Nolio API: migration guide from v1 to v2

This guide is for developers who hold Nolio API v1 credentials. It lists everything that changes when you move an integration from API v1 (`https://www.nolio.io/api/`) to API v2 (`https://www.nolio.io/api/v2/`), route by route.

- **API v1 end of life: June 30, 2027.** Until then, your v1 apps keep working unchanged. After that date, API v1 is shut down.
- **v2 reference documentation:** https://www.nolio.io/api/v2/docs/
- **v2 OpenAPI schema (machine-readable):** https://www.nolio.io/api/v2/openapi.json
- **This guide in Markdown (for AI agents):** https://media.cdn.nolio.io/api/migration-v1-to-v2.md
- **This guide in PDF:** https://media.cdn.nolio.io/api/migration-v1-to-v2.pdf
- **API terms:** https://www.nolio.io/en/api-terms/
- **Questions:** contact@nolio.io

The OpenAPI schema is the source of truth for every v2 field, parameter and status code. This guide tells you what changed and where to look; it does not repeat the full reference.

---

0. If you use an AI coding agent

Most migrations will be done with an agent (Claude Code, Cursor, Codex, Copilot...). Paste the prompt below at the start of the session, in the repository of your integration.

```
You are migrating this codebase from the Nolio API v1 to the Nolio API v2.

Sources of truth, in this order:
1. The v2 OpenAPI schema: https://www.nolio.io/api/v2/openapi.json
2. The migration guide: https://media.cdn.nolio.io/api/migration-v1-to-v2.md
Fetch both before writing any code. Never invent an endpoint, field, parameter or
enum value that is in neither source. The structured workout format (workout nodes
and targets) is specified in section 8 of the guide only: the OpenAPI schema types it
as a plain array of objects. Validate each mapped workout with "dry_run": true before
writing. If something is missing from both, say so and stop.

Rules that apply everywhere in v2:
- Base URL https://www.nolio.io/api/v2/ and every path ends with "/" (no redirect: a
  missing slash is a 404).
- Ids are opaque prefixed strings (usr_..., trn_..., ptrn_..., spt_...). Never parse
  them, never compute them. v1 integer ids are rejected: convert the ones you
  store with GET migration/ids/, as described in section 11 of the guide.
- id_partner no longer exists. There is no duplicate check on creates.
- Lists return {"data": [...], "has_more": bool}; paginate with limit (max 100) and
  starting_after=<id of the last row>. Exceptions: metrics/hrv/rmssd/ and
  metrics/hrv/measures/ return a bare array; real/trainings/{id}/messages/ returns
  {"data": [...]} without pagination.
- Errors return {"error": {"code": "...", "message": "...", ...}}. Branch on "code".
```

```
- Quantities are objects: {"value": 3600, "unit": "s"}.
- Updates are PATCH (partial), deletes are DELETE (204). No PUT on resources.
- Competitions are separate resources from trainings, on both calendars.
- A coach targets an athlete with ?athlete=usr_... on reads, updates, deletes and
  realized creates, and with "athletes": ["usr_..."] (one athlete per create) or
  "group": "grp_..." in the body on planned creates.
- A PATCH or DELETE reaches any event the user can see, not only the ones your app
  created: only touch ids your integration stored.

Work plan:
1. Inventory every v1 call in this codebase (paths under /api/ such as /api/get/...,
   /api/create/..., /api/update/..., /api/delete/...) and every stored v1 id.
2. For each call, use the route table in the guide (section 6) to find the v2 route,
   then check its request and response in the OpenAPI schema.
3. Propose the mapping table to me before changing code, including the stored-id
   migration plan (section 11) and the webhook receiver rewrite (section 9).
4. Migrate one resource at a time, with tests against recorded v2 responses.
```

1. What changes, in 12 rules

1. **New apps, new credentials.** An app is bound to one API version for life. v1 credentials get `403 wrong_api_version` on v2. You get a new pair of v2 apps; every user of a partner integration authorizes again (section 3).
2. **Personal use = API key, not OAuth.** For your own account (and the athletes you coach), v2 uses a personal API key `nolio_sk_...`. OAuth is reserved for the partner app (section 3).
3. **The v2 partner app starts blocked**, even if your v1 app was activated. Request activation again from the portal (section 3).
4. **Resource paths and HTTP verbs.** `POST /api/create/training/` becomes `POST /api/v2/real/trainings/`, `POST /api/update/...` becomes `PATCH .../{id}/`, `POST /api/delete/...` becomes `DELETE .../{id}/` (section 6).
5. **Trailing slash is mandatory.** `/api/v2/me` returns a JSON 404, not a redirect.
6. **New opaque ids.** `usr_...`, `trn_...`, `ptrn_...`, `spt_...`. Integer ids are rejected: convert the ids you store with `GET migration/ids/` (section 11).
7. **No `id_partner`.** You can no longer address an event by your own id, and Nolio no longer deduplicates creates or file uploads (section 5.3).
8. **Competitions are their own resources**, on the realized calendar (`/real/competitions/`) and on the planned one (`/planned/competitions/`). Reading `/real/trainings/` alone misses every race.
9. **Granular scopes.** Request `real:read planned:write ...` explicitly. A new app starts with read scopes only (section 4).
10. **Team permission.** In a team with several coaches, writes and deletes on an athlete's planned, realized, metrics and settings data return `403 team_scope_forbidden` until a team manager allows API access (section 4.3).
11. **Metrics have no upsert.** `POST /update/metric/` becomes a read, then `PATCH` or `POST` (section 7.1).
12. **Structured workouts use a new schema.** `structured_workout` becomes `workout`, with different field names and units, and lists no longer carry it (section 8). Webhooks are rewritten as well (section 9).

2. Timeline

Date	What happens
v2 release	API v2 opens to every developer. New developer accounts get v2 apps only. The v1 developer portal stays open to accounts holding a v1 app, but creates no new v1 app.
v2 release + 30 days	Accept the API terms (https://www.nolio.io/en/api-terms/), for v1 and v2, or stop using the API. Continued use after 30 days counts as acceptance.
Until June 30, 2027	Your v1 apps keep working unchanged. Run v1 and v2 side by side while you migrate.
June 30, 2027	API v1 is shut down. v1 tokens stop working.

If a serious reason prevents you from migrating in time, write to contact@nolio.io with an explanation.

---

3. Step 1: get your v2 apps

Open **Settings > MCP and API** in Nolio: https://www.nolio.io/customize/?show=27

Your portal holds two v2 apps:

App	Use it for	Authentication	Accounts reached
Personal	Your own scripts and tools	Personal API key `nolio_sk_...` (no OAuth)	Your account, plus the athletes you coach (with `?athlete=usr_...`)
Partner	A service offered to other Nolio users	OAuth2 authorization code for user data; an API key of the partner app for app settings	Every user who authorizes the app

Conditions (API terms): the personal app requires an active paid Nolio subscription (article 3.2); activating the partner app requires the API access plan (articles 3.3 and 16). Without them, calls return `403 permission_denied`.

Personal app

- Create the key in the portal. Two key slots let you rotate without downtime: create the new key, deploy it, then revoke the old one. A key created in the portal lives until you revoke it; a key created with `POST app/pats/` (scope `app:manage`) can carry an expiry date.
- Send it as `Authorization: Bearer nolio_sk_...`.
- OAuth on the personal app is refused at authorization: `/api/authorize/` and `/api/token/` answer `403 {"error": "access_denied"}`. If your v1 personal app synced a few accounts through OAuth, move those flows to the API key (your own account and coached athletes) or to the partner app (anyone else).

Partner app

- It starts blocked. A first request from the portal opens a **development mode limited to 5 accounts**. A second request moves it to production, with no account limit.
- Activation of a v1 partner app does not carry over to v2: request it again.
- New `client_id` and `client_secret`. Existing v1 tokens are not valid on v2: **each user must go through the OAuth consent again** with the v2 app (switch order per user: section 11).
- The partner app has its own two API key slots. Use one of them for the app-level endpoints `GET app/users/` and `app/webhook/`: a user's OAuth token is refused there with `403 pat_required`.

Check what your app can do at any time with `GET /api/v2/app/` (scopes, quotas) and, with an API key of that app, `GET /api/v2/app/users/` (linked users).

Testing. There is no sandbox (API terms, article 3.7): test on your own account or on test accounts you create, then with the 5 accounts of the partner app's development mode.

---

4. Authentication and scopes

4.1 OAuth2 (partner app)

	v1	v2
Authorize URL	`https://www.nolio.io/api/authorize/`	Same
Token URL	`https://www.nolio.io/api/token/`	Same
Revoke	`https://www.nolio.io/api/deauthorize/`	Same
Server metadata		`https://www.nolio.io/.well-known/oauth-authorization-server`
Client authentication	HTTP Basic or `client_secret_post`	Same
PKCE (S256)	Required for public clients only	Same. Recommended for every client
Access token lifetime	24 h	Same
Authorization code lifetime	10 min	Same
Refresh token	No expiry, rotated at each refresh	Same
Scopes	`read write`	Granular, see 4.2

The flow is unchanged. What changes: the v2 `client_id` / `client_secret`, the scopes you request, and the base URL of your API calls.

A successful refresh invalidates the previous refresh token, with no grace period: serialize refreshes per user, or two parallel refreshes will lock one of them out.

4.2 Scopes

v2 scopes follow `<universe>:<action>`, with independent `read`, `write` and `delete`:

Universe	Scopes	Covers
`real`	`real:read`, `real:write`, `real:delete`	Realized calendar: trainings, competitions, notes, questionnaire answers
`planned`	`planned:read`, `planned:write`, `planned:delete`	Planned calendar: trainings, competitions, notes, cycles, questionnaires, scheduled messages
`metrics`	`metrics:read`, `metrics:write`, `metrics:delete`	Metrics, records, HRV
`messaging`	`messaging:read`, `messaging:write`, `messaging:delete`	Conversations and messages

Universe	Scopes	Covers
`library`	`library:read`, `library:write`, `library:delete`	Workout, plan and exercise templates
`settings`	`settings:read`, `settings:write`, `settings:delete`	Profile, zones, sports, tags
`team`	`team:read`, `team:write`, `team:delete`	Teams, groups, members, invitations
`marketplace`	`marketplace:read`, `marketplace:write`, `marketplace:delete`	Storefront and plans for sale
`billing`	`billing:read`	Nolio subscriptions and invoices
`app`	`app:manage`	Manage the API keys of your own app

To keep what a v1 app could do, request: `real:read real:write real:delete planned:read planned:write planned:delete metrics:read metrics:write messaging:write settings:read team:read`.

- A new v2 app is created with the `:read` scopes only. Widen them in the portal before requesting write scopes in OAuth.
- The consent screen lets the user untick scopes. Always check what was granted.
- A missing scope returns `403` with `{"error": {"code": "missing_scope", "scopes": [...]}}`.

4.3 Team permission (new)

In a team with **several coaches**, the team blocks API writes and deletes by coaches on its athletes by default, for `planned`, `real`, `metrics` and `settings`. Reads are allowed by default, but a team manager can block them too: handle `team_scope_forbidden` on reads as well. A blocked call returns `403 team_scope_forbidden` with `scopes`, `team_name` and, for a manager of the team, `team_options_url`. A team manager allows it in **Team options > API and MCP**. Do not retry: show the message to the coach.

Separately, an athlete can turn off API access for their coaches (`403 athlete_api_access_denied`, with `athlete_name`, reads included) or refuse coach edits such as metrics edits (`403 permission_denied`).

---

5. Requests and responses

5.1 Base URL, paths, verbs

- Base URL: `https://www.nolio.io/api/v2/`.
- Paths are grouped by universe: `real/`, `planned/`, `metrics/`, `messaging/`, `library/`, `settings/`, `team/`, `marketplace/`, `billing/`, `recovery/`, plus `me/`, `app/`, `batch/`, `sports/`.
- Verbs: `GET` list or detail, `POST` create (201), `PATCH` partial update (200), `DELETE` (204, empty body). No `PUT` on resources.
- **Every path ends with `/`.** Without it you get a JSON 404 `not_found`, never a redirect.
- Bodies are JSON objects. Unknown body fields are a `400` naming each one. Unknown query parameters on a list are a `400` that also lists the valid ones.
- On the list endpoints of the realized and planned calendars, `?fields=id,name,date_start` selects the fields of each row.
- v1 `?id=` on `/get/training/`, `/get/planned/training/`, `/get/note/` and `/get/planned/note/` becomes the detail route `.../{id}/`.

5.2 Ids

v1	v2
`nolio_id` (integer) of a realized training	`id` = `trn_...`
Realized competition, note	`cmp_...`, `not_...`
Planned training, competition, note	`ptrn_...`, `pcmp_...`, `pnot_...`
Planned cycle, questionnaire, scheduled message	`pcyc_...`, `pqz_...`, `pmsg_...`
`athlete_id`, `user_id` (integer)	`usr_...`
`sport_id` (integer)	`spt_...` (list them with `GET /api/v2/sports/`)
Metric value, metric type (`metric_id`)	`mtr_...`, `mti_...`
HRV measure	`hrv_...`
Group, team, tag	`grp_...`, `team_...`, `tag_...`

- Ids are opaque: never parse them, never build them. Store them as strings.
- Integer ids are rejected: `400` in a body, `404` in a path.
- Migrating ids you already store: section 11.

5.3 `id_partner` is gone

v1 used your `id_partner` to address events (update, delete, message) and to avoid duplicates. v2 has no `id_partner` field and no lookup by external id.

- Store the `id` that v2 returns on create, and address the event with it.
- **There is no duplicate check.** A create or a file upload sent twice creates two objects. If a request times out, read the calendar (`GET` with `date_from` / `date_to`) before retrying.
- Keep your own mapping between your ids and Nolio ids.
- **Writes are no longer limited to your app's events.** In v1, update and delete only found the events your app created. In v2, `PATCH` and `DELETE` reach any event visible to the user, including ones from a device or typed by hand: only touch ids you stored.

5.4 Coach targeting an athlete

	v1	v2
Read, update, delete an athlete's data	`athlete_id` (body or query)	`?athlete=usr_...` (query, also on `PATCH` and `DELETE`)
Create a realized event for an athlete	`athlete_id`	`?athlete=usr_...`
Create a planned event	`athlete_id`	In the body: `"athletes": ["usr_..."]` or `"group": "grp_..."`. Omit both for yourself. `?athlete=` on a planned create is a 400
Update or delete a group's planned event		`?group=grp_...` is required
List the athletes you coach	`GET /api/get/athletes/`	`GET /api/v2/me/athletes/`

Send one athlete per planned create to keep every id: with several ids in `athletes`, Nolio creates one individual event per athlete but the response returns only one of them (its `athletes` holds a single id).

Planned competitions, notes, cycles, questionnaires and scheduled messages take one athlete or one group (several athletes is a 400).

5.5 Quantities, dates, nulls

	v1	v2
Duration	Integer, seconds	`{"value": 3600, "unit": "s"}`. Read in `s`; write in `s`, `min` or `h`
Distance	Float, kilometers	`{"value": 12, "unit": "km"}`. Read in `km`; write in `m`, `km` or `mi`
Elevation gain	Float, meters	`{"value": 80, "unit": "m"}`. Read in `m`; write in `m` or `ft`
Date	`YYYY-MM-DD`	Same
Time, realized events and notes (both calendars)	`hour_start` `HH:MM:SS` or `""`	`hour_start` `HH:MM:SS`, read-only: a create or update that sends it is a 400 (v1 ignored it)
Time, planned trainings and competitions	`hour_start` `HH:MM:SS` or `""`	`time` (`HH:MM`) or `moment` (time of day)
Time, metric values / HRV	`hour` `HH:MM` or `""` / `HH:MM`	`hour` `HH:MM:SS` or `null` / `HH:MM`
Missing value	`0` or `""`	`null`
Sport	`sport` (name) + `sport_id`	`sport` = `spt_...`
RPE, feeling, athlete comment	`rpe`, `feeling`, `description`	Grouped in `debrief: {rpe, feeling, comment}`

A bare number where v2 expects a quantity object is a 400.

Realized training, v1 (`GET /api/get/training/`, excerpt):

```
{"nolio_id": 123456, "name": "Intervals 30/30", "sport": "Trail", "sport_id": 52,
 "date_start": "2026-06-07", "hour_start": "10:30:00", "duration": 3600, "distance": 12.0,
 "elevation_gain": 80.0, "rpe": 8, "feeling": 4, "description": "Good legs.",
 "planned_name": "Intervals 30/30"}
```

The same training, v2 (`GET /api/v2/real/trainings/trn_9aK2xQ7bLm4Fw/`, excerpt):

```
{"id": "trn_9aK2xQ7bLm4Fw", "name": "Intervals 30/30", "sport": "spt_o9Qqbid1vRktj",
 "user": "usr_2HVKGglXTOhYy", "date_start": "2026-06-07", "hour_start": "10:30:00",
 "duration": {"value": 3600, "unit": "s"}, "distance": {"value": 12, "unit": "km"},
 "elevation_gain": {"value": 80, "unit": "m"},
 "debrief": {"rpe": 8, "feeling": 4, "comment": "Good legs."},
 "tags": ["tag_R7hVe2Kd1Ns8Pq"], "planned": "ptrn_4Lq8vN2cXe7Rt"}
```

5.6 Pagination and filters

	v1	v2
Page size	`limit`, default 30, max 300 (trainings)	`limit`, default 10, max 100
Next page	None (narrow the date window)	`starting_after=<id of the last row>`
Response	Bare JSON array	`{"data": [...], "has_more": true}`
Date window	`from`, `to`	`date_from`, `date_to` on calendars and metric values; `from`, `to` kept on records and HRV
Order	Depends on the route	Newest first

```
params = {"date_from": "2026-01-01", "date_to": "2026-06-30", "limit": 100}
rows = []
while True:
    page = requests.get(f"{BASE}/real/trainings/", headers=headers, params=params).json()
    rows += page["data"]
    if not page["has_more"]:
        break
    params["starting_after"] = page["data"][-1]["id"]
```

Exceptions: `metrics/hrv/rmssd/` and `metrics/hrv/measures/` still return a bare array with the v1 limits. `metrics/records/` returns `{"data": [...], "has_more": false, "windows": [...]}`. `real/trainings/{trn_id}/messages/` returns `{"data": [...]}` without pagination.

v1 trainings with `from` and `to` matched any training overlapping the window; v2 trainings filter on `date_start`. Notes and cycles keep overlap semantics and also take `?date=`.

5.7 Errors

v1 returned plain text in most errors (for example `400 "Invalid training_id"`). v2 always returns JSON in one format:

```
{"error": {"code": "validation_error", "message": "Invalid request.",
           "fields": {"duration": ["Expected an object {value, unit}."]}}}
```

HTTP	`code`	Meaning
400	`validation_error`, `read_only_field`, `bad_request`	Invalid body or parameter. `fields` gives the details per field
400	`unknown_record_window`	Records window not in the catalog (`windows` lists the valid ones)
401	`not_authenticated`, `authentication_failed`	Missing, unknown or expired token
403	`missing_scope`	Scope not granted (`scopes` lists the missing ones)
403	`team_scope_forbidden`	Team blocks this API access (section 4.3): writes and deletes by default, reads if a manager chose to
403	`athlete_api_access_denied`	The athlete turned off API access for their coaches

HTTP	code	Meaning
403	`permission_denied`	No active paid Nolio subscription on the app owner, athlete you do not coach, or athlete-side edit setting
403	`wrong_api_version`	v1 credentials on v2 (v2 credentials on `/api/` get a v1 403)
403	`pat_required`	An OAuth token used on `app/users/` or `app/webhook/` (use an API key of the app)
404	`not_found`	Object does not exist **or is not visible to you**, or path without trailing slash
409	`conflict`	Slot already taken (metrics), concurrent edit (workouts, section 8), overlapping cycle, plan on sale or purchased
429	`throttled`	Quota exceeded, see `Retry-After`
503	`service_unavailable`	Retry later

"Does not exist" was a 400 in v1 and is a 404 in v2. More than 30 failed authentications with the same token in an hour also return a 429.

5.8 Quotas

Quotas are counted per app, per hour and per day, with fixed windows. They are recomputed every night from the number of accounts the app serves. A call refused with a 429 counts too: do not retry in a loop, wait for `Retry-After`.

	v1	v2
Personal app	1,000 / h, 10,000 / day	Athlete account: 1,000 / h, 5,000 / day. Coach account: 2,000 / h, 10,000 / day. Plus 100 / h and 250 / day per athlete coached
Partner app	Base + 50 / h and 250 / day per linked account	Base + 50 / h and 250 / day per linked account and per athlete the owner coaches
Records	20 / min	20 / min
Response headers	None	`X-RateLimit-Limit-Hour`, `X-RateLimit-Remaining-Hour`, `X-RateLimit-Limit-Day`, `X-RateLimit-Remaining-Day`; `Retry-After` on 429
Read your quotas		`GET /api/v2/app/`

`POST /api/v2/batch/` costs one unit per operation in addition to the call itself (section 10), and `GET /api/v2/migration/ids/` one unit per id (section 11).

---

6. Route-by-route table

All v2 paths are relative to `https://www.nolio.io/api/v2/`. `{id}` stands for the opaque id of the object (prefix given in the path). Partner-specific v1 routes are not covered here: contact contact@nolio.io.

6.1 User and team

v1	v2	Scope	What changes
`GET /get/user/`	`GET settings/profile/` (or `GET me/` for id and name)	`settings:read` (`me/`: any)	`id` is `usr_...`. Adds `gender`, `lang`, `timezone`, `units`. `PATCH settings/profile/` is new. A coach reads an athlete's profile with `?athlete=`

v1	v2	Scope	What changes
`GET /get/user/meta/`	`GET metrics/items/` then `GET metrics/values/?item=mti_...&date_from=&date_to=`	`metrics:read`	No aggregated endpoint: one call per metric type. Values use `date_start` instead of `date`. Values are served with the unit of the metric type (`hh:mm:ss`, `mm:ss:cc`...), where v1 relabeled these units `seconds` or `hundredths`
`GET /get/athletes/`	`GET me/athletes/`	any scope except `marketplace:*` and `app:manage`	`nolio_id` becomes `id` (`usr_...`). `teams` is dropped (groups: `GET me/groups/`). The `wants_coach` parameter is gone: read yourself with `GET me/`. Adds `has_api_access` and `?q=` search. Paginated. Includes the athletes you coach as an assistant
`GET /get/teams/`	`GET team/teams/`, `GET team/teams/{team_id}/members/`, `GET team/groups/`, `GET team/groups/{grp_id}/members/` (or `GET me/groups/`)	`team:read`	The nested v1 tree becomes flat resources. Visible to any team member, not only managers

6.2 Realized calendar

v1	v2	Scope	What changes
`GET /get/training/`	`GET real/trainings/`, `GET real/trainings/{trn_id}/`	`real:read`	**Competitions are excluded**: read `GET real/competitions/` too. Multi-sport segments are not listed separately. Fields: see 5.5. `zones` becomes `time_in_zones` (on lists with `?time_in_zones=true`). `planned_name`, `planned_sport` ... become `planned` (`ptrn_` / `pcmp_` id of the planned event). `load_foster` / `load_coggan` become `load: {coggan, foster}` (plus `coggan_load`, a load set by hand). `file_url` is on the detail only. Dropped: `date_end`, `kilojoules`, `avg_watt`, `max_watt`, `np`, `ftp`, `rftp`, `weight`, `critical_power`, `wbal`, `rest_hr_user`, `max_hr_user`, `elevation_loss`, `is_competition`. New filters: `sport`, `duration_min` / `max`, `distance_min` / `max`, `elevation_gain_min` / `max`, `rpe_min` / `max`, `feeling_min` / `max`, `name`, `planned`
`GET /get/training/info/`	`GET real/trainings/{trn_id}/` + `GET real/trainings/{trn_id}/streams/` + `GET real/trainings/{trn_id}/laps/`	`real:read`	Split in three. `tags` are `tag_...` ids instead of names. v1 `streams` (one object per sample) becomes `.../streams/` (one array per stream). Lap `start` / `end` become `start_index` / `end_index`
`GET /get/training/streams/`	`GET real/trainings/{trn_id}/streams/` (competitions: `GET real/competitions/{cmp_id}/streams/`)	`real:read`	Id in the path. Same shape as v1 (`file_url`, `stream_<name>` arrays, `custom_laps`). Adds `stream_gps`, power streams, `dynamic_streams` and `source`. 204 when there is no file (unchanged)
`POST /create/training/`	`POST real/trainings/`	`real:write`	No `id_partner`. `sport_id` becomes `sport` (`spt_...`, enabled for the athlete). Quantities as objects. `rpe`, `feeling`, `description` move into `debrief`. Coach: `?athlete=usr_...`. Returns the full object
`POST /update/training/`	`PATCH real/trainings/{trn_id}/`	`real:write`	Addressed by id, not `id_partner`. Partial update. A future `date_start` is rejected
`POST /delete/training/`	`DELETE real/trainings/{trn_id}/`	`real:delete`	204. Can be undone (section 10)

v1	v2	Scope	What changes
`POST /create/competition/`	`POST real/competitions/`	`real:write`	Same as trainings, plus `event`, `location`, `goal_type`, `result_link`, `perf`. `debrief.feeling` is read-only on competitions. Laps: `GET real/competitions/{cmp_id}/laps/`
`POST /update/competition/`	`PATCH real/competitions/{cmp_id}/`	`real:write`	Same as trainings
`POST /delete/competition/`	`DELETE real/competitions/{cmp_id}/`	`real:delete`	204
`GET /get/note/`	`GET real/notes/`, `GET real/notes/{not_id}/`	`real:read`	The v1 response key `type` becomes `note_type`, with slugs (table below). `injury_type`, `discomfort_type`, `sick_type`, and the `duration` and `sports` of availability notes are not returned in v2. `date`, `date_from`, `date_to` use overlap
`POST /create/note/`	`POST real/notes/`	`real:write`	Fields: `name`, `date_start`, `description`, `color`
`POST /update/note/`	`PATCH real/notes/{not_id}/`	`real:write`	Addressed by id
`POST /delete/note/`	`DELETE real/notes/{not_id}/`	`real:delete`	204
`POST /create/training/message/`	`POST real/trainings/{trn_id}/messages/`	`messaging:write`	Training in the path, body is `{"content": "..."}` only. Coach accounts only. Returns the message `{id, thread, sender, content, sent_at, is_edited}`. Read the thread with `GET` on the same path (`messaging:read`)
`POST /mark_as_view/training/`	`POST real/trainings/{trn_id}/mark-as-viewed/` (competitions: `POST real/competitions/{cmp_id}/mark-as-viewed/`)	`real:write`	Returns the full training
`POST /upload/file/`	`POST real/trainings/import/`	`real:write`	`id_partner` removed: **no duplicate check**. Own account only (no `athlete_id`). Base64 up to 25 MB. Still 202 (processed asynchronously)
	`GET real/quiz_results/`	`real:read`	New: questionnaire answers

`note_type` slugs: `injury` (v1 `blessure`), `discomfort` (`inconfort`), `sick` (`malade`), `availability` (`dispo`), `unavailability` (`indispo`), `rest` (`repos`), `menstruation` (`menstruel`), `goal` (`goal`).

6.3 Planned calendar

v1	v2	Scope	What changes
`GET /get/planned/training/`	`GET planned/trainings/` , `GET planned/trainings/{ptrn_id}/`	`planned:read`	**Competitions are excluded**: read `GET planned/competitions/` too. Lists carry `has_structured_workout` only: read `workout` on the detail (section 8). Adds `?group=grp_...` , `?with_revision=true` , `?time_in_zones=true`
`POST /create/planned/training/`	`POST planned/trainings/`	`planned:write`	Audience in the body (`athletes` or `group`). `structured_workout` becomes `workout` . `hour_start` becomes `time` (`HH:MM`) or `moment` . New: `tags` , `recurrence` , `dry_run` . `plan_id` has no v2 equivalent: planned events no longer carry a plan link (to put a template on a calendar, use `POST library/plan_frames/{pfr_id}/apply/` ; applied plans are listed with `GET planned/applied-plans/`)
`POST /update/planned/training/`	`PATCH planned/trainings/{ptrn_id}/`	`planned:write`	`workout` is replaced as a whole. Optional concurrency check with `If-Match` (409 on conflict)
`POST /delete/planned/training/`	`DELETE planned/trainings/{ptrn_id}/`	`planned:delete`	204
`POST /create/planned/competition/`	`POST planned/competitions/`	`planned:write`	Same as planned trainings, plus `event` , `location` , `goal_type` . One athlete or one group per create
`POST /update/planned/competition/`	`PATCH planned/competitions/{pcmp_id}/`	`planned:write`	Same as planned trainings
`POST /delete/planned/competition/`	`DELETE planned/competitions/{pcmp_id}/`	`planned:delete`	204
`GET /get/planned/note/`	`GET planned/notes/` , `GET planned/notes/{pnot_id}/`	`planned:read`	Same response changes as realized notes (`type` becomes `note_type` , type-specific fields are not returned). **Multi-day notes are now cycles**: `GET planned/cycles/`
`POST /create/planned/note/`	`POST planned/notes/` (multi-day: `POST planned/cycles/`)	`planned:write`	Fields: `name` , `date_start` , `description` , `color` , `athletes` (one) or `group` . `hour_start` is read-only: no time can be set on create or update. `plan_id` has no equivalent. Cycles cannot overlap (409): overlapping v1 multi-day notes do not migrate as they are
`POST /update/planned/note/`	`PATCH planned/notes/{pnot_id}/`	`planned:write`	Addressed by id

v1	v2	Scope	What changes
`POST /delete/planned/note/`	`DELETE planned/notes/{pnot_id}/`	`planned:delete`	204

6.4 Metrics, records, HRV

v1	v2	Scope	What changes
`GET /get/metric/`	`GET metrics/values/{mtr_id}/` (list: `GET metrics/values/`)	`metrics:read`	`date` becomes `date_start`. `type` is gone: the metric type is `item` (`mti_...`), named in `GET metrics/items/`. New: `description` (free text of the value). `hour` changes from `HH:MM` or `""` to `HH:MM:SS` or `null`. `source` is a lowercase slug
`POST /update/metric/`	`POST metrics/values/` and `PATCH metrics/values/{mtr_id}/`	`metrics:write`	**No upsert**, see 7.1. `metric_id` becomes `item` (`mti_...`, from `GET metrics/items/`), `new_value` becomes `value`. Adds `hour`, `description`. Deleting is new: `DELETE metrics/values/{mtr_id}/` (`metrics:delete`)
`GET /get/records/`	`GET metrics/records/`	`metrics:read`	Response `{data, has_more, windows}`. `sports` takes `spt_...` ids. `cat` and `record_type` are optional: without them you get only the `windows` catalog, `data` is empty. New `top` (1 to 10). `training_id` becomes `trn_` / `cmp_`. Paces are converted to the sport's unit
`GET /get/hrv/rmssd/`	`GET metrics/hrv/rmssd/`	`metrics:read`	Same parameters. `source` is a case-insensitive slug. `id` is `mtr_...`. Still a bare array
`GET /get/hrv/measures/`	`GET metrics/hrv/measures/`	`metrics:read`	`id` is `hrv_...`. Limits, 90-day window and `with_raw` unchanged. Still a bare array

6.5 Webhook configuration

v1	v2
3 URLs set in the v1 portal (realized, planned, metrics)	One URL per app: **Settings > MCP and API > Webhooks** in Nolio, or `GET` / `PUT` / `DELETE app/webhook/` authenticated with an API key of that same app (for the partner app: one of its own key slots)
`webhook_key`	Signing secret `whsec_...`, rotated with `POST app/webhook/rotate-secret/`

---

7. Endpoints that changed shape

7.1 Metrics: create or update

v1 `POST /update/metric/` created or replaced the value of a day, whatever its hour. In v2 a value is identified by (item, date, `hour`). To reproduce the v1 behavior:

1. Read the day: `GET metrics/values/?item=mti_...&date_from=2026-10-06&date_to=2026-10-06`.
2. If a value exists, update it with `PATCH metrics/values/{mtr_id}/`.
3. Otherwise create it with `POST metrics/values/` and `{"item": "mti_...", "date_start": "2026-10-06", "value": 52.4}`.

A `POST` returns `409 conflict` only when a value exists with the same item, date **and** `hour`: a `POST` without `hour` on a day that already holds a value with an hour creates a second value.

A coach writing an athlete's metrics needs the athlete's permission to edit metrics, plus the team permission (4.3).

7.2 User metadata

v1 `GET /get/user/meta/` returned one object keyed by metric name. In v2, list the metric types with `GET metrics/items/`, then read each type's values with `GET metrics/values/?item=mti_...`.

7.3 Training detail

v1 `GET /get/training/info/` returned everything in one call. In v2: the detail (`GET real/trainings/{trn_id}/`), the streams (`.../streams/`) and the laps (`.../laps/`).

---

8. Structured workouts

8.1 Transport

	v1	v2
Field	`structured_workout` (JSON array)	`workout` (JSON array)
Where	Body of `create/update planned training` and `planned competition`; returned in `GET /get/planned/training/` lists	Inline on `planned/trainings/`, `planned/competitions/`, `library/training_frames/` and plan items. **On the detail only**: lists carry `has_structured_workout`
Update	Replaced only if the array is not empty (`[]` cannot clear)	Replaced as a whole; `null` clears it. Keep the `id` of nodes you keep
Validation	Generic 400 "Structured workout format error", envelope saved anyway on update	Whole request rejected, field-level errors in `error.fields`
Validate without saving	No	`"dry_run": true` on creates of planned trainings, planned competitions and workout templates returns `{dry_run, is_valid, summary}`
Concurrent edits	Last write wins	Read `revision`, send it back as `If-Match` (or `?revision=`): 409 if someone changed it meanwhile
Read back	Not re-postable	The `workout` array can be sent back as is (`resolved` is ignored). Event-level read-only fields (`id`, `summary`, `revision`, `has_structured_workout`, `load`, `time_in_zones`) return `400 read_only_field` in a write body

8.2 Nodes

v1	v2
`"type": "step"`	`"kind": "step"`
`"type": "repetition"`, `value`, `steps`	`"kind": "repeat"`, `times`, `steps` (new: `skip_last`). Up to 3 levels of nesting

v1	v2
`step_duration_type: "duration"` + `step_duration_value` (s)	`"duration": {"value": 600, "unit": "s"}` (`s`, `min`, `h`)
`step_duration_type: "distance"` + `step_duration_value` (m)	`"distance": {"value": 5000, "unit": "m"}` (`m`, `km`, `mi`)
`open_duration: true`	`"lap_advance": true`
`intensity_type: "active"`	`"intensity": "work"`
`intensity_type`: `warmup`, `cooldown`, `rest`, `ramp_up`, `ramp_down`, `max_effort`	`intensity`: same values (new: `freeride`). Optional: v1 defaulted a missing value to `active`, v2 leaves it empty
`comment`	`comment`
One target per step	`targets`: 1 or 2 targets (primary, secondary). **Required**: a step without target says `[{"type": "no_target"}]`

A step with neither `duration` nor `distance` is a note step, not sent to devices.

8.3 Targets

v1 `target_type`	v2 target
`power` (W)	`{"type": "raw", "value": 200, "value_max": 250, "unit": "W"}`
`heartrate` (bpm)	`{"type": "raw", ..., "unit": "bpm"}`
`speed` (km/h)	`{"type": "raw", ..., "unit": "km/h"}` (also `mph`, `m/s`)
`pace` (m/s, number)	`{"type": "raw", "value": "4:20", "value_max": "4:50", "unit": "/km"}` (also `/mi`). **A string `m:ss`**, rounded to the second
`pace_min100` (m/s, number)	`{"type": "raw", "value": "1:45", "unit": "/100m"}` (also `/500m`)
`cadence` (rpm)	`{"type": "raw", ..., "unit": "rpm"}` (bike) or `"ppm"` (run), usually as second target
`no_target`	`{"type": "no_target"}`
`target_value_max` alone	`value` alone. **The single value moved from max to `value`**
`target_value_min` + `target_value_max`	`value` (low) + `value_max` (high). Order does not matter on write
(not writable in v1)	`{"type": "metric", "metric": "FTP", "percent": 90, "percent_max": 95}`: percent of a metric (`metric` = `key` of an item from `GET metrics/items/` whose `is_wb_compatible` is true)
(not writable in v1)	`{"type": "zone", "zone": "Tempo", "stream": "watts"}` or `"zone": 4`: a zone of the athlete
(not writable in v1)	`{"type": "rpe", "rpe": 7}`, `{"type": "rir", "rir": 2}`

Strength exercises, supersets and EMOM / For Time / AMRAP circuits could not be written in v1, so a migration does not need them. In v2 they are `kind: "exercise"` (with an `exf_...` exercise id), `superset`, `emom`, `for_time`, `amrap`; the OpenAPI schema does not detail their fields.

8.4 Example

v1:

```
{"structured_workout": [
  {"type": "step", "intensity_type": "warmup", "step_duration_type": "duration", "step_duration_value":
600,
   "target_type": "power", "target_value_min": 200, "target_value_max": 250, "open_duration": true},
  {"type": "repetition", "value": 3, "steps": [
    {"type": "step", "intensity_type": "active", "step_duration_type": "duration",
"step_duration_value": 180,
     "target_type": "power", "target_value_min": 300, "target_value_max": 350},
    {"type": "step", "intensity_type": "rest", "step_duration_type": "duration", "step_duration_value":
180,
     "target_type": "no_target"}]},
  {"type": "step", "intensity_type": "cooldown", "step_duration_type": "distance",
"step_duration_value": 5000,
   "target_type": "heartrate", "target_value_min": 100, "target_value_max": 150}]}
```

v2 (`POST planned/trainings/`):

```
{"date_start": "2026-10-12", "name": "3x3min threshold", "sport": "spt_o9Qqbid1vRktj",
 "athletes": ["usr_2HVKGglXTOhYy"],
 "workout": [
  {"kind": "step", "intensity": "warmup", "duration": {"value": 600, "unit": "s"},
   "targets": [{"type": "raw", "value": 200, "value_max": 250, "unit": "W"}], "lap_advance": true},
  {"kind": "repeat", "times": 3, "steps": [
    {"kind": "step", "intensity": "work", "duration": {"value": 180, "unit": "s"},
     "targets": [{"type": "raw", "value": 300, "value_max": 350, "unit": "W"}]},
    {"kind": "step", "intensity": "rest", "duration": {"value": 180, "unit": "s"},
     "targets": [{"type": "no_target"}]}]},
  {"kind": "step", "intensity": "cooldown", "distance": {"value": 5000, "unit": "m"},
   "targets": [{"type": "raw", "value": 100, "value_max": 150, "unit": "bpm"}]}]}
```

The sport must be one of the athlete's sports that supports structured workouts.

---

9. Webhooks

	v1	v2
Configuration	3 URLs (realized, planned, metrics)	One URL per app, and the list of events you want
Authentication	Static header `X-Nolio-Key` = `webhook_key`	`Webhook-Signature: t=<unix timestamp>,v1=<hex>`, HMAC-SHA256 of `"<t>.<raw body>"` with your `whsec_...` secret
Payload	Notification `{notif_type, object_type, object_id, user_id, date_object, livemode}`: you call the API afterwards	`{"id": "evt_...", "type": "real.training.created", "created": 1759744800, "data": {"object": {...}}}`: `data.object` is the object as the API returns it on `GET`
Deletion	`deleted_event` + `object_id`	`data.object` = `{"id": "trn_...", "is_deleted": true}`

	v1	v2
Events	`new/updated/deleted` x event, planned event, metric	`<universe>.<resource>.<created/updated/deleted>` for `real.training`, `real.competition`, `real.note`, `metrics.value`, `planned.training`, `planned.competition`, `planned.note`, `planned.cycle`, `planned.quiz`, `planned.message`, plus `access.ended`, `billing.subscription.*`, `team.member.*`, `team.group.*`, `team.invitation.refused`
Default subscription	Every event of a configured URL	If you choose none: `planned.training.*` and `real.training.*` created and updated
Planned events for N athletes	One notification per athlete	One event per app
Retries	None (one attempt, 10 s timeout)	5 attempts (after 30 s, 5 min, 30 min, 2 h), any 2xx is a success, 10 s timeout. Disabled automatically after 20 consecutive failed deliveries
Test	`livemode: false`, `object_id: 0`	Event `ping` with an empty object. No `livemode` field

The list of event types, with which ones are on by default, is in `GET app/webhook/` (`available_events`). An event is only sent if your app holds the `<universe>:read` scope it belongs to. `team.*` events carry a `data.context`. `access.ended` carries one (`data.context.team`) only when the access ended through a team: treat it as optional. `team.member.*` events describe a member joining, leaving, being paused or resumed. `messaging.message.*` events (a message posted, edited or deleted) carry `data.context.users`, the participants who see the message: read the thread with the token of one of them.

`access.ended` is delivered whatever your subscription, as long as your webhook is configured and enabled: one raised while it is paused or disabled is not replayed. It means your app no longer has access to that user's data: delete it on your side (API terms, article 7.2). Reconcile regularly with `GET app/users/` (API key of the app), which lists the users you still reach.

Data from restricted connectors is always sent in its restricted form in webhooks (section 12).

Receiver, in Python:

```
import hashlib, hmac, time

def verify(raw_body: bytes, header: str, secret: str, tolerance: int = 300) -> bool:
    parts = dict(item.split("=", 1) for item in header.split(","))
    timestamp, signature = parts["t"], parts["v1"]
    if abs(time.time() - int(timestamp)) > tolerance:
        return False
    expected = hmac.new(secret.encode(), f"{timestamp}.".encode() + raw_body,
hashlib.sha256).hexdigest()
    return hmac.compare_digest(expected, signature)
```

What to change in your receiver:

- Send `events` explicitly with `PUT app/webhook/`, listing every type your v1 receiver handled: the default subscription only covers `planned.training` and `real.training` created and updated, so deletions, competitions, notes and metric values are not delivered without it.
- Verify the signature on the raw body, before parsing. Drop the `X-Nolio-Key` check.
- Dispatch on `type`. Ignore unknown types (including `ping`): new ones will be added.
- Deduplicate on `id`: a retry delivers the same `id` again.

- Use `data.object` directly instead of calling the API back.
- Handle `access.ended`.
- Answer 2xx quickly and process asynchronously.

---

10. New in v2 that helps the migration

- **Batch:** `POST batch/` runs up to 100 operations in one call (`POST`, `PATCH`, `DELETE` under `planned/` and `library/`, plus `POST library/training_frames/{id}/apply/`), with `dry_run`. One quota unit per operation.
- **Undo:** every `DELETE` is logged. `GET recovery/` lists recent deletions and `POST recovery/{id}/undo/` restores one. A batch is undone as a whole with `POST recovery/batches/{bat_id}/undo/`, unless it contains an update (`is_undoable: false`).
- `dry_run` on creates of planned trainings, planned competitions and workout templates validates without saving. For any other write, use `POST batch/` with `dry_run`.
- **New resources:** planned cycles, questionnaires and scheduled messages, the template library (`library/`), applied plans, zones, tags, sports, teams and groups management, stats and training load, messaging, marketplace, billing. See the reference documentation.

---

11. Migrating the ids you already store

v2 does not accept v1 integer ids. `GET migration/ids/` converts the Nolio ids your database stores from v1 (`athlete_id`, `nolio_id`, `sport_id`, `metric_id`…) into v2 ids, until June 30, 2027 (then `410`). For each user:

1. **The user authorizes your v2 app** (partner app) or, for your own account, you create an API key (personal app).
2. **Convert the ids you store** for that user, with that user's token (or your API key for your own account):
    - `GET migration/ids/?type=real.event&ids=101,102,103` takes 1 to 100 v1 ids and returns `{"data": [{"v1_id": 101, "id": "trn_..."}], "not_found": [103]}`.
    - **Users first** (`type=user`): yourself and the athletes you coach. Then, as a coach, convert each athlete's events with `&athlete=usr_...`: one series of calls per athlete.
    - An id lands in `not_found` when this token could not read it with the matching `GET`: deleted, not visible, missing scope or wrong `type`. Same rules and scopes as the `GET`.
    - Each id costs one quota unit on top of the call (section 5.8): plan large histories over several hours.
    - **Events your app created in v1** are known to you only by your `id_partner`, never by a Nolio id: list the calendar on the date range you hold, then match each one on **date + sport + name** (and `duration` for realized trainings). Store the v2 `id` next to your key.
    - **Teams and groups** had no v1 id: `GET team/teams/` and `GET me/groups/`.

v1 id you store	`type`	v2 id
`athlete_id`, `user_id`	`user`	`usr_...`
`sport_id`	`sport`	`spt_...`
`metric_id` (metric type)	`metric_item`	`mti_...`
Metric value	`metric_value`	`mtr_...`
`nolio_id` of a realized training or competition	`real.event`	`trn_...` or `cmp_...`

v1 id you store	`type`	v2 id
Realized note	`real.note`	`not_...`
`nolio_id` of a planned training or competition	`planned.event`	`ptrn_...` or `pcmp_...`
Planned note	`planned.note`	`pnot_...`, or `pcyc_...` for a multi-day note

12. Restricted connector data

Some connected services impose their own terms on their data, in v1 and in v2: Strava and Whoop (trainings and metrics), Oura (metrics). Restricted rows are still returned, with the measured fields set to `null` and a `restriction` object:

```
{"restriction": {"code": "connector_terms", "source": "strava", "message": "...", "fields":
["duration", "distance"], "doc": "https://www.nolio.io/en/api-terms/#annex"}}
```

Difference in v2: on `real/trainings/` and `real/competitions/`, a `duration_*`, `distance_*` or `elevation_gain_*` filter leaves restricted rows out of the list. Details per provider: https://www.nolio.io/en/api-terms/#annex