Topit Taste API: integration guide for coding agents

For partner engineers and coding agents. Base URL: `https://partners.mytopit.com/v1`. Everything here is also in the console at partners.mytopit.com.

Paste this to your coding agent

```
You are integrating the Topit Taste API (base URL https://partners.mytopit.com/v1) into our
server-side image pipeline. Follow these rules:

1. Read the API key only from the server environment variable TOPIT_API_KEY. Never hard-code
   it, never send it to a browser or mobile app, never print or log it, never commit it.
   Every request sends the header: Authorization: Bearer $TOPIT_API_KEY
2. Call Topit only from our backend. Use the request and response shapes in the guide at
   https://mytopit.com/business/taste-api/agent-guide
   (PDF: https://mytopit.com/business/taste-api-agent-guide.pdf). If a field, route or
   error is not in the guide, do not guess: check the guide again, or stop and ask a human.
3. Score candidate images with POST /v1/score, sending "brand" (if not the default brand)
   and a "ref" per image. Do not publish an image whose held_by_floor is true.
4. Handle 401, 404, 409, 422, 429 and 503 as the guide says. Retry 429 and 503 only after
   the Retry-After header. Never retry 401, 404, 409 or 422 unchanged.
5. Write tests that use mocked Topit responses. Do not put real keys in tests or fixtures.
```

1. Keys and authentication

1. Use your live server key. It is shown once, when it is created in the console. Store it in a secrets manager and expose it to your backend as the environment variable `TOPIT_API_KEY`.
2. The server key never goes in a browser, a mobile app, a log line, an error report, a screenshot or a git commit. If it leaks, rotate it in the console (Keys) and update the environment variable.
3. Every request sends exactly this header, plus `Content-Type: application/json` for JSON bodies:

```
Authorization: Bearer $TOPIT_API_KEY
```

A missing or wrong key answers `401 {"detail": "invalid key"}`. Every key has a per-minute rate limit; a `429` carries `Retry-After`.

2. Scoring images: POST /v1/score

Scoring is turned on per account. If `/v1/score` answers 404, ask Topit to turn it on for yours.

Send 1 to 10 images per call. Each image has your own `ref` and exactly one source: `url` (a public https URL, up to 8 MB), `image_base64` (all base64 images in one call together at most about 6.4 MB), or `asset_id` (an image you already sent to `/v1/assets`). JPEG, PNG, WebP, GIF, BMP and TIFF are read. The answer comes back in the same response. Topit keeps no image from this call.

```
curl -X POST https://partners.mytopit.com/v1/score \
  -H "Authorization: Bearer $TOPIT_API_KEY" \
  -H "Content-Type: application/json" \
  -d '{
    "brand": "client-a",
    "assets": [
      {"ref": "gen-001-a", "url": "https://cdn.example.com/gen/001-a.jpg"},
      {"ref": "gen-001-b", "url": "https://cdn.example.com/gen/001-b.jpg"}
    ]
  }'
```

Response (200):

```
{
  "results": [
    {"ref": "gen-001-a", "brand_fit": 0.929, "score": 92.9, "band": "top",
     "fit_floor": 0.5, "held_by_floor": false, "percentile_in_catalog": 25.0,
     "embedded": "service", "latency_ms": 640.2},
    {"ref": "gen-001-b", "brand_fit": 0.6275, "score": 62.7, "band": "good",
     "fit_floor": 0.5, "held_by_floor": false, "percentile_in_catalog": 12.5,
     "embedded": "service", "latency_ms": 655.8}
  ],
  "label": "brand fit, calibrated to your reference library",
  "note": "Brand fit measures how close an image looks to your reference library ...",
  "latency_ms": 702.4,
  "brand": "client-a"
}
```

Field	Meaning
`score`	Brand fit from 0 to 100 (`brand_fit` times 100), measured against the brand's reference images.
`held_by_floor`	True when the image is below the brand's floor (`fit_floor`) and should not be published.
`percentile_in_catalog`	Where the image ranks among the brand's own active images (absent while the brand has no active images).
`error` (per image)	An image Topit cannot read comes back as `{"ref", "error": {"code", "message"}}`, for example `not_an_image` or `too_many_pixels`; the other images in the call are still scored.

`brand` is optional. Without it the image is scored against your default brand.

3. Brands and reference images

1. The default brand is your own. Any call without `brand` uses it. Its reference images are at `POST /v1/brand/reference`.
2. If your account scores images for several brands (turned on per account), create each brand once: `POST /v1/brands` with `{"key": "client-a", "name": "Client A"}` (201). The key is 1 to 40 lowercase letters, digits and hyphens. `GET /v1/brands` lists your brands and how many reference images each has. You can also do this on the console's Brand page.

3. References. A named brand is scored once it has at least 10 reference images (on-brand images it has already used); the default brand is scored once its reference library is calibrated. Add them in any of three ways:

```
# a) by URL (each needs your own external_id)
POST /v1/brands/client-a/reference
{"assets": [{"external_id": "client-a-ref-01", "url": "https://cdn.example.com/client-a/ref-
01.jpg"}, ...]}

# b) by file upload (multipart, field "files", up to 300 files of 8 MB)
POST /v1/brands/client-a/reference/upload

# c) images you already sent to /v1/assets for this brand
POST /v1/brands/client-a/reference
{"external_ids": ["client-a-img-0001", "client-a-img-0002", ...]}
```

   For the default brand, use the same bodies at `/v1/brand/reference` and `/v1/brand/reference/upload`.

4. `brand` on other calls. `POST /v1/score` takes `"brand": "client-a"` (section 2). `POST /v1/assets` takes `"brand": "client-a"` so the images in that call belong to that brand; an image stays in the brand it was first added to. `GET /v1/assets?brand=client-a` lists one brand's images (`brand=default` for the default brand).

Each brand is scored only against its own references, floor and settings. A brand is never scored with another brand's library.

4. Errors and what to do

Response	Meaning	Fix
`401 {"detail": "invalid key"}`	Missing, wrong or revoked key.	Check that `TOPIT_API_KEY` is set on the server and the header is exactly `Authorization: Bearer <key>`. Do not retry unchanged.
`404`	Scoring is not turned on for your account, or `unknown_brand`: no brand with that key on your account.	Ask Topit to turn scoring on; check the brand key against `GET /v1/brands`, or create it with `POST /v1/brands`.
`409 no_reference_library`	A named brand has fewer than 10 reference images: the body names `brand`, `reference_count` and `reference_minimum`. For the default brand, its reference library is not calibrated yet, and the body is a plain message.	Add references (section 3), then score again.
`422`	The request is malformed: more than 10 images, or an image without exactly one source. The body names the field.	Fix the request. Do not retry unchanged.
`429 score_busy`	Too many of your score calls are already in flight.	Wait for `Retry-After` (1 second), then retry.

`429` rate limit	Too many requests this minute.	Wait for the `Retry-After` header, then retry.
`429 monthly_call_cap`	Your account's monthly call limit is used, if your plan has one.	Stop retrying until `Retry-After` (the next month) or until the limit is raised.
`503 embed_busy`	Topit is briefly at capacity.	Wait for `Retry-After`, then retry with backoff.

Error bodies look like `{"detail": {"error": "unknown_brand", "message": "no brand 'nope' for this partner; GET /v1/brands lists them"}}`. Log `detail.error` and the message; never log the key.

5. Usage: GET /v1/usage

Monthly usage, newest first (`?months=1` for this month only). Each month includes `scoring: {calls, assets_scored}` and, per brand, `brands: [{brand, calls, assets_scored}]` (`brand: null` is the default brand). The console's Usage page shows the same numbers.

```
GET /v1/usage?months=1
-> {"months": [{"month": "2026-11-01",
                "scoring": {"calls": 1, "assets_scored": 2},
                "brands": [{"brand": "client-a", "calls": 1, "assets_scored": 2}], ...}], ...}
```

Checklist before going live

1. `TOPIT_API_KEY` is set only in the server environment and appears in no code, logs or client bundle.
2. Every named brand you score has at least 10 references (`GET /v1/brands`).
3. Every call sends `brand` (unless it is the default brand) and a unique `ref` per image.
4. Images with `held_by_floor: true` are not published.
5. 429 and 503 are retried only after `Retry-After`; other errors are logged and fixed.