their answers against the facts you declared. A brand must have a declared display name and at least one substantive fact on record, or the call returns `brand_name_required`, declare them in the app first.

REQUEST

```
{ "host": "acme.com" }
```

Response: per-model known/unknown with the stored answers, fact-check results (false facts named with claimed vs actual), and which declared angles were not mentioned.

### POST /picks

Runs a buyer query and returns each model's ranked picks.

REQUEST

```
{ "query": "best b2b buyer intent data", "channel": "memory" }
```

`channel` is "memory" (no browsing, the models' stored beliefs) or "search" (live retrieval). Results are cached per (query, channel); a repeat serves the stored run free. Optional: "models" narrows which assistants run.

### POST /consideration

The deliberation layer: each model lists up to 15 names it would weigh for the query, then its picks from that list. Positions are 1-indexed in this API. Request shape matches `/picks` (query; channel where applicable).

### POST /meaning

Locates a page or a text against a buyer question: how close it sits and where it lands. Send the question plus either a url to fetch or the text itself. This is the replacement for the legacy `/address` endpoint.

### POST /placement

The Category Fit audit for a store or product set, graded against the shelf you declare. Accepts `store_url`, an optional panel selector ("products" default; services panels by name), and the declared shelf. Response includes per-product landed paths and verdicts. Note: `placement_score` in the response is a composite fit index on a 0 to 100 display scale, an internal diagnostic, not a grade; every response carries this note.

### POST /attribution

Measures every phrase in a text (or a fetched URL) by removing it and re-scoring against a target shelf or question: per-phrase deltas plus a keep/remove handoff for rewriting. Embeddings are cached after the first run, so re-measuring an unchanged text is free.

### POST /content

The Content measurement: where a text lands against a question and which phrases help or cost. Caching keys on the text itself, so repeats of an unchanged draft cost nothing.

### POST /audit and GET /audit/{report_id}

A real catalog outruns an HTTP timeout, so the whole-business audit is asynchronous: POST accepts the job and returns status "running" with a `report_id`; poll the GET until status is "complete" (or "failed", with the reason). Polls are free and always safe, the charge settles exactly once on the completed audit, and a dropped run resumes through the same poll.

REQUEST

```
{ "store_url": "https://acme.com", "panel": "products" }
```

A single offering can be audited with scope "offering" and an offering url instead of the full store.

### POST /questions and GET/PATCH/DELETE

Saved buyer questions, shared with the app's Buyer Match panel. Create is charged; everything else is free.

CREATE REQUEST

```
{ "store_url": "acme.com", "question_text": "best b2b buyer intent data",
  "label": "Intent head term" }
```

Creating a question classifies it against your existing measured segments and starts no run, new data collection is always a deliberate, separate step. Below the classification floor the answer is an explicit "not classified", never a guess.

### GET /usage

Free. Returns this month's credit balance (used and included) and a per-endpoint request breakdown. Metadata only, query text is never logged to this surface.

**06**

# Deprecated endpoints

| OLD | STATUS | USE INSTEAD |
|---|---|---|
| `GET /address` | Legacy; kept for keys minted against it; Deprecation header on every response; not being rebuilt | `POST /meaning` |
| `POST /meaning-locate` | Deprecated alias | `POST /meaning` |
| `POST /memory-score` | Deprecated alias | `POST /memory` |