New Endpoints — everything the feed release adds

Author: Josh S. Sakweli, Backend Lead Team
Last Updated: 2026-09-18
Version: v1.0 (backend branch `fet/feed_baking`)

Base URL: `https://dev.api.nexgate.co` (staging) · `https://api.nexgate.co` (production)

Short Description: Every endpoint here is **new** — a new URL on a new controller. Nothing in this document replaces a field or changes an existing response. It is in two parts: **Part A** is what the apps call (the five feed surfaces and the interactions endpoint), **Part B** is what the admin panel and the operations team call (sponsored slots, business rules, moderation labels, interest mapping, backfills and the per-account debug trace).

> **Its sibling document is** `08_AMENDED_ENDPOINTS.md` — the endpoints you already have that changed behaviour. If a path is in this file it did not exist before; if it is in that file it is not new.

Hints:

- Every endpoint needs `Authorization: Bearer <access token>`. There are no public endpoints here.
- Part B additionally needs an admin role, and says which one per endpoint.
- The five feed surfaces share one contract: `feedSessionId` + `items[]` + `nextCursor`. Learn it once.
- `nextCursor: null` means "nothing more right now" — not "never again". Pull-to-refresh starts a new session.
- Feed surfaces never fail loudly. If the recommendation service or Redis is down you get a simpler feed, not an error.
- Reporting back is not optional: a feed you do not report on cannot learn. See `05_INTERACTIONS_CLIENT_GUIDE.md`.

How this document is organised

	Part	Who calls it	Controllers

□□	**Part A — App-facing**	Android, iOS, web	`HomeFeedController`, `ReelsFeedController`, `MarketFeedController`, `EventsFeedController`, `FeedListsController`, `InteractionController`
□□	**Part B — Admin & operations**	Admin panel, ops	`PromotionAdminController`, `LabelAdminController`, `InterestAdminController`, `FeedBackfillController`

Each entry carries an access chip:

Chip	Meaning
□□ **User**	Any signed-in account
□□ **Admin**	`ROLE_SUPER_ADMIN` or `ROLE_STAFF_ADMIN`
□□ **Super admin**	`ROLE_SUPER_ADMIN` only

Index

Part A — App-facing

#	Endpoint	Purpose	Replaces
A1	`GET /api/v1/feed/home`	The home feed: posts and products	`api/v1/e-social/feed`
A2	`GET /api/v1/feed/home/has-new`	The "new posts" bubble	the feed SSE stream
A3	`GET /api/v1/feed/reels`	The reels feed	`api/v1/e-social/clip/feed`
A4	`GET /api/v1/feed/marketplace`	Marketplace "For You"	—
A5	`GET /api/v1/feed/events`	Events "For You"	—
A6	`GET /api/v1/feed/products/{productId}/similar`	"Similar items" on a product page	—
A7	`GET /api/v1/feed/shops/suggested`	"Shops you might like"	—
A8	`POST /api/v1/feed/interactions/batch`	Report what the user did	—

Part B — Admin & operations

#	Endpoint	Purpose	Role
B1	`POST /api/admin/feed/sponsored`	Start a sponsored campaign	Admin
B2	`GET /api/admin/feed/sponsored`	List active campaigns	Admin
B3	`DELETE /api/admin/feed/sponsored/{id}`	Stop a campaign	Admin
B4	`POST /api/admin/feed/business-rules`	Boost or suppress something for a period	Admin
B5	`GET /api/admin/feed/business-rules`	List running and upcoming rules	Admin
B6	`DELETE /api/admin/feed/business-rules/{id}`	End a rule now	Admin
B7	`GET /api/admin/feed/debug/{accountId}`	Why this person saw this feed	Admin
B8	`POST /api/admin/feed/labels`	Label a post, product, shop or account	Admin
B9	`DELETE /api/admin/feed/labels/{labelId}`	Remove a label	Admin
B10	`GET /api/admin/feed/labels`	Labels on one target, with history	Admin
B11	`GET /api/admin/feed/interests`	The interest list and every mapping	Admin
B12	`GET /api/admin/feed/interests/unmapped`	What still needs mapping	Admin
B13	`PUT /api/admin/feed/interests/product-categories/{categoryId}`	Map a product category	Admin
B14	`PUT /api/admin/feed/interests/event-categories/{categoryId}`	Map an event category	Admin
B15	`PUT /api/admin/feed/interests/hashtags/{hashtag}`	Map a hashtag	Admin
B16	`POST /api/admin/feed/backfills`	Republish existing data into the feed	Super admin
B17	`GET /api/admin/feed/backfills`	Recent backfill runs	Super admin

#	Endpoint	Purpose	Role
B18	`POST /api/admin/feed/backfills/{runId}/cancel`	Stop a run	Super admin

Standard Response Format

Same envelope as the rest of the platform.

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Home feed",
  "action_time": "2026-09-18T10:30:45",
  "data": { }
}
```

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "No active campaign 42",
  "action_time": "2026-09-18T10:30:45",
  "data": "No active campaign 42"
}
```

Errors used in this document: `400` bad request (a rule or campaign that makes no sense, an unknown label, an unknown backfill aggregate), `401` missing/expired token, `403` wrong role, `404` not found (unknown id, unknown surface, no debug trace), `422` validation failure with a field map, `500` server error.

Part A — App-facing

The shared feed contract

A4, A5, A1 and A3 all answer with the same three things:

Field	Type	Description
`feedSessionId`	string	The session this page came from. **Send it back** in every interaction as `context.feedSessionId` — it is how a like is tied to the feed that showed the item
`items`	array	The items, already in display order
`nextCursor`	string | null	Pass it as `cursor` for the next page. `null` = nothing more right now

And every item carries:

Field	Type	Description
`position`	integer	The item's place in the feed, from 0. **Send it back** as `context.position`
`source`	string	Why this item is here. **Send it back** as `context.source`
`visibility`	string	`ALLOW`, or `INTERSTITIAL` → show a "sensitive content, tap to view" cover before the media

`source` **values**: `FOLLOWING`, `CELEBRITY`, `RECO`, `INTEREST`, `FALLBACK`, `SPONSORED`, `TRENDING`, `SEARCH`. Do not invent others — the interactions endpoint rejects unknown ones.

A1. Home feed

Purpose: The main feed. Mixes posts from people the user follows, recommended posts, and products.

Endpoint: `GET` `/api/v1/feed/home`

Access: User · **Authentication**: `Authorization: Bearer <access token>`

Replaces: `api/v1/e-social/feed`

Query Parameters

Parameter	Type	Required	Description	Validation	Default

`cursor`	string	No	Cursor from the previous page. Omit for the first page, and omit on pull-to-refresh to start a new session	opaque	none
`limit`	integer	No	Items per page	clamped to 1–30	15

Success Response JSON Sample


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Home feed",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "feedSessionId": "fs-77aa31c8",
    "items": [
      {
        "type": "POST",
        "position": 0,
        "source": "FOLLOWING",
        "visibility": "ALLOW",
        "post": { "postId": "4f2a77e1-0c8b-4d2f-9a11-7b6c3e5d8f90" },
        "product": null
      },
      {
        "type": "PRODUCT",
        "position": 1,
        "source": "RECO",
        "visibility": "ALLOW",
        "post": null,
        "product": { "productId": "8c31a0d2-77e4-4b1a-9f3c-2d5e6a7b8c90" }
      }
    ],
    "nextCursor": "eyJzIjoiZnMtNzdhYTMxYzgiLCJwIjoxNX0"
  }
}
```

Success Response Fields

Field	Description
`items[].type`	`POST` or `PRODUCT`. Exactly one of `post` / `product` is filled; the other is `null`
`items[].post`	A full `PostResponse` — the same object the rest of the API returns for a post
`items[].product`	A `ProductSummaryResponse` — the same object the marketplace list returns

Notes

- The feed is built once per session and paged from it, so a page read is fast and the order is stable while the user scrolls.
- Items are filtered again at page-read time, so something deleted, blocked or muted mid-scroll does not appear.
- If the feed store is unavailable the user gets a simpler feed from recent posts, never an error.

A2. New posts available ("the bubble")

Purpose: Tells the app whether to show the "N new posts" bubble at the top of the home feed.

Endpoint: `GET` `/api/v1/feed/home/has-new`

Access: User · **Authentication**: `Authorization: Bearer <access token>`

Replaces: the old feed SSE stream (`api/v1/e-social/feed/stream`)

Query Parameters

Parameter	Type	Required	Description	Validation
`since`	integer	**Yes**	`publishedAt` of the newest post the app is currently showing, as epoch **milliseconds**	must parse as a long

Success Response JSON Sample

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "New posts",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "count": 7,
    "more": false,
    "authors": [
      { "accountId": "…", "userName": "mariam", "profileMedia": { } }
    ],
    "nextPollIn": 300
  }
}
```


Success Response Fields

Field	Description
`count`	How many new posts, **capped at 20**
`more`	`true` when there were more than 20 — show "20+"
`authors`	Up to 3 authors, for the avatars on the bubble
`nextPollIn`	Seconds to wait before asking again. **Read this from the response, do not hard-code it** — the server lengthens it under load, with no app release

When to call it

Moment	Action
Feed comes on screen	Call once
Feed stays on screen	Call again after `nextPollIn` seconds
A `feed.new` event arrives on the `/api/v1/events` stream	Call immediately
`count > 0` and the user has scrolled down	Show the bubble
`count > 0` and the user is already at the top	No bubble — fetch and insert the posts
Bubble tapped	Call `GET /api/v1/feed/home` **with no cursor** and prepend the result above the existing list

> **The `feed.new` event.** The existing chat/events stream now carries an event named `feed.new` with an empty body on the account's personal channel. It carries no posts and no count — it only means "call `has-new` now". It is throttled to at most one a minute, and it is an optimisation: if the stream is down, the next poll finds the posts anyway.

A3. Reels feed

Purpose: The vertical clip feed.

Endpoint: `GET` `/api/v1/feed/reels`

Access: User · **Authentication**: `Authorization: Bearer <access token>`

Replaces: `api/v1/e-social/clip/feed`

Query Parameters

Parameter	Type	Required	Description	Validation	Default
`cursor`	string	No	Cursor from the previous page	opaque	none
`limit`	integer	No	Clips per page	clamped to 1–30	10

Success Response JSON Sample


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Reels",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "feedSessionId": "fs-91bd4410",
    "items": [
      {
        "position": 0,
        "source": "RECO",
        "clip": {
```

```
          "postId": "…",
          "visibility": "ALLOW",
          "media": { "id": "…", "durationMs": 14200, "variants": { "hls":
"https://…/master.m3u8" } }
        }
      }
    ],
    "nextCursor": "eyJzIjoiZnMtOTFiZDQ0MTAiLCJwIjoxMH0"
  }
}
```


Notes

- `clip` is the same `ClipResponse` the existing clip screen already renders — the reels screen does not need a new renderer.
- Here `visibility` sits **on the clip**, not on the item wrapper.
- Reels have their own "already served" memory: scrolling reels does not use up the home feed, and vice versa.

A4. Marketplace "For You"

Purpose: A personalised product feed for the marketplace tab.

Endpoint: `GET` `/api/v1/feed/marketplace`

Access: User · **Authentication**: `Authorization: Bearer <access token>`

Query Parameters

Parameter	Type	Required	Description	Validation	Default
`cursor`	string	No	Cursor from the previous page	opaque	none
`limit`	integer	No	Products per page	clamped to 1–30	20

Success Response JSON Sample

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Marketplace for you",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "feedSessionId": "fs-2210ab7e",
    "items": [
      { "position": 0, "source": "RECO", "visibility": "ALLOW", "product": { "productId": "…"
} },
      { "position": 1, "source": "SPONSORED", "visibility": "ALLOW", "product": { "productId":
"…" } }
    ],
    "nextCursor": null
  }
}
```


`source` **here means**: `RECO` (the model), `FOLLOWING` (a shop the user subscribes to), `SPONSORED` (a paid slot — label it in the UI), `FALLBACK` (trending, new shops, or products seen in posts).

A5. Events "For You"

Purpose: Event discovery.

Endpoint: `GET` `/api/v1/feed/events`

Access: User · **Authentication**: `Authorization: Bearer <access token>`

Query Parameters

Parameter	Type	Required	Description	Validation	Default
`cursor`	string	No	Cursor from the previous page	opaque	none
`limit`	integer	No	Events per page	clamped to 1–30	20

Success Response JSON Sample

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Events for you",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "feedSessionId": "fs-55c0d913",
    "items": [
      { "position": 0, "source": "FOLLOWING", "visibility": "ALLOW", "event": { "eventId": "…"
} }
    ],
    "nextCursor": "eyJzIjoiZnMtNTVjMGQ5MTMiLCJwIjoyMH0"
  }
}
```


`source` **here means**: `RECO`, `FOLLOWING` (an organiser the user follows), `FALLBACK` (upcoming events, by bookings). `event` is the usual `EventSummaryResponse`.

A6. Similar items

Purpose: The "Similar items" strip on a product page.

Endpoint: `GET` `/api/v1/feed/products/{productId}/similar`

Access: User · **Authentication**: `Authorization: Bearer <access token>`

Path Parameters

Parameter	Type	Required	Description
`productId`	UUID	Yes	The product being viewed

Query Parameters

Parameter	Type	Required	Description	Validation	Default
`limit`	integer	No	How many products	clamped to 1–50	20

Success Response JSON Sample

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Similar items",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "items": [
      { "source": "RECO", "visibility": "ALLOW", "product": { "productId": "…" } },
      { "source": "CATEGORY", "visibility": "ALLOW", "product": { "productId": "…" } }
    ]
  }
}
```


No paging and no `feedSessionId` — this is a strip, not a feed. `source` is `RECO` (the model's similar items) or `CATEGORY` (best sellers of the same category, used when the model has nothing).

A7. Suggested shops

Purpose: The "Shops you might like" strip.

Endpoint: `GET` `/api/v1/feed/shops/suggested`

Access: User · **Authentication**: `Authorization: Bearer <access token>`

Query Parameters

Parameter	Type	Required	Description	Validation	Default
`limit`	integer	No	How many shops	clamped to 1–30	10

Success Response JSON Sample


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Suggested shops",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "items": [
```

```
      {
        "source": "NEW",
        "shopId": "…",
        "shopName": "Kilimanjaro Coffee",
        "shopSlug": "kilimanjaro-coffee",
        "shopDescription": "Single-origin roasters",
        "logo": { },
        "isVerified": true,
        "city": "Arusha",
        "region": "Arusha",
        "subscriberCount": 1284
      }
    ]
  }
}
```


`source`: `RECO`, `NEW` (opened in the last 30 days) or `POPULAR` (most subscribers). **No contact details are returned** — the viewer does not follow this shop yet.

A8. Report interactions

Purpose: Send what the user did, in batches. This is the only way the recommendation model learns.

Endpoint: `POST` `/api/v1/feed/interactions/batch`

Access: ◫ User · **Authentication**: `Authorization: Bearer <access token>`

Success status: `202 Accepted`

> **Full definitions of every action live in `05_INTERACTIONS_CLIENT_GUIDE.md`.** That document is the contract; this entry is the reference. What your batch becomes once the backend has it is `10_EVENT_CATALOGUE.md` §5.

> **This is the only endpoint on the whole platform that puts anything on the event stream from the app.** Everything else the model learns — posts, products, shops, events, profiles, follows, blocks, orders, wishlists, and even

> likes and purchases — is published automatically from committed database rows. The app never announces that something exists or changed; it only reports what it alone can see: impressions, views, watch time, skips, clicks and searches.

Request JSON Sample


```
{
  "sessionId": "s-8812e0c4",
  "client": { "platform": "ANDROID", "appVersion": "2.4.1", "networkType": "CELLULAR" },
  "events": [
    {
      "clientEventId": "c-000184",
      "action": "IMPRESSION",
      "targetType": "POST",
      "targetId": "4f2a77e1-0c8b-4d2f-9a11-7b6c3e5d8f90",
      "occurredAt": "2026-09-18T10:22:31.900Z",
      "context": { "surface": "REELS", "position": 3, "feedSessionId": "fs-77aa", "source":
"RECO" }
    },
    {
      "clientEventId": "c-000185",
      "action": "VIEW",
      "targetType": "POST",
      "targetId": "4f2a77e1-0c8b-4d2f-9a11-7b6c3e5d8f90",
      "occurredAt": "2026-09-18T10:22:41.100Z",
      "watchMs": 9000,
      "mediaDurationMs": 6000,
      "loopCount": 1,
      "soundOn": true,
      "mediaIndex": 0,
      "context": { "surface": "REELS", "position": 3, "feedSessionId": "fs-77aa", "source":
"RECO" }
    }
  ]
}
```


Request Body Parameters

Parameter	Type	Required	Description	Validation
`sessionId`	string	No	The app session	—
`client.platform`	string	No	`ANDROID`, `IOS`, `WEB`	enum
`client.appVersion`	string	No	e.g. `2.4.1`	—
`client.networkType`	string	No	`WIFI`, `CELLULAR`, `OFFLINE`, `UNKNOWN`	enum
`events`	array	Yes	The batch	**max 100 per request**
`events[].clientEventId`	string	Yes	Your own id, unique within the batch	max 64 chars
`events[].action`	string	Yes	See the action table below	enum
`events[].targetType`	string	Yes*	`POST`, `PRODUCT`, `SHOP`, `EVENT`, `PROFILE`	*not required for `SEARCH`
`events[].targetId`	string	Yes*	UUID of the target	must be a UUID
`events[].occurredAt`	string	Yes	When it happened, on the device	ISO 8601
`events[].dwellMs` / `watchMs`	integer	No	Time on screen / watched	0 – 6 h
`events[].mediaDurationMs`	integer	No	Length of the media	0 – 12 h
`events[].loopCount`	integer	No	Replays	0 – 1000
`events[].soundOn`	boolean	No	Was sound on	—
`events[].mediaIndex`	integer	No	Which item of a carousel	0 – 100
`events[].context.surface`	string	No	`FEED`, `REELS`, `SEARCH`, `PRODUCT_PAGE`, `SHOP_PAGE`, `EVENT_PAGE`, `PROFILE_PAGE`, `RECOMMENDATION`, `NOTIFICATION`, `CHAT`	enum
`events[].context.position`	integer	No	The item's position in the feed	—
`events[].context.feedSessionId`	string	No	From the feed response	—
`events[].context.source`	string	No	From the item	enum
`events[].context.viaPostId`	string	No	The post a product was reached through	UUID
`events[].context.query`	string	No	The search text, for `SEARCH`	—

Unknown fields are ignored, deliberately: a newer app adding a field must never lose the batch.

Actions the app sends

`IMPRESSION`, `VIEW`, `SKIP`, `CLICK`, `SEARCH`, `NOT_INTERESTED`, `HIDE`.

Actions the app must NOT send

`LIKE`, `UNLIKE`, `SHARE`, `BOOKMARK`, `COMMENT`, `ADD_TO_CART`, `REMOVE_FROM_CART`, `PURCHASE`, `BLOCK`, `REPORT`, `MUTE_AUTHOR`.

The backend sends these itself when the change commits, so an app that also sent them would count every like twice. They come back rejected with `SENT_BY_SERVER`. Call the normal feature API (the like endpoint, the cart endpoint) and the interaction is recorded for you.

Success Response JSON Sample

```
{
  "success": true,
  "httpStatus": "ACCEPTED",
  "message": "Interactions received",
  "action_time": "2026-09-18T10:30:45",
  "data": { "accepted": 2, "dropped": 0, "rejected": [] }
}
```

Field	Description
`accepted`	Kept
`dropped`	Valid but not kept (rate limit, or the event log was briefly unavailable). **Do not resend** — these are signal, not records
`rejected`	Invalid. **Do not resend.** Each entry is `{ clientEventId, reason }` — a rejection is a bug in the app; log it in debug builds

Rejection reasons

`CLIENT_EVENT_ID_REQUIRED_MAX_64`, `UNKNOWN_ACTION`, `SENT_BY_SERVER`, `UNKNOWN_TARGET_TYPE`, `TARGET_ID_NOT_UUID`, `DUPLICATE_CLIENT_EVENT_ID`, `OCCURRED_AT_NOT_ISO_8601`, `DURATION_OUT_OF_RANGE`, `LOOP_COUNT_OUT_OF_RANGE`, `MEDIA_INDEX_OUT_OF_RANGE`, `UNKNOWN_SURFACE`, `UNKNOWN_SOURCE`, `POSITION_OUT_OF_RANGE`, `VIA_POST_ID_NOT_UUID`, `BATCH_LIMIT_100`, `EMPTY_EVENT`.

Part B — Admin & operations

Everything below is on `/api/admin/feed`. A signed-in account without the role gets `403`.

Sponsored campaigns

A sponsored campaign places one product at fixed slots in **home and marketplace** sessions, while it is inside its window. A viewer sees the same campaign at most `feed.sponsored.capPerDay` times in 24 hours (a `feed_params` setting), so a campaign cannot follow someone down the page.

Campaigns and rules are re-read from the database **at most every 60 seconds**, so a change can take up to a minute to show up in feeds.

> **Not built yet**: billing, budgets and self-serve. A campaign today is created by staff and runs between two dates, free.

B1. Start a campaign

Endpoint: `POST` `/api/admin/feed/sponsored` · **Access**: Admin

Request JSON Sample

```
{
  "productId": "8c31a0d2-77e4-4b1a-9f3c-2d5e6a7b8c90",
  "startsAt": "2026-10-01T00:00:00Z",
  "endsAt": "2026-10-15T00:00:00Z"
}
```

Parameter	Type	Required	Validation
`productId`	UUID	Yes	the product must exist, else `400 Product not found`
`startsAt`	ISO 8601 instant	Yes	—
`endsAt`	ISO 8601 instant	Yes	must be after `startsAt`, else `400 endsAt must be after startsAt`

Success Response

```
{
  "success": true, "httpStatus": "OK", "message": "Campaign added",
  "action_time": "2026-09-18T10:30:45",
  "data": { "id": 42, "productId": "8c31a0d2-…", "startsAt": "2026-10-01T00:00:00Z", "endsAt":
"2026-10-15T00:00:00Z", "active": false }
}
```


`active` is `true` only while the campaign is inside its window.

B2. List campaigns

Endpoint: `GET` `/api/admin/feed/sponsored` · **Access**: Admin

Returns an array of the object above — running and upcoming campaigns.

B3. Stop a campaign

Endpoint: `DELETE` `/api/admin/feed/sponsored/{id}` · **Access**: Admin

Parameter	Type	Required	Description
`id`	integer	Yes	The campaign id from B1/B2

Ends it immediately. An id that is not an active campaign gives `404 No active campaign {id}`.

Business rules

A business rule multiplies the feed score of everything matching it, for a period. It is the lever for "push this category this week" or "quieten this shop while we investigate" — without touching code. **Several matching rules multiply together**, and like campaigns they are re-read at most every 60 seconds.

B4. Add a rule

Endpoint: `POST` `/api/admin/feed/business-rules` · **Access**: Admin

Request JSON Sample

```
{
  "matchType": "CATEGORY",
  "matchValue": "b81d9a34-5f2c-4c7e-9a10-3e5d7f9b1c22",
  "multiplier": 1.4,
  "startsAt": "2026-10-01T00:00:00Z",
  "endsAt": "2026-10-08T00:00:00Z"
}
```

Parameter	Type	Required	Description	Validation
`matchType`	string	Yes	`PRODUCT`, `SHOP`, `CATEGORY` (all three match products) or `ACCOUNT` (matches posts by, and events organised by, that account)	enum
`matchValue`	string	Yes	The id of that product / shop / category / account	36-character UUID
`multiplier`	number	Yes	Above 1 boosts, below 1 suppresses	**0.1 – 3.0**
`startsAt`	ISO 8601 instant	Yes	—	—
`endsAt`	ISO 8601 instant	Yes	—	must be after `startsAt`

Success Response

```
{
  "success": true, "httpStatus": "OK", "message": "Rule added",
  "action_time": "2026-09-18T10:30:45",
  "data": { "id": 17, "matchType": "CATEGORY", "matchValue": "b81d9a34-…", "multiplier": 1.4,
            "startsAt": "2026-10-01T00:00:00Z", "endsAt": "2026-10-08T00:00:00Z" }
}
```

B5. List rules

Endpoint: `GET` `/api/admin/feed/business-rules` · **Access**: Admin

Running and upcoming rules, as an array of the object above. Expired rules are not listed.

B6. End a rule now

Endpoint: DELETE `/api/admin/feed/business-rules/{id}` · **Access**: ◫ Admin

`404 No running or upcoming rule {id}` if it has already finished.

B7. Why did this person see this feed?

Purpose: The per-account debug trace. For support and for tuning: it shows every item the last feed build produced, and **every item it threw away and why**.

Endpoint: GET `/api/admin/feed/debug/{accountId}` · **Access**: ◫ Admin

Path & Query Parameters

Parameter	Type	Required	Description	Default
`accountId`	UUID	Yes	Whose feed	—
`surface`	string	No	`HOME`, `REELS`, `MARKET`, `EVENTS`	`HOME`

How to switch it on

Tracing is **off** unless the account is listed. Add the account id to the `feed.debug.accounts` row in the `feed_params` table, then have that person (or a test account) open the feed. The trace is kept for **1 hour**.

Success Response JSON Sample


```
{
  "success": true, "httpStatus": "OK", "message": "Last session trace",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "surface": "HOME",
    "builtAt": 1789412345678,
    "shown": [
      {
        "position": 0, "type": "POST", "id": "4f2a77e1-…", "source": "FOLLOWING",
        "baseScore": 1.81, "score": 2.35, "visibility": "ALLOW",
        "trace": ["NewShopBoost x1.3"]
      }
```

```
    ],
    "removed": [ { "type": "POST", "id": "9b10…", "source": "FALLBACK", "filter": "Seen" } ]
  }
}
```

Errors

Status	When
`404 Unknown surface {x}`	`surface` is not one of the four
`404 No trace: add the account to feed_params 'feed.debug.accounts', then open that feed (kept 1 h)`	Tracing was off, or the trace expired

Moderation labels

A label is a moderation outcome attached to a post, product, shop or account. The feed reads labels; it has no special cases of its own. **Labels do not delete anything** — they decide who stops seeing it.

The rule (first match wins, and strangers are treated more strictly than followers, because following was the user's choice and discovery is ours):

Label	Viewer follows the author	Viewer does not follow
`suspended` (account or item), `prohibited_item`	removed	removed
`counterfeit_suspected`	shown	removed
`spam_suspected` (account)	shown	removed
`reported_threshold`	shown	removed
`low_quality_media`	shown	removed
`sensitive_media`	shown behind a cover (`INTERSTITIAL`)	removed

`reported_threshold` is applied automatically when a post passes the report threshold. The rest are applied by staff, by a rule, or by a model.

B8. Add a label

Endpoint: POST `/api/admin/feed/labels` · **Access**: Admin

Request JSON Sample

```
{
  "targetType": "POST",
  "targetId": "4f2a77e1-0c8b-4d2f-9a11-7b6c3e5d8f90",
  "label": "sensitive_media",
  "expiresInHours": 72
}
```

Parameter	Type	Required	Description	Validation
`targetType`	string	Yes	`POST`, `PRODUCT`, `SHOP`, `ACCOUNT`	else `400 Unknown target type`
`targetId`	UUID	Yes	What to label	—
`label`	string	Yes	`suspended`, `spam_suspected`, `counterfeit_suspected`, `sensitive_media`, `prohibited_item`, `reported_threshold`, `low_quality_media`	else `400 Unknown label '…'`
`expiresInHours`	integer	No	**Omit for a permanent label**	must be positive

B9. Remove a label

Endpoint: `DELETE` `/api/admin/feed/labels/{labelId}` · **Access**: ◫ Admin

`404` if that label id does not exist.

B10. Labels on one target

Endpoint: `GET` `/api/admin/feed/labels?targetType={type}&targetId={id}` · **Access**: ◫ Admin

Success Response JSON Sample

```
{
  "success": true, "httpStatus": "OK", "message": "Labels",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "targetType": "POST",
    "targetId": "4f2a77e1-…",
    "active": ["sensitive_media"],
    "history": [
      {
```

```
        "id": 91, "label": "sensitive_media", "source": "MANUAL", "created_by": "…",
        "created_at": "2026-09-18T09:00:00Z", "expires_at": "2026-09-21T09:00:00Z",
"removed_at": null
      }
    ]
  }
}
```


`active` is what is in force now; `history` includes removed and expired labels, so a decision can be explained later. `source` is `MANUAL`, `RULE`, `REPORT_THRESHOLD` or `MODEL`.

Interest mapping

Interests are the shared language between social, marketplace and events: one person has one interest profile, and a like on a post can move what they see in the shop. For that to work, **every product category, event category and popular hashtag must be mapped to interests**.

> ⚠ **This mapping is not finished.** Use B12 to see what is still missing. Until a category is mapped, its items reach people only through follows, trending and fallbacks — not through interests.

B11. The interest list and all mappings

Endpoint: `GET` `/api/admin/feed/interests` · **Access**: Admin


```
{
  "success": true, "httpStatus": "OK", "message": "Interest mappings",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "interests": [ { "id": "…", "name": "Fashion" } ],
    "productCategories": [ { "key": "b81d9a34-…", "name": "Sneakers", "interestIds": ["…"] }
],
    "eventCategories":   [ { "key": "…", "name": "Live music", "interestIds": ["…"] } ],
    "hashtags":          [ { "key": "mitumba", "name": "#mitumba", "interestIds": ["…"] } ]
  }
}
```

For a hashtag, `key` is the tag **without** the `#`.

B12. What still needs mapping

Endpoint: `GET` `/api/admin/feed/interests/unmapped` · **Access**: Admin

```
{
  "success": true, "httpStatus": "OK", "message": "Still to map",
  "action_time": "2026-09-18T10:30:45",
  "data": {
    "productCategories": [ { "id": "…", "name": "Sneakers" } ],
    "eventCategories":   [ { "id": "…", "name": "Workshops" } ],
    "hashtags":          [ { "hashtag": "mitumba", "posts": 812 } ]
  }
}
```

A product category counts as mapped when a parent category is mapped. Hashtags listed are the most used unmapped tags on public posts of the last 30 days — the ones worth mapping first.

B13–B15. Map a category or hashtag

Endpoints: `PUT` `/api/admin/feed/interests/product-categories/{categoryId}` · `/event-categories/{categoryId}` · `/hashtags/{hashtag}` · **Access**: Admin

Request JSON Sample

```
{ "interestIds": ["a1b2c3d4-…", "e5f6a7b8-…"] }
```

Parameter	Type	Required	Description	Validation
`interestIds`	array of UUID	Yes	**Replaces** the whole mapping for that key. An **empty array removes** the mapping	at most **5**; every id must exist, else `404 Unknown interest id in […]`

For `/hashtags/{hashtag}`, pass the tag with or without `#` — it is normalised. A value that is not a usable tag gives `404 Not a hashtag: …`. An unknown category id gives `404 … not found: {id}`.

Backfills

A backfill republishes data that already exists into the feed's event log — used when a new field is added, after an incident, or when a topic is rebuilt from scratch. Existing rows are read and re-sent; **nothing is modified**.

B16. Start a backfill

Endpoint: `POST` `/api/admin/feed/backfills` · **Access**: Super admin

Request JSON Sample

```
{ "aggregate": "product" }
```

Parameter	Type	Required	Description	Validation
`aggregate`	string	Yes	`post`, `follow`, `block`, `product`, `shop`, `event`, `profile`, `order`, `wishlist`, or `all`	else `400 Unknown aggregate '…'`

Only one run per aggregate at a time. Asking again while one is pending or running returns that existing run instead of starting a second.

B17. Recent runs

Endpoint: `GET` `/api/admin/feed/backfills` · **Access**: Super admin

```
{
  "success": true, "httpStatus": "OK", "message": "Backfill runs",
  "action_time": "2026-09-18T10:30:45",
  "data": [
    {
      "id": 7, "aggregate": "product", "status": "RUNNING", "lastKey": "8c31a0d2-…",
      "keysDone": 18400, "keysFailed": 0, "requestedBy": "…",
      "createdAt": "2026-09-18T10:00:00Z", "startedAt": "2026-09-18T10:00:02Z",
      "finishedAt": null, "error": null
    }
  ]
}
```

Field	Description
`status`	`PENDING`, `RUNNING`, `DONE`, `CANCELLED`, `FAILED`

Field	Description
`lastKey`	Where it has reached — a cancelled or failed run resumes from here rather than starting over
`keysDone` / `keysFailed`	Progress
`error`	Why it failed, when it did

B18. Cancel a run

Endpoint: POST `/api/admin/feed/backfills/{runId}/cancel` · **Access**: □□ Super admin

Stops the run at its current key. `404` if that run id does not exist.

Related documents

Doc	What it covers
`08_AMENDED_ENDPOINTS.md`	Existing endpoints that changed, and the old endpoints being deleted
`07_FEED_ENDPOINTS_GUIDE.md`	The app team's walkthrough: paging, the bubble, failure behaviour, switch checklist
`05_INTERACTIONS_CLIENT_GUIDE.md`	Exact definition of every interaction the app sends
`06_RECOMMENDATION_DEVELOPER_GUIDE.md`	For the AI developer: topics, and the lists the feed reads back
`01_ARCHITECTURE.md`	The design behind all of it

Revision #2
Created 18 September 2026 04:22:59 by Admin Qbit
Updated 18 September 2026 04:41:09 by Admin Qbit