OpSense Notifications

Usage guide for the Mopuan integration

opsenseapi · Version 1.0 · 6 November 2026

1. What this service is for

OpSense creates a notification for an employee when something needs their attention, for example:

- a new work order,
- a note added to a work order assigned to them,
- a work order inspection that needs review,
- a panic alert from someone in their team.

`SRVGetNotifications` returns these notifications so you can show them in Mopuan. It is read-only. Calling it does not change anything.

A push notification is only an alert. The list from this service is the full record. When a push arrives while the notification screen is open, load the list again.

2. Authentication

Send the company's `api_key` as a normal request parameter, as a POST field, the same as for the other OpSense services. Do not send it as an HTTP header or in the URL.

The key decides which company the call is for. You only receive that company's notifications.

Situation	Response
`api_key` missing	`error_code` 10015
`api_key` invalid or inactive	`error_code` 10016

Every response is wrapped the same way:

```
{"result": { ... success or error body ... }}
```

3. Request

```
POST https://<api-host>/v1/SRVGetNotifications
Content-Type: application/x-www-form-urlencoded
```

Send the parameters as **form fields**. A JSON request body is not read: if the fields are sent as JSON, the service ignores them and returns every notification of the company.

Field	Required	Notes

`api_key`	**Required**	See Authentication.
`mopuan_user_id`	Optional	Return only this employee's notifications. Your Mopuan employee id.
`user_id`	Optional	Return only this employee's notifications, by OpSense user id. Ignored if `mopuan_user_id` is sent.
`limit`	Optional	Page size: return only the newest `limit` notifications, newest first. A whole number above 0.
`before_id`	Optional	Return only notifications older than this id. Send the `next_before_id` of the previous page to get the next page.

What you get back

You send	You get
`api_key` only	Every notification of the company, newest first.
`mopuan_user_id`	Every notification of that employee, newest first.
`mopuan_user_id`, `limit=30`	First page: that employee's 30 newest notifications, newest first, plus `has_more` and `next_before_id`.
`mopuan_user_id`, `limit=30`, `before_id=<next_before_id>`	Next page: the 30 notifications after the previous page, plus `has_more` and `next_before_id`.

If `limit` is missing, 0 or not a number, you get every notification, as in the first two rows, and the response has no `has_more` or `next_before_id`.

Paging

1. Ask for the first page with `limit` and no `before_id`.
2. When the employee scrolls to the end and `has_more` is `true`, ask for the next page with the same `limit` and `before_id` set to `next_before_id`. Add the new notifications to the end of the list.
3. When `has_more` is `false`, there are no older notifications. Stop asking.
4. To refresh, ask for the first page again (without `before_id`) and replace the list.

Notifications created while the employee is scrolling do not move the pages. No notification is repeated or skipped. New ones appear on the next refresh.

Examples

First page, 30 newest notifications:

```
curl -X POST "https://<api-host>/v1/SRVGetNotifications" \
 -d "api_key=<api_key>" \
 -d "mopuan_user_id=345" \
 -d "limit=30"
```

Next page:

```
curl -X POST "https://<api-host>/v1/SRVGetNotifications" \
 -d "api_key=<api_key>" \
 -d "mopuan_user_id=345" \
 -d "limit=30" \
 -d "before_id=25481"
```

Every notification of the employee, without paging:

```
curl -X POST "https://<api-host>/v1/SRVGetNotifications" \
 -d "api_key=<api_key>" \
 -d "mopuan_user_id=345"
```

4. Response

A response to a request with `limit`:

```
{
 "result": {
   "result": "success",
   "data": [
     {
       "id": 25510,
       "user_id": 87,
       "mopuan_user_id": 345,
       "link": "https://<portal-host>/calendar",
       "title": "New Note on Work Order",
       "description": "New note added by Jane Smith",
       "added_date": "2026-11-06 14:02:05"
     }
   ],
   "has_more": true,
   "next_before_id": 25481
 }
}
```

Field	Notes
`data`	The notifications. An empty list when there are none.
`has_more`	Only when `limit` is sent. `true` if older notifications exist.
`next_before_id`	Only when `limit` is sent. Send it as `before_id` to get the next page. `null` when `has_more` is `false`.

Each notification in `data`:

Field	Notes
`id`	Notification id. A newer notification always has a higher id.
`user_id`	OpSense user id of the employee the notification is for.

`mopuan_user_id`	Mopuan employee id of the same employee. Empty if the employee is not linked to Mopuan.
`link`	Page in the OpSense portal that the notification is about.
`title`	Short title. Plain text.
`description`	One-line description. Plain text.
`added_date`	When the notification was created, `YYYY-MM-DD HH:MM:SS`, in the company's time zone.

5. Recommendations

- **Always send `mopuan_user_id`.** Without it you receive the whole company's notifications, which is slow and shows employees notifications that are not theirs.
- **Page list screens.** Send `limit` (30 to 50 is enough for one screen) and load older pages with `before_id`. Each response stays small and fast however many notifications the employee has.
- **Keep the order of the response.** Notifications always come newest first, sorted by `id`. If you sort them yourself, sort by `id`, not by `added_date`: several notifications can share the same second.
- **Show `title` and `description` as text.** Do not render them as HTML.