# In-App Notifications (The Bell)

**Author**: Josh S. Sakweli, Backend Lead Team **Last Updated**: 2026-09-18 **Version**: v1.0

**Base URL**: `http://localhost:8765/api/v1` (local) — `https://dev.api.nexgate.co/api/v1` (staging)

> **This document is ONLY for the in-app notification bell** — the □□ icon, its red badge, and the list that opens when you tap it. It is **not** about push notifications on the lock screen, email, SMS, or chat unread badges. Those are different systems; Part 0 tells you which one you need.
>
> **v1.0** — verified end to end on staging on 2026-09-18: one account liked another's post, the owner's badge went from 0 to 1, and the bell list showed "joshdoe liked your post" pointing at the right post. Every JSON sample below is a real staging response.

**Short Description**: Everything that happens to a user while they are not looking — a like, a new follower, an order shipped, a ticket bought — is saved as one row in their bell list. This API reads that list, counts unread rows for the badge, marks rows read, and deletes them. The server also sends a live `notification.added` event on the existing event stream so the badge moves without the app polling.

**Hints**:

- Every endpoint needs `Authorization: Bearer <accessToken>`. A user only ever sees their own rows — there is no way to read someone else's bell.
- **Pages start at 1, not 0.** `page=0` returns `400 "Page index must not be less than zero"`.
- **The row's `title` is the whole sentence** ("joshdoe liked your post"). `message` is an optional second line (a snippet of the post or comment). It is an empty string when there is nothing to quote, e.g. a photo-only post. Never show an empty grey line.
- **`createdAt` and `readAt` carry no timezone. They are UTC.** Append `Z` before parsing, or "2 minutes ago" will be off by your offset.
- **Tapping a row opens `targetType` + `targetId`.** If your app does not know the `targetType`, stay on the bell list. That rule is what lets the backend add new notification types without breaking old app versions (Part 3).
- Ignore unknown keys inside `data`. It carries values the server used to build the sentence; some of them are internal.

---

# Part 0 — Is this the right document?

| You are building… | System | Where |
|---|---|---|
| The □□ icon, its badge, the list of "X liked your post" rows | **In-app notifications (this doc)** | `/api/v1/notifications/*` + `notification.added` on `/api/v1/events` |
| A banner on the lock screen when the app is closed | Push (Firebase) | Not live yet. Needs the app to register its device token first — separate doc |
| Unread counts on chat threads | Chat | `chatbox-api-doc.md` — `unread.changed` |
| Email and SMS | Notification server | Nothing for the app to do |

The same event can reach a user through several of these at once. A like creates **one bell row** and, once push is live, **one push**. They are independent: reading the bell row does not clear the push, and vice versa.

---

# Part 1 — How the bell works

```
somebody likes your post
          │
          ▼
backend saves one row in YOUR bell list  ──────►  row: isRead=false
          │
          ▼
backend sends on /api/v1/events:
   event: notification.added
   data:  {"type":"SOC_POST_LIKED","unreadCount":4}
          │
          ▼
app sets the badge to 4  (no request needed)
if the bell list is open → refetch page 1
```

## What the app does, screen by screen

| Moment | Call | Why |
|---|---|---|
| App starts or comes back to the foreground | `GET /notifications/unread-count` | The event stream was not connected while you were away; this is the truth |

| Moment | Call | Why |
|---|---|---|
| `notification.added` arrives | none — set the badge to `unreadCount` from the event | The event already carries the new count |
| User opens the bell | `GET /notifications/me?page=1&size=20` | Newest first |
| User scrolls to the bottom | `GET /notifications/me?page=2…` while `hasNext` is `true` | |
| User taps a row | `PUT /notifications/{id}/read`, then open `targetType`/`targetId` | Mark read first so the badge is right when they come back |
| "Mark all as read" button | `PUT /notifications/read-all` | Then set the badge to 0 locally |
| Swipe to delete | `DELETE /notifications/{id}` | |
| "Clear read" button | `DELETE /notifications/read` | Removes every row already read |

# Things the server does NOT tell you live

- **Marking read sends no event.** If the same user has the app open on a phone and a tablet and reads on one, the other keeps its old badge until it next calls `unread-count`. Call it whenever the app returns to the foreground.
- **Deleting sends no event** either. Update your list locally.

# The live event

The bell rides the **same** stream the chat uses — `GET /api/v1/events` (see `chatbox-api-doc.md` → "SSE stream" for connecting, reconnecting with `Last-Event-ID`, and why web needs fetch-based streaming). Do not open a second stream for notifications.

| Event | Data | Meaning |
|---|---|---|
| `notification.added` | `{"type": "SOC_POST_LIKED", "unreadCount": 4}` | A row was added to this user's bell. `unreadCount` is the new total, counted after the row was saved |

`type` lets you do something special for a few types (a sound for a new order, for example). For everything else, just update the badge.

---

# Part 2 — The notification object

Every list and single-row endpoint returns rows in this shape. Real staging row:

```
{
  "id": "6e902bb6-9b38-49c6-825e-22b951f39584",
  "userId": "02db7d92-b426-47b2-9171-cf15339c1376",
  "shopId": null,
  "serviceId": "2b5f3f0c-6d6e-4c4a-a10a-34b622a1fc68",
  "serviceType": "SOCIAL",
  "targetType": "POST",
  "targetId": "2b5f3f0c-6d6e-4c4a-a10a-34b622a1fc68",
  "title": "joshdoe liked your post",
  "message": "",
  "type": "SOC_POST_LIKED",
  "priority": "LOW",
  "isRead": false,
  "data": {
    "type": "SOC_POST_LIKED",
    "actor": { "name": "joshdoe" },
    "postId": "2b5f3f0c-6d6e-4c4a-a10a-34b622a1fc68",
    "targetId": "2b5f3f0c-6d6e-4c4a-a10a-34b622a1fc68",
    "targetType": "POST",
    "mailAccount": "general"
  },
  "createdAt": "2026-09-18T19:18:40.736493",
  "readAt": null
}
```


| Field | Type | Description |
|---|---|---|
| `id` | UUID | The row. Use it to mark read or delete |
| `userId` | UUID | The owner — always the signed-in user |
| `shopId` | UUID / null | Set when the row is about one of the user's shops (new order, low stock…) |
| `serviceId` | string | **Legacy — do not use.** Kept for older apps. Use `targetId` |
| `serviceType` | string | Which area of the app it belongs to — use it for filter tabs. See values |
| `targetType` | string | Which screen a tap opens. See Part 3 |
| `targetId` | string / null | The id that screen needs. `null` for screens that need none (e.g. `WALLET`) |

| Field | Type | Description |
|---|---|---|
| `title` | string | **The sentence to show.** Always present |
| `message` | string | Optional second line — up to ~80 characters of the post, comment or review. **Can be `""` (e.g. a photo-only post) — hide the line when empty** |
| `type` | string | The exact notification type, e.g. `SOC_POST_LIKED`, `ORD_SHIPPED`. 136 exist; new ones will be added. Do not switch on it for navigation — use `targetType` |
| `priority` | string | `LOW`, `NORMAL`, `MEDIUM`, `HIGH`, `URGENT`. Use it for styling only (e.g. bold `HIGH`/`URGENT` rows) |
| `isRead` | boolean | `false` until the user reads it |
| `data` | object | The values used to build the sentence. Useful for avatars (`actor.name`) or extra text. **Ignore keys you do not know** |
| `createdAt` | string | When it happened. ISO 8601, **UTC, no offset** |
| `readAt` | string / null | When it was marked read. UTC, no offset |

# `serviceType` values

For filter tabs ("All", "Social", "Orders"…) and for endpoint 5.

| Value | Covers |
|---|---|
| `SOCIAL` | Likes, comments, follows, mentions, reposts |
| `CHAT` | Group invites, join requests, offers, missed calls, meetings |
| `LIVE` | Live streams and Spaces |
| `ORDER` | Orders you bought or sold |
| `SHOP` | Your shop: applications, stock, reviews, WhatsApp setup |
| `CART` | Items in your cart |
| `CHECKOUT` | Checkout sessions |
| `PAYMENT` | Payments made and received |
| `WALLET` | Wallet top-ups, balance, payouts |

| Value | Covers |
|---|---|
| `INSTALLMENT` | Installment plans and dues |
| `GROUP_PURCHASE` | Group buying |
| `EVENT` | Events, tickets, bookings |
| `USER` | Your account: security, sessions, devices |
| `PROMOTIONAL` | Offers and announcements |
| `ADMIN` | Only admins ever receive these |

---

# Part 3 — Where a tap goes (`targetType`)

Map each `targetType` you support to a screen, passing `targetId` when the table says it carries one. **Anything else — a `targetType` you have not mapped, or `CUSTOM` / `NONE` — stays on the bell list, with the row marked read.** Never crash and never show an error on an unknown value; the backend adds new values over time.

| `targetType` | `targetId` is… | Opens |
|---|---|---|
| `POST` | post id | The post |
| `COMMENT` | comment id | The comment, inside its post |
| `POLL` | poll id | The poll |
| `STREAM` | stream id | A live stream |
| `PROFILE` | account id | Somebody's profile |
| `FOLLOW_REQUESTS` | — | Pending follow requests |
| `REPORT_HISTORY` | — | The user's reports |
| `MODERATION_NOTICE` | report id | A moderation decision |
| `CONVERSATION` | conversation id | A chat thread |
| `CHAT_LIST` | — | The chat list |
| `GROUP_INVITES` | conversation id | Group invites |
| `GROUP_JOIN_REQUESTS` | conversation id | Join requests for a group you manage |
| `CALL` | call id | A call (e.g. missed-call details) |
| `MEETING` | meeting id | A meeting |
| `SPACE` | space id | A Space |

| `targetType` | `targetId` is… | Opens |
|---|---|---|
| `OFFER` | offer id | A personalised offer in chat |
| `SHOP_INBOX` | shop id | The shop's chat inbox |
| `ORDER` | order id | An order you bought |
| `SHOP_ORDER` | order id | An order your shop received |
| `ORDER_TRACKING` | order id | Delivery tracking |
| `CART` | — | The cart |
| `PRODUCT` | product id | A product |
| `INVENTORY` | shop id | The shop's stock screen |
| `DISPUTE` | dispute id | A dispute |
| `REVIEW` | review id | A review |
| `GROUP_PURCHASE` | group id | A group purchase |
| `AGREEMENT` | agreement id | An installment agreement |
| `SHOP` | shop id | A shop's public page |
| `SHOP_DASHBOARD` | shop id | Your shop's dashboard |
| `SHOP_APPLICATION` | shop id | Your shop application |
| `WABA_SETTINGS` | shop id | The shop's WhatsApp settings |
| `DOWNLOAD` | order id | Digital download for an order |
| `TRANSACTION` | transaction id | One transaction |
| `TRANSACTION_HISTORY` | — | All transactions |
| `CHECKOUT` | session id | A checkout session |
| `WALLET` | — | The wallet |
| `PAYOUT_SETTINGS` | — | Payout settings |
| `OTP_SCREEN` | — | The code entry screen |
| `EVENT` | event id | An event |
| `EVENT_DASHBOARD` | event id | Organiser's dashboard for an event |
| `EVENTS_LIST` | — | The events list |
| `TICKET` | ticket id | A ticket |
| `BOOKING` | booking id | A booking |
| `MY_BOOKINGS` | booking id | My bookings, scrolled to that booking |
| `CLAIM` | claim id | An organiser's fund claim |
| `SCANNER_MODE` | event id | The ticket scanner for an event |

| `targetType` | `targetId` is… | Opens |
|---|---|---|
| `EVENT_REVIEW` | event id | Review an event you attended |
| `HOME` | — | Home |
| `ONBOARDING` | — | Finish onboarding |
| `ACCOUNT_SETTINGS` | — | Account settings |
| `SECURITY_SETTINGS` | — | Security settings |
| `ACTIVE_SESSIONS` | — | Signed-in sessions |
| `CHAT_DEVICES` | — | Chat devices |
| `APPEAL_FORM` | — | Appeal a decision |
| `ADMIN_*` (7 values) | varies | Admin panel only — the consumer apps can treat these as unknown |
| `CUSTOM`, `NONE` | — | Stay on the bell list |

**If the target no longer exists** (the post was deleted, for example), the screen's own endpoint returns `404`. Show that screen's normal "not available" state; do not delete the bell row automatically.

---

# Standard Response Format

All API responses follow a consistent structure using our Globe Response Builder pattern:

## Success Response Structure

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Operation completed successfully",
  "action_time": "2026-09-18T19:23:04.211589898",
  "data": { }
}
```

## Error Response Structure

```
{
  "success": false,
```

```
  "httpStatus": "BAD_REQUEST",
  "message": "Error description",
  "action_time": "2026-09-18T19:23:06.226159977",
  "data": "Error description"
}
```

# Standard Response Fields

| Field | Type | Description |
|---|---|---|
| `success` | boolean | Always `true` for successful operations, `false` for errors |
| `httpStatus` | string | HTTP status name (OK, BAD_REQUEST, NOT_FOUND, etc.) |
| `message` | string | Human-readable message describing the operation result |
| `action_time` | string | ISO 8601 timestamp of when the response was generated |
| `data` | object/string | Response payload for success, error details for failures |

Responses may also carry `action` and `context` (always `null` here) — ignore them.

# The page object

Endpoints 1, 2, 5 and 6 return this inside `data`:

```
{
  "notifications": [ /* notification objects, newest first */ ],
  "currentPage": 1,
  "pageSize": 20,
  "totalElements": 1,
  "totalPages": 1,
  "hasNext": false,
  "hasPrevious": false,
  "isFirst": true,
  "isLast": true
}
```

| Field | Description |
|---|---|
| `notifications` | The rows, newest first |

| Field | Description |
|---|---|
| `currentPage` | The page you asked for (starts at 1) |
| `pageSize` | Rows per page |
| `totalElements` | Rows across all pages |
| `totalPages` | Number of pages |
| `hasNext` | Ask for `currentPage + 1` while this is `true` |
| `hasPrevious`, `isFirst`, `isLast` | Convenience flags |

> New rows can arrive while the user scrolls, which shifts every later page by one. If you see the same `id` twice, keep one.

---

# HTTP Method Badge Standards

- **GET** - GET - Green (Safe, read-only operations)
- **PUT** - PUT - Yellow (Update)
- **DELETE** - DELETE - Red (Remove resources)

---

# Endpoints

| # | Method | Path | Use it for |
|---|---|---|---|
| 1 | GET | `/notifications/me` | The bell list |
| 2 | GET | `/notifications/unread` | Only unread rows |
| 3 | GET | `/notifications/unread-count` | The badge |
| 4 | GET | `/notifications/summary` | Total / unread / read counts |
| 5 | GET | `/notifications/service/{serviceType}` | A filter tab |
| 6 | GET | `/notifications/shop/{shopId}` | A shop owner's shop notifications |
| 7 | GET | `/notifications/{notificationId}` | One row |
| 8 | PUT | `/notifications/{notificationId}/read` | Mark one read |
| 9 | PUT | `/notifications/read` | Mark several read |

| # | Method | Path | Use it for |
|---|---|---|---|
| 10 | PUT | `/notifications/read-all` | Mark everything read |
| 11 | DELETE | `/notifications/{notificationId}` | Delete one |
| 12 | DELETE | `/notifications/batch` | Delete several |
| 13 | DELETE | `/notifications/read` | Delete every read row |

---

# 1. List My Notifications

**Purpose**: The bell list — every row for the signed-in user, read and unread, newest first.

**Endpoint**: `GET` `{base_url}/notifications/me`

**Access Level**: □□ Protected

**Authentication**: Bearer Token

**Request Headers**:

| Header | Type | Required | Description |
|---|---|---|---|
| Authorization | string | Yes | `Bearer <accessToken>` |

**Query Parameters**:

| Parameter | Type | Required | Description | Validation | Default |
|---|---|---|---|---|---|
| page | integer | No | Page number | Min: 1 | 1 |
| size | integer | No | Rows per page | Min: 1 | 20 |

**Success Response JSON Sample** (staging, 2026-09-18):


```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Notifications retrieved successfully",
  "action_time": "2026-09-18T19:23:04.211589898",
  "data": {
    "notifications": [
      {
        "id": "6e902bb6-9b38-49c6-825e-22b951f39584",
```

```
        "userId": "02db7d92-b426-47b2-9171-cf15339c1376",
        "shopId": null,
        "serviceId": "2b5f3f0c-6d6e-4c4a-a10a-34b622a1fc68",
        "serviceType": "SOCIAL",
        "targetType": "POST",
        "targetId": "2b5f3f0c-6d6e-4c4a-a10a-34b622a1fc68",
        "title": "joshdoe liked your post",
        "message": "",
        "type": "SOC_POST_LIKED",
        "priority": "LOW",
        "isRead": false,
        "data": { "actor": { "name": "joshdoe" }, "postId": "2b5f3f0c-6d6e-4c4a-a10a-
34b622a1fc68" },
        "createdAt": "2026-09-18T19:18:40.736493",
        "readAt": null
      }
    ],
    "currentPage": 1,
    "pageSize": 5,
    "totalElements": 1,
    "totalPages": 1,
    "hasNext": false,
    "hasPrevious": false,
    "isFirst": true,
    "isLast": true
  }
}
```


**Success Response Fields**: see the page object and the notification object. With no rows, `notifications` is `[]` and `message` is `"No notifications found"`.

**Error Response JSON Sample** (staging, `page=0`):


```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Page index must not be less than zero",
  "action_time": "2026-09-18T19:23:06.226159977",
  "data": "Page index must not be less than zero"
}
```

**Standard Error Types**:

- `400 BAD_REQUEST`: `page` is 0 or negative
- `401 UNAUTHORIZED`: Token missing, invalid or expired

---

# 2. List Unread Notifications

**Purpose**: Only rows with `isRead=false`, newest first — for an "Unread" tab.

**Endpoint**: `GET` `{base_url}/notifications/unread`

**Access Level**: Protected

**Authentication**: Bearer Token

**Query Parameters**:

| Parameter | Type | Required | Description | Validation | Default |
|---|---|---|---|---|---|
| page | integer | No | Page number | Min: 1 | 1 |
| size | integer | No | Rows per page | Min: 1 | 20 |

**Success Response**: identical in shape to endpoint 1.

> Marking a row read removes it from this list, so every later page shifts up by one. If you mark rows read while the user scrolls this tab, refetch from page 1 instead of asking for the next page.

**Standard Error Types**:

- `400 BAD_REQUEST`: `page` is 0 or negative
- `401 UNAUTHORIZED`: Token missing, invalid or expired

---

# 3. Get Unread Count

**Purpose**: The number on the badge.

**Endpoint**: `GET` `{base_url}/notifications/unread-count`

**Access Level**: 🔒 Protected

**Authentication**: Bearer Token

**Success Response JSON Sample** (staging):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Unread count retrieved successfully",
  "action_time": "2026-09-18T19:23:05.489594598",
  "data": { "unreadCount": 1 }
}
```

**Success Response Fields**:

| Field | Description |
|---|---|
| unreadCount | Rows with `isRead=false`. Show `99+` above 99 |

Call it when the app starts and every time it returns to the foreground. While the app is open, `notification.added` keeps the badge current without calling this.

**Standard Error Types**:

- `401 UNAUTHORIZED`: Token missing, invalid or expired

---

# 4. Get Summary

**Purpose**: Total, unread and read counts in one call.

**Endpoint**: `GET` `{base_url}/notifications/summary`

**Access Level**: 🔒 Protected

**Authentication**: Bearer Token

**Success Response JSON Sample** (staging):

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Notification summary retrieved successfully",
```

```
  "action_time": "2026-09-18T19:23:04.820411225",
  "data": { "total": 1, "unread": 1, "read": 0 }
}
```

**Success Response Fields**:

| Field | Description |
|---|---|
| total | All rows |
| unread | Rows with `isRead=false` — same number as endpoint 3 |
| read | `total - unread` |

**Standard Error Types**:

- `401 UNAUTHORIZED`: Token missing, invalid or expired

---

# 5. List by Area (`serviceType`)

**Purpose**: One filter tab — e.g. only `SOCIAL`, or only `ORDER`.

**Endpoint**: `GET` `{base_url}/notifications/service/{serviceType}`

**Access Level**: Protected

**Authentication**: Bearer Token

**Path Parameters**:

| Parameter | Type | Required | Description | Validation |
|---|---|---|---|---|
| serviceType | string | Yes | The area | One of the `serviceType` values, upper case. An unknown value returns an empty list, not an error |

**Query Parameters**:

| Parameter | Type | Required | Description | Validation | Default |
|---|---|---|---|---|---|
| page | integer | No | Page number | Min: 1 | 1 |
| size | integer | No | Rows per page | Min: 1 | 20 |

**Success Response**: identical in shape to endpoint 1.

**Standard Error Types**:

- `400 BAD_REQUEST`: `page` is 0 or negative
- `401 UNAUTHORIZED`: Token missing, invalid or expired

---

# 6. List a Shop's Notifications

**Purpose**: For a shop owner — only the rows about one of their shops (new orders, low stock, reviews...).

**Endpoint**: `GET` `{base_url}/notifications/shop/{shopId}`

**Access Level**: Protected (returns only the signed-in user's own rows for that shop — another person's shop gives an empty list)

**Authentication**: Bearer Token

**Path Parameters**:

| Parameter | Type | Required | Description | Validation |
|---|---|---|---|---|
| shopId | UUID | Yes | The shop | Valid UUID |

**Query Parameters**:

| Parameter | Type | Required | Description | Validation | Default |
|---|---|---|---|---|---|
| page | integer | No | Page number | Min: 1 | 1 |
| size | integer | No | Rows per page | Min: 1 | 20 |

**Success Response**: identical in shape to endpoint 1; every row has this `shopId`.

**Standard Error Types**:

- `400 BAD_REQUEST`: `shopId` is not a UUID, or `page` is 0 or negative
- `401 UNAUTHORIZED`: Token missing, invalid or expired

---

# 7. Get One Notification

**Purpose**: Fetch a single row — e.g. when the app opens from a link that carries a notification id.

**Endpoint**: `GET` `{base_url}/notifications/{notificationId}`

**Access Level**: □□ Protected (own rows only)

**Authentication**: Bearer Token

**Path Parameters**:

| Parameter | Type | Required | Description | Validation |
|---|---|---|---|---|
| notificationId | UUID | Yes | The row | Valid UUID |

**Success Response JSON Sample**:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Notification retrieved successfully",
  "action_time": "2026-09-18T19:23:04.211589898",
  "data": { /* one notification object */ }
}
```

**Error Response JSON Sample** (staging — a row that does not exist or is not yours):

```
{
  "success": false,
  "httpStatus": "NOT_FOUND",
  "message": "Notification not found or access denied",
  "action_time": "2026-09-18T19:23:06.905963899",
  "data": "Notification not found or access denied"
}
```

**Standard Error Types**:

- `401 UNAUTHORIZED`: Token missing, invalid or expired
- `404 NOT_FOUND`: No such row, or it belongs to someone else (the two are deliberately indistinguishable)

---

# 8. Mark One as Read

**Purpose**: Call when the user taps a row.

**Endpoint**: **PUT** `{base_url}/notifications/{notificationId}/read`

**Access Level**: □□ Protected (own rows only)

**Authentication**: Bearer Token

**Path Parameters**:

| Parameter | Type | Required | Description | Validation |
|---|---|---|---|---|
| notificationId | UUID | Yes | The row | Valid UUID |

**Success Response JSON Sample**:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Notification marked as read",
  "action_time": "2026-09-18T19:25:00.000000000",
  "data": null
}
```

Marking a row that is already read succeeds and changes nothing — safe to retry. The badge is not sent back; subtract 1 locally.

**Standard Error Types**:

- `401 UNAUTHORIZED`: Token missing, invalid or expired
- `404 NOT_FOUND`: No such row, or not yours

---

# 9. Mark Several as Read

**Purpose**: Mark a selection read in one call.

**Endpoint**: **PUT** `{base_url}/notifications/read`

**Access Level**: □□ Protected (own rows only)

**Authentication**: Bearer Token

**Request JSON Sample**:

```
{
  "notificationIds": [
    "6e902bb6-9b38-49c6-825e-22b951f39584",
    "0c1d2e3f-4a5b-6c7d-8e9f-0a1b2c3d4e5f"
  ]
}
```

**Request Body Parameters**:

| Parameter | Type | Required | Description | Validation |
|---|---|---|---|---|
| notificationIds | array of UUID | Yes | Rows to mark read | Not empty |

**Success Response JSON Sample**:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "2 notification(s) marked as read",
  "action_time": "2026-09-18T19:25:00.000000000",
  "data": null
}
```

**All or nothing**: if even one id is missing or belongs to someone else, **no row is changed** and the call returns `400`. Drop ids of rows you have deleted before sending.

**Error Response JSON Sample**:

```
{
  "success": false,
  "httpStatus": "BAD_REQUEST",
  "message": "Some notifications not found or access denied",
  "action_time": "2026-09-18T19:25:00.000000000",
  "data": "Some notifications not found or access denied"
}
```

**Standard Error Types**:

- `400 BAD_REQUEST`: A listed id is missing or not yours; nothing was changed
- `401 UNAUTHORIZED`: Token missing, invalid or expired
- `422 UNPROCESSABLE_ENTITY`: `notificationIds` is empty

---

# 10. Mark All as Read

**Purpose**: The "Mark all as read" button.

**Endpoint**: `PUT` `{base_url}/notifications/read-all`

**Access Level**: □□ Protected

**Authentication**: Bearer Token

**Success Response JSON Sample**:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "All notifications marked as read",
  "action_time": "2026-09-18T19:25:00.000000000",
  "data": null
}
```

Set the badge to 0 locally afterwards.

**Standard Error Types**:

- `401 UNAUTHORIZED`: Token missing, invalid or expired

---

# 11. Delete One

**Purpose**: Swipe to delete.

**Endpoint**: `DELETE` `{base_url}/notifications/{notificationId}`

**Access Level**: □□ Protected (own rows only)

**Authentication**: Bearer Token

**Path Parameters**:

| Parameter | Type | Required | Description | Validation |
|---|---|---|---|---|
| notificationId | UUID | Yes | The row | Valid UUID |

**Success Response JSON Sample**:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "Notification deleted successfully",
  "action_time": "2026-09-18T19:25:00.000000000",
  "data": null
}
```


Deleting an **unread** row lowers the unread count — adjust the badge. Deleted rows cannot be recovered.

**Standard Error Types**:

- `401 UNAUTHORIZED`: Token missing, invalid or expired
- `404 NOT_FOUND`: No such row, or not yours

---

# 12. Delete Several

**Purpose**: Delete a selection in one call.

**Endpoint**: `DELETE` `{base_url}/notifications/batch`

**Access Level**: □□ Protected (own rows only)

**Authentication**: Bearer Token

> This DELETE carries a JSON body. Most HTTP clients support that, but some default helpers drop it — check yours sends `Content-Type: application/json` and the body.

**Request JSON Sample**:


```
{
  "notificationIds": [
    "6e902bb6-9b38-49c6-825e-22b951f39584"
  ]
}
```


**Request Body Parameters**:

| Parameter | Type | Required | Description | Validation |
|---|---|---|---|---|
| notificationIds | array of UUID | Yes | Rows to delete | Not empty |

**Success Response JSON Sample**:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "1 notification(s) deleted successfully",
  "action_time": "2026-09-18T19:25:00.000000000",
  "data": null
}
```

**All or nothing**, like endpoint 9: one bad id and nothing is deleted.

**Standard Error Types**:

- `400 BAD_REQUEST`: A listed id is missing or not yours; nothing was deleted
- `401 UNAUTHORIZED`: Token missing, invalid or expired
- `422 UNPROCESSABLE_ENTITY`: `notificationIds` is empty

---

# 13. Delete All Read

**Purpose**: The "Clear read" button — removes every row already read. Unread rows stay.

**Endpoint**: DELETE `{base_url}/notifications/read`

**Access Level**: □□ Protected

**Authentication**: Bearer Token

**Success Response JSON Sample**:

```
{
  "success": true,
  "httpStatus": "OK",
  "message": "3 read notification(s) deleted successfully",
  "action_time": "2026-09-18T19:25:00.000000000",
  "data": 3
}
```

**Success Response Fields**:

| Field | Description |
|---|---|
| data | How many rows were deleted (a number, not an object) |

The badge does not change — only read rows were removed.

**Standard Error Types**:

- `401 UNAUTHORIZED`: Token missing, invalid or expired

---

# Integration Checklist

- ☐ Badge loads from `unread-count` on app start **and** on every return to the foreground
- ☐ `notification.added` on the existing `/api/v1/events` stream sets the badge from `unreadCount` — no second stream
- ☐ Pages start at 1; paging stops when `hasNext` is `false`; duplicate `id`s are dropped
- ☐ `title` is shown as the sentence; `message` line hidden when `""`
- ☐ `createdAt` parsed as UTC
- ☐ A tap marks the row read, then opens `targetType` + `targetId`
- ☐ **An unknown `targetType` stays on the bell list — no crash, no error**
- ☐ A `404` from the target screen shows that screen's "not available" state
- ☐ Badge adjusted locally after mark-read and delete (no event comes back)
- ☐ Batch mark-read and batch delete send only ids still in the list
- ☐ Unknown keys in `data` and unknown `type` values are ignored

---

# Quick Reference Guide

## Common HTTP Status Codes

- `200 OK`: Successful request
- `400 Bad Request`: Invalid request data (e.g. `page=0`, an id that is not yours in a batch)
- `401 Unauthorized`: Authentication required/failed
- `404 Not Found`: Row does not exist or is not yours
- `422 Unprocessable Entity`: Validation errors (empty `notificationIds`)
- `500 Internal Server Error`: Server error

# Authentication

- **Bearer Token**: Include `Authorization: Bearer <accessToken>` in headers

# Data Format Standards

- **Dates**: ISO 8601, **UTC without an offset** (`2026-09-18T19:18:40.736493`)
- **IDs**: UUID strings
- **Pagination**: `page` (from 1) and `size`; response carries `currentPage`, `totalPages`, `hasNext`

---

Revision #6
Created 18 October 2025 20:44:11 by Admin Qbit
Updated 18 September 2026 19:43:12 by Admin Qbit