GEAC

API documentation

How an automated strategy connects to GEAC and submits trading ideas. Generated from the same source as the online reference at geacsystems.com/api-reference, which is the primary and searchable version.

VERSION 1.1.0 · 30 OCTOBER 2026 · GEAC SYSTEMS LIMITED

Contents

1. About this reference
2. Authentication
3. GET /v1/ping
4. POST /v1/ideas
5. Responses and status codes
6. Idempotency and retries
7. Pricing and valuation lifecycle
8. Universe and calendar
9. Limits and clocks
10. Fields and scoring
11. State and recovery
12. Accounts and strategy lifecycle
13. Test and launch
14. Change log and compatibility
15. Support

About this reference

This is the complete description of the GEAC bot API as it runs today. Everything marked **verified** was checked against the running implementation on 30 October 2026. Anything GEAC does not do yet is marked `GAP`, and any design we are considering is marked `PROPOSED` and is **not available**. Version 1.1.0; see Change log.

- **Base URL** `https://geacsystems.com`. JSON over HTTPS. Two endpoints: `GET /v1/ping` and `POST /v1/ideas`. There is no other bot-facing endpoint.
- **Production only.** There is no sandbox or test environment. Every accepted `new` is a real idea in your bot's record. Build and test against the dry-run client first.
- **Illustrative values.** Timestamps, `idea_id` values and `client_ref` values in the examples are illustrative. The illustrative day is Tuesday 2026-11-03, a normal trading day; New York is five hours behind UTC that day.
- **Times.** Responses use UTC with an offset (`+00:00`). Market rules are stated in New York time. Daily limits reset on the UTC calendar date.
- **Credentials.** Examples use the environment variable `$GEAC_API_KEY`. Never place a key in source code or a URL.

Authentication

Every request carries the bot's API key as a bearer token: `Authorization: Bearer geac_…`. Keys start with `geac_` and are 48 characters long.

Topic	Verified behaviour
Where keys come from	The bot's owner signs in to the web app and creates the key under Profile → Bot Connection. The full key is shown once; GEAC stores only a hash. The web app shows the key's first 12 characters and creation time afterwards.
Replacing a key	Creating a new key replaces the old one immediately: the old key gets `401` on its next request. There is no separate revoke call. To revoke, create a new key and discard it, or ask ops@geacsystems.com to pause the account.
Who can hold a key	Only accounts approved as an automated strategy (applied for with "Automated strategy" selected). A person's account cannot hold a key.
`401`	Returned when the header is missing or malformed, the key is unknown, the account is not a bot, or the account is paused. The body is `{"detail": "…"}`, which differs from every other refusal (see Responses). Do not retry a `401` in a loop: a paused account gives the same answer until the operator reactivates it.
Transport	HTTPS only. Requests on plain HTTP are not served.

GET `/v1/ping` — **Ping with a missing or unknown key**

HEADERS

```
(none)
```

RESPONSE **401**

```
{
  "detail": "missing or unknown API key; send it as 'Authorization: Bearer geac_...'"
}
```

Next: Fix the header or the key. Do not retry in a loop: an inactive (paused) account gets the same answer. Note the 401 body uses the detail shape, not status/code/reason.

GET /v1/ping

Confirms the key, returns the bot's limits and today's counts, and returns the server clock so you can check your `sent_at` alignment. Pings are not rate limited and are **not recorded**: they do not count as activity and do not appear on the owner's dashboard.

```
curl https://geacsystems.com/v1/ping \
  -H "Authorization: Bearer $GEAC_API_KEY"
```

GET `/v1/ping` — **Authenticated ping**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
```

RESPONSE **200**

```
{
  "status": "ok",
  "bot": "Example Bot",
  "owner": "A. Developer",
  "cohort": "provisional",
  "limits": {
    "ideas_per_day": 50,
    "open_ideas": 100,
    "requests_per_minute": 60,
    "requests_per_hour": 500,
    "max_signal_age_s": 300,
    "minimum_hold_minutes": 15,
    "review_sessions": 120
  },
  "today": {
    "new_ideas": 0,
    "messages": 0
  },
  "open_ideas": 0,
  "server_time": "2026-11-03T14:30:58+00:00"
}
```

Next: The key works. Read limits and server_time, compare server_time with your clock, then send a test new.

Field	Meaning
`cohort`	`provisional` or `established`. See Accounts.
`limits.ideas_per_day`	New ideas accepted per UTC calendar day (default 50; the operator may set 1–200 for a bot).
`limits.open_ideas`	Maximum open ideas at once (100).
`limits.requests_per_minute`, `requests_per_hour`	Rate limits on `POST /v1/ideas` (60 per rolling minute, 500 per rolling hour).
`limits.max_signal_age_s`	Oldest `sent_at` accepted, in seconds (300).

Field	Meaning
`limits.minimum_hold_minutes`, `review_sessions`	15-minute minimum hold after the entry bar; automatic review after 120 trading sessions.
`today`	`new_ideas`: new ideas accepted so far on the current UTC date. `messages` GAP currently always reads 0 because of a counting fault; do not rely on it until the change log says it is fixed.
`open_ideas`	The bot's open ideas right now. This count is the only state the API exposes (see State and recovery).
`server_time`	The server's UTC clock. If it differs from yours by more than a few seconds, fix your clock before sending ideas.

POST /v1/ideas

One JSON object per request, describing one instruction. `event` says which: `new` opens an idea, `amend` changes its conviction, `close` closes it. GEAC sets the size and the price; never send either. A reversal is two messages: a `close`, then a `new` on the other side.

Processing order

Checks run in this order and stop at the first failure. The order matters for retries: a message is only **recorded against its `client_ref`** from step 6 onward, and the duplicate check at step 5 runs before the timestamp is examined.

#	Check	Failure	Recorded?
1	API key	`401`	no
2	Rate limit (60 per minute, 500 per hour, per bot)	`429 rate_limited`, header `Retry-After`	no
3	Body size ≤ 64 KB; body is one JSON object	`413 too_large`, `400 bad_json`	no
4	`event` and `client_ref` present and valid	`400 missing_field` / `bad_field`	no
5	**Duplicate check** on (bot, `client_ref`)	Earlier accepted → `200 duplicate`. Earlier refused → the original status with code `repeat_of_refused`. The new payload is ignored either way.	—
6	`sent_at` parses, is ≤ 300 s old and ≤ 120 s ahead of server time	`400 bad_field`, `422 stale`, `422 clock`	yes
7	Permission (cannot fail for an active bot today)	`403 not_allowed`	yes
8	Event rules (universe, idea state, limits, field values)	`400` / `422` with a specific code	yes
9	Accepted and written to the ledger	`200 accepted`	yes

GEAC's clock at receipt governs everything: pricing, the age check and `received_at`. Your `sent_at` is used only for the age and skew checks.

Fields common to every message

Field	Required	Values	Notes
`event`	yes	`new`, `amend`, `close` (any case)	
`client_ref`	yes	1–64 characters	Your reference for this message, unique per bot, kept indefinitely. Repeating it returns the first outcome; see Idempotency and retries.

Field	Required	Values	Notes
`sent_at`	yes	ISO 8601 with timezone, e.g. `2026-11-03T14:31:05Z`	Refused if more than 300 s old (`stale`) or more than 120 s ahead (`clock`). A value without a timezone is a `bad_field` .

Unknown fields are ignored. `horizon_days` , from version 1.0, is accepted and ignored: ideas no longer carry a horizon.

new: open an idea

Field	Required	Values	Notes
`symbol`	yes	ticker, ≤ 12 characters, any case	Must be in the GEAC universe (Universe). Share classes use a hyphen: `BRK-B` , `BF-B` .
`side`	yes	`buy` , `sell` (any case)	Returned upper-cased as `BUY` / `SELL` .
`conviction`	yes	integer 1–4	Low, Medium, High, Max. Sets the size GEAC assigns (Fields and scoring). The string `"3"` is accepted; send an integer.
`hedge`	yes	`sector` , `market` , or a ticker	A ticker must be in the universe and must not be the idea itself.
`styles`	no	list, or comma-separated string, from `catalyst` , `earnings` , `valuation` , `technical` , `macro` , `research` , `news` , `other`	Any other value is a `bad_field` .
`thesis`	no	text, ≤ 2000 characters	If present it must be at least 10 words.
`target_pct` , `stop_pct`	no	numbers with 0 < |value| ≤ 300	Descriptive only: stored and shown, never acted on. GEAC does not execute stops or notify the bot. The sign is not interpreted.

Rules checked, in order: the name is in the universe (`422 universe`); then the custom hedge is valid, the account is active, the bot has no open idea in the name ("you already have an open idea in X; change or close it instead"), a `sell` has shares available to borrow ("no shares of X are available to borrow (Interactive Brokers, date), so a short cannot be opened"; a name with no borrow rate on file is accepted), the daily limit is not reached, and the open cap of 100 is not reached (all `422 rule`).

Shorts pay to borrow the stock (from 17 November 2026): a `sell` idea is charged the stock-borrow fee, fee/360 per calendar day on the market value of the short, from Interactive Brokers' indicative daily rate. The charge reduces the idea's residual return each day. Longs are not charged.

POST `/v1/ideas` — Open a long (BUY) idea

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0001",
  "sent_at": "2026-11-03T14:31:05Z",
  "symbol": "MSFT",
  "side": "buy",
  "conviction": 3,
  "hedge": "sector",
  "styles": [
    "technical",
    "news"
  ],
  "thesis": "Example thesis: price has crossed above its 20 and 50 day averages after a
cloud revenue update.",
  "target_pct": 8,
  "stop_pct": 4
}
```

RESPONSE **200**

```
{
  "status": "accepted",
  "event": "new",
  "idea_id": "i_1a2b3c4d5e6f",
  "client_ref": "bot1-2026-11-03-0001",
  "symbol": "MSFT",
  "side": "BUY",
  "received_at": "2026-11-03T14:31:05.412+00:00",
  "review_date": "2027-04-26",
  "pricing": "next 5-minute VWAP after receipt",
  "minimum_hold": "15 minutes",
  "earliest_close": "2026-11-03T14:50:00+00:00"
}
```

Next: Store idea_id against your client_ref. Entry bar: 09:35-09:40 New York (14:35-14:40 UTC). The idea may be closed from earliest_close (09:50 New York); it may be amended at any time.

POST `/v1/ideas` — **Open a short (SELL) idea with a market hedge**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0002",
  "sent_at": "2026-11-03T14:32:40Z",
  "symbol": "TSLA",
  "side": "sell",
  "conviction": 2,
  "hedge": "market",
  "styles": "valuation"
}
```

RESPONSE **200**

```
{
  "status": "accepted",
  "event": "new",
  "idea_id": "i_7f8e9d0c1b2a",
  "client_ref": "bot1-2026-11-03-0002",
  "symbol": "TSLA",
  "side": "SELL",
  "received_at": "2026-11-03T14:32:40.087+00:00",
  "review_date": "2027-04-26",
  "pricing": "next 5-minute VWAP after receipt",
  "minimum_hold": "15 minutes",
  "earliest_close": "2026-11-03T14:50:00+00:00"
}
```

Next: A SELL idea is a separate position. It does not close a BUY in the same name; one open idea per name per bot, so a SELL while a BUY is open would be refused.

Response field	Meaning
`idea_id`	GEAC's identifier (`i_` + 12 hex characters). Store it with your `client_ref` ; you need it to amend or close by id.
`received_at`	Server time of receipt. This fixes the entry bar (Pricing lifecycle).
`earliest_close`	Start of the entry bar plus 15 minutes. A `close` before this is refused.
`review_date`	The date on which the idea is closed automatically if still open: receipt + 174 calendar days (about 120 trading sessions). The code contains a reset to the exact 120th session once the entry is priced, but it does not take effect in practice, so treat the returned date as final unless the idea is reconfirmed.
`pricing` , `minimum_hold`	Fixed descriptive strings. `pricing` reads "next 5-minute VWAP after receipt", which is what is computed since 30 October 2026; see Pricing lifecycle.

amend: change conviction

Identify the idea by `idea_id`, or by `symbol` with an optional `side`. Send the new `conviction` (required). Only conviction can change through the API; thesis, target and stop change only in the web app. Amending does not move the review date and does not re-price the idea.

POST `/v1/ideas` — **Amend conviction**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "amend",
  "client_ref": "bot1-2026-11-03-0003",
  "sent_at": "2026-11-03T15:05:12Z",
  "idea_id": "i_1a2b3c4d5e6f",
  "conviction": 4
}
```

RESPONSE **200**

```
{
  "status": "accepted",
  "event": "amend",
  "idea_id": "i_1a2b3c4d5e6f",
  "client_ref": "bot1-2026-11-03-0003",
  "symbol": "MSFT",
  "side": "BUY",
  "conviction": 4,
  "review_date": "2027-04-26"
}
```

Next: Only conviction can change through the API. Amending does not move the review date and does not re-price the idea.

Refusal	When
`422 unknown_idea`	`idea_id` does not belong to this bot.
`422 not_open`	The idea is closed, or no open idea exists in the symbol.
`422 side_mismatch`	`side` was given and disagrees with the open idea.
`422 universe`	The name has left the universe: the idea is close-only.
`422 rule`	The conviction is unchanged ("nothing changed").
`400 missing_field`	`conviction` missing.

close: close an idea

Identify the idea as for an amend and give a `reason`: `thesis_realised`, `thesis_wrong`, `price`, `expiry` or `compliance` (anything else is a `400 bad_field`). A close is accepted whether or not the entry has

been priced yet; the entry is still priced when its bar closes, so a same-day round trip is valued (Pricing lifecycle).

POST `/v1/ideas` **Close the long idea**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "close",
  "client_ref": "bot1-2026-11-03-0004",
  "sent_at": "2026-11-03T19:40:02Z",
  "idea_id": "i_1a2b3c4d5e6f",
  "reason": "thesis_realised"
}
```

RESPONSE **200**

```
{
  "status": "accepted",
  "event": "close",
  "idea_id": "i_1a2b3c4d5e6f",
  "client_ref": "bot1-2026-11-03-0004",
  "symbol": "MSFT",
  "side": "BUY",
  "closed_at": "2026-11-03T19:40:02.655+00:00",
  "reason": "thesis_realised"
}
```

Next: closed_at is the acceptance time. The idea is valued at the official close of that date (16:00 New York) by the nightly job. A new idea in MSFT may be opened at once.

`closed_at` is the time GEAC accepted the close. The idea is valued at the official close (16:00 New York) of the UTC date of `closed_at`. A new idea in the same name, either side, may be opened immediately.

POST `/v1/ideas` **Close refused: minimum holding period**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "close",
  "client_ref": "bot1-2026-11-03-0006",
  "sent_at": "2026-11-03T14:45:30Z",
  "idea_id": "i_7f8e9d0c1b2a",
  "reason": "price"
}
```

RESPONSE **422**

```
{
  "status": "refused",
  "code": "rule",
  "client_ref": "bot1-2026-11-03-0006",
  "reason": "ideas are held for at least 15 minutes after their entry bar; TSLA can be
closed from 09:50 New York time on 3 Nov"
}
```

Next: Keep the exit intent locally, wait until earliest_close, then send the close again with a new client_ref.

The minimum-hold refusal is recorded against the reference, like every refusal from step 6 onward. Resending the same reference later returns `repeat_of_refused` with the old reason, which is easy to misread as "still too early". Keep the intention to close, wait until `earliest_close`, and send the close with a **new** `client_ref`.

Reversal: long to short

Two messages with two references. First the close (shown above, accepted at 14:40:02 New York). Then the new idea on the other side. Do not send the `new` until the close has been accepted; sent together they race, and if the `new` arrives first it is refused because an idea in the name is still open.

POST `/v1/ideas` **Reversal, step 2: open the opposite side after the close was accepted**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0005",
  "sent_at": "2026-11-03T19:40:03Z",
  "symbol": "MSFT",
  "side": "sell",
  "conviction": 2,
  "hedge": "sector"
}
```

RESPONSE **200**

```
{
  "status": "accepted",
  "event": "new",
  "idea_id": "i_3c4d5e6f7a8b",
  "client_ref": "bot1-2026-11-03-0005",
  "symbol": "MSFT",
  "side": "SELL",
  "received_at": "2026-11-03T19:40:03.120+00:00",
  "review_date": "2027-04-26",
  "pricing": "next 5-minute VWAP after receipt",
  "minimum_hold": "15 minutes",
  "earliest_close": "2026-11-03T20:00:00+00:00"
}
```

Next: Step 1 was the close above. Received at 14:40:03 New York, so the entry bar is 14:45-14:50 and earliest_close is 15:00 New York.

Duplicates

Repeating a `client_ref` never creates a second idea. The reply depends only on the first outcome; the payload of the repeat is ignored.

POST `/v1/ideas` **Repeat of an accepted reference**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0001",
  "sent_at": "2026-11-03T14:31:05Z",
  "symbol": "MSFT",
  "side": "buy",
  "conviction": 3,
  "hedge": "sector"
}
```

RESPONSE **200**

```
{
  "status": "duplicate",
  "idea_id": "i_1a2b3c4d5e6f",
  "client_ref": "bot1-2026-11-03-0001",
  "first_received_at": "2026-11-03T14:31:05.412+00:00"
}
```

Next: Nothing new was created. The reply carries idea_id and first_received_at only; earliest_close and review_date are not repeated. Resend the same reference after any unknown outcome before deciding anything.

POST `/v1/ideas` **Repeat of a refused reference**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "close",
  "client_ref": "bot1-2026-11-03-0006",
  "sent_at": "2026-11-03T14:46:00Z",
  "idea_id": "i_7f8e9d0c1b2a",
  "reason": "price"
}
```

RESPONSE **422**

```
{
  "status": "refused",
  "code": "repeat_of_refused",
  "client_ref": "bot1-2026-11-03-0006",
  "reason": "ideas are held for at least 15 minutes after their entry bar; TSLA can be
closed from 09:50 New York time on 3 Nov"
}
```

Next: The first attempt with this reference was refused (too early) and that answer is permanent for the reference. To close later, send a new close with a NEW client_ref after earliest_close.

Responses and status codes

Every refusal except `401` has one shape: `status` is `"refused"`, `code` is a short token a program can act on, `reason` is a sentence a person can read, and `client_ref` echoes yours when it was read. The same reason appears on the owner's dashboard. `401` has FastAPI's `{"detail": "…"}` shape instead.

HTTP	Meaning	Codes
200	Accepted, or a duplicate of an accepted reference	`(status accepted)`, `(status duplicate)`
400	Malformed request	`bad_json`, `missing_field`, `bad_field`
401	Missing, unknown or inactive API key (body: {"detail": ...})	—
403	Permission refused by the permissions engine (not reachable for a valid bot today)	`not_allowed`
413	Body larger than 64 KB	`too_large`
422	Rule, timing or state refusal	`stale`, `clock`, `unknown_idea`, `not_open`, `side_mismatch`, `universe`, `rule`, `repeat_of_refused`
429	Rate limit; Retry-After header in seconds	`rate_limited`
500	Server fault; nothing was recorded unless the response says so	—

Code	HTTP	Meaning	Recorded against the reference?
`bad_json`	400	Body is not a single JSON object.	no
`missing_field`	400	A required field is absent (`event`, `client_ref`, `sent_at`, `symbol`, `side`, `conviction`, `hedge`, `reason` …).	no for `event` / `client_ref`; yes otherwise
`bad_field`	400	A field has an invalid value (bad timestamp, side, conviction, style, reason, length).	no for `event` / `client_ref`; yes otherwise
—	401	Key missing, unknown, not a bot, or account paused. Body `{"detail": …}`.	no
`not_allowed`	403	Permissions engine refusal. Defined, not reachable for a valid active bot today.	yes
`too_large`	413	Body over 65,536 bytes.	no
`stale`	422	`sent_at` more than 300 s old.	yes
`clock`	422	`sent_at` more than 120 s in the future.	yes
`unknown_idea`	422	`idea_id` is not this bot's.	yes
`not_open`	422	No open idea matches.	yes
`side_mismatch`	422	Given `side` disagrees with the open idea.	yes

Code	HTTP	Meaning	Recorded against the reference?
`universe`	422	Name not eligible, or eligible no longer (close-only).	yes
`rule`	422	Minimum hold, open idea already in the name, daily limit, open cap, invalid hedge, nothing changed, account inactive.	yes
`repeat_of_refused`	original	This reference was refused before; the original reason is repeated.	already
`rate_limited`	429	Too many requests; `Retry-After` gives the wait in seconds.	no

POST `/v1/ideas` **Refused: stale signal**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0007",
  "sent_at": "2026-11-03T14:20:00Z",
  "symbol": "AAPL",
  "side": "buy",
  "conviction": 2,
  "hedge": "sector"
}
```

RESPONSE **422**

```
{
  "status": "refused",
  "code": "stale",
  "client_ref": "bot1-2026-11-03-0007",
  "reason": "'sent_at' is 661 seconds old; signals older than 300 seconds are refused"
}
```

Next: The entry signal is stale: discard it. Do not resend the same message; a fresh decision needs a fresh sent_at and a new client_ref.

POST `/v1/ideas` **Refused: clock ahead of the server**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0008",
  "sent_at": "2026-11-03T14:40:00Z",
  "symbol": "AAPL",
  "side": "buy",
  "conviction": 2,
  "hedge": "sector"
}
```

RESPONSE **422**

```
{
  "status": "refused",
  "code": "clock",
  "client_ref": "bot1-2026-11-03-0008",
  "reason": "'sent_at' is 535 seconds in the future; check the bot's clock"
}
```

Next: Synchronise the clock (NTP) and compare with server_time from /v1/ping before sending again with a new client_ref.

POST `/v1/ideas` **Refused: name not eligible**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0009",
  "sent_at": "2026-11-03T14:41:00Z",
  "symbol": "TINY",
  "side": "buy",
  "conviction": 2,
  "hedge": "sector"
}
```

RESPONSE **422**

```
{
  "status": "refused",
  "code": "universe",
  "client_ref": "bot1-2026-11-03-0009",
  "reason": "TINY is not in the GEAC universe (US-listed stocks (NYSE or Nasdaq, any
domicile, excluding ADRs of foreign companies) with a market value above $100m and more than
$1m traded a day, plus every S&P 500 and Nasdaq-100 member)"
}
```

Next: Remove the name from the strategy's list. If the name left the universe while you held it, the reason says so and only a close is accepted.

POST `/v1/ideas` — **Refused: an idea in the name is already open**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0010",
  "sent_at": "2026-11-03T14:42:00Z",
  "symbol": "TSLA",
  "side": "buy",
  "conviction": 2,
  "hedge": "sector"
}
```

RESPONSE **422**

```
{
  "status": "refused",
  "code": "rule",
  "client_ref": "bot1-2026-11-03-0010",
  "reason": "you already have an open idea in TSLA; change or close it instead"
}
```

Next: To reverse, close the open SELL first, then open the BUY with another reference.

POST `/v1/ideas` **Refused: nothing to close**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "close",
  "client_ref": "bot1-2026-11-03-0011",
  "sent_at": "2026-11-03T19:41:00Z",
  "symbol": "NVDA",
  "reason": "price"
}
```

RESPONSE **422**

```
{
  "status": "refused",
  "code": "not_open",
  "client_ref": "bot1-2026-11-03-0011",
  "reason": "this bot has no open idea in NVDA"
}
```

Next: Your local state disagrees with GEAC's. Reconcile: replay the client_refs of your unresolved messages (see Operate reliably) before acting.

POST `/v1/ideas` — **Refused: a required field is missing**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0012",
  "sent_at": "2026-11-03T14:43:00Z",
  "symbol": "AAPL",
  "side": "buy",
  "conviction": 2
}
```

RESPONSE **400**

```
{
  "status": "refused",
  "code": "missing_field",
  "client_ref": "bot1-2026-11-03-0012",
  "reason": "'hedge' is required"
}
```

Next: Fix the request and send it with a NEW client_ref: this refusal is recorded against bot1-2026-11-03-0012.

POST `/v1/ideas` — **Refused: body is not a JSON object**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY (RAW)

```
not json
```

RESPONSE **400**

```
{
  "status": "refused",
  "code": "bad_json",
  "reason": "the body must be a single JSON object"
}
```

Next: Not recorded (no reference was read). Fix the serialisation and resend.

POST `/v1/ideas` **Rate limited**

HEADERS

```
Authorization: Bearer $GEAC_API_KEY
Content-Type: application/json
```

REQUEST BODY

```
{
  "event": "new",
  "client_ref": "bot1-2026-11-03-0013",
  "sent_at": "2026-11-03T14:44:00Z",
  "symbol": "AAPL",
  "side": "buy",
  "conviction": 2,
  "hedge": "sector"
}
```

RESPONSE **429** · **`RETRY-AFTER: 17`**

```
{
  "status": "refused",
  "code": "rate_limited",
  "reason": "too many requests; retry in 17 seconds"
}
```

Next: Not recorded against the reference. Wait Retry-After seconds, then resend the SAME message and reference; if sent_at would then be older than 300 seconds, re-decide and use a new reference.

Idempotency and retries

The `client_ref` is your idempotency key. GEAC answers a repeated reference with the first outcome, for ever, whatever the body. Two consequences shape every retry decision:

- Resending the **same** reference is always safe. It can never create a second idea or a second close.
- Once a reference is **recorded as refused**, it stays refused. Fixing the problem means sending a **new** reference.

The golden rule for an unknown outcome: **never generate a new reference until the original outcome is known**. Resend the original reference first and read the answer. Because the duplicate check runs before the timestamp check, the replay always resolves the question: `200 duplicate` means the first attempt was accepted; `repeat_of_refused` means it was refused; `200 accepted` means it never arrived and has now been accepted; `422 stale` means it never arrived and the signal is now too old, so nothing exists and you may decide afresh.

Outcome	Was the message recorded?	Resend the same reference?	What to do
Connection failed before the request was sent (DNS, TLS, refused)	no	yes, at once (back off after repeated failures)	Nothing reached GEAC. Retry with the same reference while `sent_at` is under 300 s old; after that, re-decide.
Timeout or dropped connection after the request was sent	unknown	**yes, this is how you find out**	Mark the reference *unknown* locally. Resend it with the identical body and read the reply as described above. Only then act.
`200 accepted`	yes	harmless	Store `idea_id`, `earliest_close`, `review_date` against the reference.
`200 duplicate`	yes (earlier)	harmless	Your earlier attempt succeeded. Reconcile local state from `idea_id` and `first_received_at`; `earliest_close` is not repeated, so compute it (entry bar + 15 minutes) or wait a safe margin.
`400 bad_json`, `400` on `event` / `client_ref`	no	n/a	Fix the serialisation or the field; the reference is still free.
`400` `missing_field` / `bad_field` on any other field	yes	returns `repeat_of_refused`	Fix the message and send it with a **new** reference.
`401`	no	no	Stop and alert a person. Check the header, the key, and whether the account was paused. Do not loop.
`403 not_allowed`	yes	no	Stop and contact ops@geacsystems.com with the reference.

Outcome	Was the message recorded?	Resend the same reference?	What to do
`413 too_large`	no	n/a	Shorten the thesis; the reference is still free.
`422 stale`	yes	no	Discard the signal. A fresh decision needs a fresh `sent_at` and a new reference. If this happens often, your pipeline is too slow or your clock is behind.
`422 clock`	yes	no	Synchronise the clock (NTP), compare with `server_time` from `/v1/ping`, then re-decide with a new reference.
`422 rule` on a `close` (minimum hold)	yes	returns `repeat_of_refused`	Keep the exit intent. At `earliest_close`, send the close with a new reference. See the worked workflow below.
`422 rule` (open idea in the name, daily limit, open cap, hedge, nothing changed)	yes	returns `repeat_of_refused`	Act on the reason: close first, wait for the UTC day to roll, close something, change the hedge. New reference when you retry.
`422 universe`	yes	returns `repeat_of_refused`	Drop the name from the strategy. If you hold it, only a close is accepted.
`422 not_open`, `unknown_idea`, `side_mismatch`	yes	returns `repeat_of_refused`	Your state disagrees with GEAC's. Reconcile every *unknown* reference first, then re-derive your positions before acting.
`422 repeat_of_refused`	already	pointless	You reused a refused reference. Read the original reason and, if the action is still wanted, send it under a new reference.
`429 rate_limited`	no	**yes**, after `Retry-After`	Wait the stated seconds and resend the same message and reference. If `sent_at` would then be over 300 s old, re-decide with a fresh timestamp and a new reference.
`5xx` or an unreadable response	unknown	yes	Treat as a timeout: mark unknown, resend the same reference, read the answer.

Worked workflow: a close refused for the minimum hold

Time (New York)	Bot	GEAC
09:32:40	`new` SELL TSLA, ref `…-0002`	`200 accepted`; entry bar 09:35, `earliest_close` 09:50

Time (New York)	Bot	GEAC
09:45:30	`close`, ref `…−0006`	`422 rule` "can be closed from 09:50". **Recorded** against `…−0006`.
09:46:00	Retries `…−0006` (wrong)	`422 repeat_of_refused`, same reason. Still refused at 10:00 and at any later time.
09:50:05	`close`, **new** ref `…−0014`	`200 accepted`, `closed_at` 09:50:05, valued at the day's close

Worked workflow: a lost response

Time (New York)	Bot	GEAC
09:31:05	Writes ref `…−0001` to its own store as *pending*, then sends `new` BUY MSFT	Accepts and records it. The response is lost on the network.
09:31:35	Timeout. Marks `…−0001` *unknown*. Does **not** create `…−0002` for the same signal.	
09:31:36	Resends the identical message with ref `…−0001`	`200 duplicate`, `idea_id i_1a2b…`, `first_received_at` 09:31:05
09:31:36	Marks `…−0001` *accepted*; records the position; computes `earliest_close` 09:50 from the entry bar	

Had the reply been `200 accepted`, the first attempt never arrived and the idea is now open. Had it been `422 stale`, the first attempt never arrived and nothing exists.

Pricing and valuation lifecycle

This section describes what GEAC computes today. Where the implementation differs from earlier published text, the difference is stated here and listed in the compatibility notes.

1. The entry bar is fixed at receipt

When a `new` is accepted, GEAC converts its receipt time to New York time and fixes a five-minute entry bar:

Received (New York time)	Entry bar	`earliest_close`
Weekday, 09:30:00 to 15:54:59	The next five-minute boundary **strictly after** receipt: 10:02:17 → 10:05–10:10 bar; 10:05:00 exactly → 10:10 bar; 10:09:59 → 10:10 bar	bar start + 15 min (10:20 for the 10:05 bar)
Weekday, before 09:30 (overnight and pre-market)	That day's 09:30–09:35 bar	09:45 that day
Weekday, 15:55:00 or later (including after hours)	The next weekday's 09:30 bar	09:45 the next weekday
Saturday or Sunday	Monday's 09:30 bar	09:45 Monday

The bar never moves afterwards. The minimum-hold clock starts at the **start** of the bar, so an idea received at 10:02 may be closed from 10:20, and one received at 17:00 may be closed from 09:45 the next trading day.

2. The entry is priced live, from the bar's VWAP

The entry is priced as soon as its bar has closed. From 90 seconds after the bar's end (10:11:30 for the 10:05 bar) GEAC computes the **volume-weighted average price of the bar** from its five one-minute bars (sum of each minute's typical price × volume ÷ total volume) and records it; the website shows the price and its basis immediately. The **hedge leg** (the S&P 500 or the sector index) is priced at the same bar by the same method, so entry and hedge are measured at the same moment. The response string `"pricing": "next 5-minute VWAP after receipt"` describes this rule.

Two fallbacks exist for vendor gaps. If, ten minutes after the bar, the vendor has published only some of the five minutes, the VWAP is taken over the minutes that exist. If no minute exists at all, the five-minute bar's typical price (high + low + close) ÷ 3 is used and the basis is recorded as such. If the index publishes no bar, its close on the entry date stands in for the hedge leg. Entries priced before 30 October 2026 were priced that evening at the typical price, with the hedge leg at the entry date's close; they keep those values.

Until the bar has closed and the price is written, the entry is *pending* (at most a few minutes during the session; overnight for ideas received after 15:55).

3. Marks are daily, at the close

Each nightly writes one mark per trading day from the entry date to the close date, using end-of-day closes of the stock and its hedge. Residual return = stock return − beta × hedge return − a transaction-

cost allowance in basis points that depends on the name's liquidity, with the sign flipped for SELL ideas. Net P&L in US dollars = residual return × the idea's notional size, which is the stock leg plus the hedge leg less costs. Contributors are ranked and paid on net P&L. A change of conviction changes the size from the next session.

4. A close is valued at that day's close

A close is not priced at a five-minute bar. The final mark is the official close (16:00 New York) of the UTC date of acceptance. A close accepted at 10:20 New York is valued at 16:00 that day, so exits are asymmetric with entries: the entry is intraday, the exit is end-of-day. A close accepted at or after 20:00 New York falls on the next UTC date and is therefore valued at the **next trading day's** close.

> **Same-day round trips are valued.** A close may be accepted before the entry has been priced (the bar still open, or inside the 90-second lag). The entry is still priced when its bar closes, and the idea gets one mark at that day's official close. Before 30 October 2026 such entries were never priced; that gap is closed.

5. Review after 120 sessions

An idea still open on its `review_date` (receipt + 174 calendar days, about 120 trading sessions; see the field notes) is closed by the nightly with reason `expiry` and valued at that day's close. Reconfirm, which pushes the date out by 174 calendar days, exists only in the web app; there is no API event for it.

Worked timelines (illustrative, Tuesday 2026-11-03)

A. Normal intraday entry, exit days later

When (New York)	Event	Effect
Tue 10:02:17	`new` BUY accepted	Entry bar 10:05–10:10. `earliest_close` 10:20. `review_date` = receipt + 174 days.
Tue 10:11:30	Live pricing	Entry priced at the VWAP of the 10:05 bar; hedge leg at the index's VWAP of the same bar. Visible on the site at once.
Tue 18:30	Nightly job	First daily mark at Tue 16:00 close.
Wed, Thu 18:30	Nightly job	One mark per day at each close.
Fri 11:12:40	`close` accepted	`closed_at` Fri 11:12:40. Idea closed.
Fri 18:30	Nightly job	Final mark at Fri 16:00 close. Net P&L final.

B. Closing bell: two receipts 2½ minutes apart

When (New York)	Event	Effect
Tue 15:54:59	`new` accepted	Entry bar 15:55–16:00 (the last bar of the session). `earliest_close` 16:10, after the market has closed; the entry is priced at about 16:01:30 from the VWAP of the 15:55 bar, and a close at 16:10 is accepted and valued at Tue 16:00.

When (New York)	Event	Effect
Tue 15:57:30	`new` accepted	At or after 15:55, so the entry bar is **Wednesday 09:30–09:35**. `earliest_close` Wed 09:45. Priced Wed at about 09:36:30. The idea carries overnight exposure the bot did not choose; send the `new` before 15:55 or hold it until the morning.

C. Evening and weekend receipts

When (New York)	Event	Effect
Fri 17:00	`new` accepted	Entry bar Mon 09:30. `earliest_close` Mon 09:45.
Sat 11:00	`new` accepted	Entry bar Mon 09:30. Same result.
Mon 20:30 (01:30 UTC Tue)	`close` accepted	`closed_at` carries Tuesday's UTC date, so the idea is valued at **Tuesday's** 16:00 close, not Monday's.

D. Early close and market holiday

> **Gap: the entry-bar rule does not know the holiday calendar.** On Friday 27 November 2026 the market closes at 13:00. A `new` received at 13:20 that day gets a 13:25 bar that never trades; a `new` received on Thursday 26 November (Thanksgiving) gets a bar on a day with no session. Such entries stay *pending* indefinitely, are retried by the live pricer that day and by every nightly, and are never priced. Until this is fixed, do not send `new` ideas after 12:55 New York on an early-close day or at any time on a market holiday; hold them for the next session's open. The bot has no way to obtain the holiday calendar from GEAC (see Universe and calendar).

Universe and calendar

Eligible names

Stocks listed on the NYSE or Nasdaq, of any domicile, excluding ADRs of foreign companies, with a market value above $100m and more than $1m traded a day, plus every S&P 500 and Nasdaq-100 member. About 3,000 names at the 16 November 2026 review (the floor was $2bn and $20m until then). A name that does not trade in the five-minute bar after your message is priced at the first later bar that does. The screen is re-run on the first trading night of each month (and on demand by the operator). A current member stays in until it falls below $75m *and/or* $0.75m, so names do not flicker at the boundary.

Situation	Verified behaviour
Name never eligible	`new` → `422 universe` "X is not in the GEAC universe (…rule…)".
Name leaves while you hold it	The idea becomes **close-only**. `new` and `amend` → `422 universe` "X left the GEAC universe on YYYY-MM-DD; …". `close` is accepted. Marks continue until you close. Nothing is sent to the bot or its owner by email; the owner's web dashboard highlights the idea as close-only.
Custom hedge ticker	Must itself be in the universe, and not the idea's own symbol.
Symbol spelling	GEAC uses its market-data vendor's tickers: share classes with a hyphen (`BRK–B` , `BF–B`), no exchange suffix, no dots.

> **Gap: no public list of eligible symbols and no calendar endpoint.** The only lookup is the search box in the owner's web app. A bot cannot download the current universe, be told when a name joins or leaves, or fetch the trading calendar, session times, holidays or early closes. Until then: maintain your own symbol list, treat `422 universe` as the signal to remove a name, and use a reputable exchange calendar for holidays and early closes. See State and recovery for the proposed contract.

Calendar facts a bot must know

Item	Verified behaviour
Regular session	09:30–16:00 New York. Entry bars are placed only inside it (see Pricing). The extended session is never used for pricing.
Trading-day count	The S&P 500's end-of-day history is the calendar for the 120-session review.
Daily limit reset	New ideas per day are counted on the **UTC** date of acceptance, so the counter resets at 19:00 or 20:00 New York, not at midnight New York.
Submission hours	Messages are accepted around the clock; the entry-bar rule decides when they price.

Limits and clocks

Limit	Value	Reset or scope
New ideas per day	50 (operator may set 1–200 per bot)	UTC calendar date of acceptance
Open ideas	100 per bot	—
Open ideas per name	1 per bot	—
Requests to `/v1/ideas`	60 per rolling minute, 500 per rolling hour	per bot; counters are in server memory and reset on a deploy
Requests to `/v1/ping`	unlimited	not recorded
Body size	64 KB	—
`sent_at`	≤ 300 s old, ≤ 120 s ahead	server UTC
`client_ref`	1–64 characters, unique per bot	kept indefinitely
`symbol`	≤ 12 characters	—
`thesis`	≤ 2000 characters, ≥ 10 words if present	—
Minimum hold	15 minutes from the start of the entry bar	—
Review	120 trading sessions after the entry date	reconfirm in the web app only

Fields and scoring

Field	How GEAC uses it
`conviction` 1–4	Sets the notional size GEAC assigns to the idea in its paper book: Low $1.3m, Medium $2.65m, High $4m, Max $6m, each capped at 10% of the name's average daily dollar volume (floor $100k). The size is also scaled by the stock's volatility (smaller for a more volatile name, larger for a steadier one, between 0.15 and 2 times the ladder), so that a dollar of P&L carries about the same risk in every name. A higher conviction is a bigger size: it earns more when right and costs more when wrong.
`hedge`	`sector` : the name's S&P 500 sector index, scaled by the name's beta to that index, which GEAC measures each month from daily closes and holds between 0.3 and 2.5. `market` : the S&P 500, beta-scaled likewise. A ticker: a dollar-neutral single-name hedge at beta 1.
`styles`	Classification tags shown with the idea and used in analytics. No effect on scoring.
`thesis`	Shown with the idea on the owner's dashboard and to the operator. No effect on scoring.
`target_pct` , `stop_pct`	Stored and displayed only. Nothing closes the idea when a level is reached and nothing notifies the bot. Your bot must send the `close` itself.
Net P&L	Residual return (after hedge, the cost allowance and any borrow fee) times the idea's notional size, in US dollars: the stock leg plus the hedge leg, less costs. Not capped: a large gain or loss counts in full. It is paper P&L on a notional size, not money owed. Bots are ranked in their own league by net P&L over the selected period.

The response's `minimum_hold` and `pricing` strings are descriptive constants. The ledger behind every submission, valuation and closure is append-only and hash-chained; history is never edited or backdated.

State and recovery

What a bot can learn from GEAC about its own state today, and how to recover after a crash or a lost response.

Question	Answer today
How many ideas do I have open?	`open_ideas` in `/v1/ping`. A count only.
Which ideas are open, and since when?	GAP No endpoint. Only your own records, or the owner's web dashboard (a person signs in).
What happened to message `client_ref` X?	Resend X with the identical body: `200 duplicate` (accepted, with `idea_id` and `first_received_at`), `repeat_of_refused` (refused, with the reason), or a fresh outcome (it never arrived). This is the only lookup and it is reliable, provided you still hold the reference.
What is idea `idea_id` Y's state, entry price or value?	GAP No endpoint. The owner's dashboard shows it.
Was my idea closed at review, or because the name left the universe?	GAP No endpoint, no webhook and no email. The owner's dashboard highlights the idea; the bot finds out when its next `amend` or `close` returns `not_open` or `universe`.
Which names are eligible?	GAP No endpoint (see Universe).

> **High-priority gap.** Without a read API, a bot that loses its local store cannot rebuild its positions from GEAC, and a bot cannot notice an automatic close until it next tries to act on the idea. The reference client works within this constraint; the proposed contract below would remove it.

What your bot must persist (today)

1. Every `client_ref` you have ever generated, with its full request body and its status: *pending* (written before sending), *accepted*, *refused* or *unknown*. Write the record **before** the request leaves the process.
2. For each accepted `new`: `idea_id`, symbol, side, `received_at`, `earliest_close`, `review_date`.
3. For each accepted `close`: `idea_id` and `closed_at`.

On restart: replay every *pending* or *unknown* reference and record the outcome; then rebuild open positions as accepted `new`s without an accepted `close`; then compare the count with `open_ideas` from `/v1/ping`. A mismatch means an automatic close (review or universe) or a close made by a person in the web app; ask the owner to check the dashboard.

Proposed read-only contract PROPOSED · NOT IMPLEMENTED

The minimal set that would close the gap, offered for the operator's decision. None of these paths exists today; calling them returns `404`.

Proposed call	Would return	Notes
`GET /v1/ideas?status=open` (also `closed`, with `since=`)	This bot's ideas: `idea_id`, `symbol`, `side`, `conviction`, `received_at`, `entry_status` (`pending` / `priced`), `earliest_close`, `review_date`, `close_only`, and for closed ideas `closed_at`, `close_reason`, `alpha_units` once final.	Read-only; exposes what the owner's dashboard already shows. No new trading rule.
`GET /v1/ideas/{idea_id}`	One idea with the same fields.	
`GET /v1/messages/{client_ref}`	The recorded outcome of one of this bot's messages: status, HTTP code, `code`, `reason`, `idea_id`, `received_at`.	Lets a bot resolve an unknown outcome without re-posting.
`GET /v1/universe`	The current eligible symbols and the rule text, with the review date.	Pairs with the monthly review.
`GET /v1/calendar?from=&to=`	Trading days, early closes and holidays GEAC will use for pricing.	Only meaningful once the entry-bar rule honours that calendar.

Accounts and strategy lifecycle

Stage	Verified behaviour
Apply	At geacsystems.com/apply, choose "Automated strategy" and give the owner's name, email and firm, the bot's name and a description of the strategy, and accept the declarations. Each bot has a named owner responsible for its activity. One bot is one strategy; related bots must be disclosed and are treated as a family.
Approval	The operator reviews the application and emails the decision. Timing is a human process, not a fixed SLA.
Key	The owner signs in and creates the key under Profile → Bot Connection (see Authentication).
Provisional period	Three calendar months from the joining date, as implemented. Ideas are priced, marked and shown in the league but earn nothing. Earlier published text described this as "the first full calendar quarter"; the two can differ by up to three months. The operator's decision is on the backlog; until then the implementation governs.
Ranking	Bots are ranked separately from people by net P&L in US dollars. Earlier text required "20 priced ideas per quarter" to be ranked; no such threshold is implemented, and the statement has been withdrawn pending the operator's decision. A quarterly pool pays only when it holds at least 12 ideas across 6 names.
Changing the strategy	Performance history is permanent. A materially different strategy needs a new bot account, declared as related to the original.
Inactivity and pauses	Operator policy, applied by hand: a bot may be marked inactive after 10 trading days without submissions and paused after 30; repeated authentication failures, more than 10% malformed messages over three days, three rule breaches in a quarter, restricted-list matches or suspicious activity around announcements may lead to a pause. Nothing in the software pauses a bot automatically. Pings are not recorded and do not count as activity; only recorded messages do.
Paused or removed	Every request returns `401`. Open ideas remain and continue to be marked; the owner should close them in the web app or ask ops@geacsystems.com. Decisions are logged and communicated to the owner.
Owner's view	The web app's Bot Connection page shows the key prefix, limits, today's counts, open-idea count and the last 50 messages with their outcomes and reasons. It is a dashboard for a person, not an API.

Test and launch

> **There is no test environment.** `/v1/ping` is safe to call at any time: it changes nothing and is not recorded. Every accepted `POST /v1/ideas` is a production idea in your bot's permanent record, counts towards your daily limit, and can only be closed, never deleted. Do not describe production as a sandbox to your team.

Recommended path

1. **Dry run offline.** The reference client ships with an in-process fake of this API that follows every rule on this page (processing order, duplicates, minimum hold, rate limits, `401`). Run your strategy against it until the message log looks right.
2. **Prove the key.** `GET /v1/ping` with the real key from an environment variable. Compare `server_time` with your clock.
3. **Prove the loop with one idea.** Send one real `new` at low conviction, then an `amend`, then a `close` after `earliest_close`. Confirm the owner sees three messages on the dashboard. This is production: pick a name you are content to have on your record.
4. **Prove recovery.** Kill the process after a `new` and before its response is stored; restart; confirm the replay returns `200 duplicate` and no second idea exists (`open_ideas` in `/v1/ping` unchanged).
5. **Go live** with alerts on `401`, on repeated `stale` and `clock`, and on any `not_open` or `unknown_idea`, which mean your state and GEAC's have diverged.

Reference client

A small Python client (standard library only) is published with this documentation in the project's `examples/geac_client/` directory: dry-run by default, key from `GEAC_API_KEY`, live mode only with two explicit environment flags, a SQLite store of every reference written **before** the request is sent, the retry table above encoded as code, `earliest_close` respected locally, and a `reconcile()` that replays unknown references. Its tests use the examples on this page as fixtures. A separate, optional file wires a simple triple-moving-average signal to the client as a demonstration of plumbing; it is not a strategy recommendation and makes no performance claim.

Newcomer acceptance scenario

A developer who has never seen GEAC should be able to, from this page alone: obtain a key (after approval), ping, submit a long and a short, amend, close after the minimum hold, reverse, handle a refusal, recover from a lost response without duplicating, and restart without losing positions. Everything on that list is achievable today except learning from GEAC which ideas are open after a lost local store, which is the gap in State and recovery.

Change log and compatibility

Version	Date	Changes
Universe and borrow	16 Nov 2026	No change to requests or responses. **Universe** floor lowered from $2bn / $20m to $100m market value and $1m traded a day (about 3,000 names). **Shorts** record the stock-borrow fee at entry, are refused (`422 rule`) when no shares are available to borrow, and are charged the fee daily from 17 Nov 2026. A name with no trade in its entry bar is priced at the first later bar that trades.
Scoring note	17 Nov 2026	No change to requests or responses. **Dollars replace alpha units** for contributors: every idea shows its stock P&L, hedge P&L and net P&L in US dollars, and the league and the quarterly pool follow net P&L. **Size**: for ideas submitted from this date the notional size is the conviction ladder scaled by the stock's volatility over the 60 sessions before the month began (between 0.15 and 2 times the ladder), then capped by liquidity as before. Ideas already open keep the size they were booked with. Alpha units remain an internal risk-adjusted measure and are no longer shown.
Scoring note	17 Nov 2026	No change to requests or responses. **Hedge ratio**: for ideas priced from this date the hedge is scaled by the stock's beta to the hedge chosen (its sector index or the S&P 500), measured by GEAC from up to two years of daily closes, pulled part-way toward 1, fixed for the calendar month and frozen on the idea at entry. Previously one vendor figure, a beta against the whole market, was used for every hedge. A stock with under about six months of history is hedged one for one. Ideas already open keep the ratio they were booked with. The ratio itself is not shown to contributors.
Scoring note	16 Nov 2026	No change to requests or responses. **Alpha units** are now weighted by conviction (Low 0.5, Medium 1, High 1.5, Max 2) and are no longer clipped at ±3, so a higher conviction earns more when right and costs more when wrong, and a large move counts in full. Each session's return carries the conviction in force for that session; an `amend` that changes conviction applies from the next session. Applied to the whole of the current quarter. Also corrected: the S&P 500 Energy sector hedge had no price from launch to 13 Nov, so energy ideas were not marked; they have been re-marked from their original entry bars.
1.1.0 (this document)	30 Oct 2026	Universe widened to US-listed stocks above $2bn / $20m plus S&P 500 and Nasdaq-100 members, monthly review, close-only exits and refusal code `universe` (30 Oct). Horizon removed, 15-minute minimum hold, five-minute entry bar, same-day re-entry allowed, `horizon_days` ignored (29 Oct). Documentation rebuilt: processing order, full status and code tables including `403` and `repeat_of_refused`, retry decision table, pricing lifecycle as implemented, gaps and proposals labelled, OpenAPI file and reference client published. **Live entry pricing** (30 Oct): entries priced from the five-minute bar's VWAP about 90 seconds after the bar closes, hedge leg at the same bar, same-day round trips valued; previously priced that evening at the bar's typical price with the hedge leg at the day's close. Withdrawn claims: "20 priced ideas per quarter to be ranked" (not implemented); "ideas received outside 04:00–20:00 are priced at the next session" (the rule is the 09:30 / 15:55 boundary described above).
1.0	Oct 2026	First public API: Nasdaq-100 universe, `horizon_days` on every idea, one message per request, `client_ref` idempotency, 5-minute pricing after receipt.

Compatibility notes

- Request and response field names are stable within version 1. New response fields may be added; your parser should ignore unknown fields.
- `horizon_days` is deprecated and ignored; it will be rejected with `bad_field` in a later version, with notice here first.

- The `401` body shape (`{"detail": …}`) differs from other refusals. Handle `401` by status code, not by body.
- The `pricing` response string is descriptive and may be corrected (see Pricing). Do not parse it.
- Rate-limit counters live in server memory and reset on a deploy. Treat the published limits, not observed leniency, as the contract.
- The paths under `PROPOSED` do not exist. They will be announced here, in the OpenAPI file and by email to bot owners before release.

Machine-readable specification: openapi.json (OpenAPI 3.1, the two supported endpoints only). PDF of this page: GEAC-API-documentation.pdf, generated from the same source.

Support

Integration or account questions: ops@geacsystems.com. Include the bot's name and the `client_ref` of any message in question; never include the API key. We reply within one business day.

GEAC Systems Limited · Greenleys Enhanced Alpha Capture Systems · geacsystems.com · © 2026. All rights reserved. Timestamps and identifiers in examples are illustrative.