SRVGetNotifications: What Changed

Change notice for the Mopuan integration

opsenseapi · Version 1.1 · 6 November 2026

1. Summary

`SRVGetNotifications` is an existing service. It now supports paging:

- two new optional request fields, `limit` and `before_id`,
- two new response fields, `has_more` and `next_before_id`, returned only when `limit` is sent,
- notifications always come newest first.

If you do not send `limit`, you get the same notifications in the same response format as before. The only difference is that they are now sorted newest first. You do not need to change anything unless you want to use paging.

2. New request fields

Field	Required	Notes
`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.

If `limit` is missing, 0 or not a number, it is ignored and you get every notification, as before.

3. New response fields

Returned only when `limit` is sent, next to `result` and `data`:

Field	Notes
`has_more`	`true` if older notifications exist.
`next_before_id`	Send it as `before_id` to get the next page. `null` when `has_more` is `false`.

4. Before and after

Request	Before	Now

`api_key`, `mopuan_user_id`	Every notification of the employee, in no fixed order	The same notifications, newest first
`api_key`, `mopuan_user_id`, `limit=30`	`limit` ignored: every notification of the employee	The employee's 30 newest notifications, newest first, plus `has_more` and `next_before_id`
`api_key`, `mopuan_user_id`, `limit=30`, `before_id=<id>`	Both fields ignored: every notification of the employee	The 30 notifications older than `<id>`, plus `has_more` and `next_before_id`

5. What stays the same

- The URL and the authentication.
- The `mopuan_user_id` and `user_id` fields.
- The `result` and `data` fields and every field of a notification.
- The error codes.

6. How to start using it

1. Send the fields as **form fields** (`application/x-www-form-urlencoded`). Fields in a JSON request body are not read, so `limit` and `before_id` sent as JSON have no effect.
2. Send `mopuan_user_id` together with `limit`. Without `mopuan_user_id`, the pages hold the whole company's notifications, not the employee's.
3. Ask for the first page with `limit` and no `before_id`.
4. When the employee scrolls to the end and `has_more` is `true`, ask for the next page with `before_id` set to `next_before_id`. Add the new notifications to the end of the list.
5. When `has_more` is `false`, stop asking. To refresh, ask for the first page again.

First page:

```
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"
```

The full description of the service is in the OpSense Notifications usage guide.