Roles, Permissions and Users

Management API guide for the Mopuan integration

opsenseapi · Version 1.0 · 9 November 2026

1. What these services are for

These services let the Mopuan app and its server manage the company's **roles**, **permissions** and **users** in OpSense. You can list, create, change and delete them. You can also set which employee reports to which supervisor.

To only read roles, permissions and hierarchy for Mopuan, use the read-only guide **MopuanRoles_API_Guide** (services `SRVGetUserRoles`, `SRVGetRoles` and `SRVGetCompanyRoleData`). This guide does not repeat them.

Service	Use it to
`SRVGetPermissionMatrix`	Read the permission matrix of one role: every module and the four flags of the role.
`SRVGetRole`	Read one role.
`SRVGetUsers`	List users (9 per page) with filters, plus the role list and supervisor choices for a user form.
`SRVGetUser`	Read one user with the supervisor ids.
`SRVSaveRole`	Create a role, or change one.
`SRVDeleteRole`	Delete a role.
`SRVSaveRolePermissions`	Replace the permissions of a role.
`SRVSaveUser`	Create a user, or change one: profile, role, supervisors, password, profile image, address.
`SRVDeleteUser`	Delete a user.
`SRVUpdateUserSupervisor`	Add one supervisor to a user, or clear all supervisors.
`SRVRemoveUserSupervisor`	Remove one supervisor from a user.

2. Hierarchy levels

Every role has a hierarchy level from 0 to 3. A **lower** number means **more** authority.

Level	Name	Can see

0	Super Admin	Every user of the company
1	Admin	Every user of the company
2	Supervisor	Themselves, their direct reports, and their direct reports' direct reports
3	User	Only themselves

A user marked **only assigned** (`only_assigned` = 1) sees only themselves, whatever the level.

The same visibility decides which users the acting employee can read, edit and delete (section 5). "Same level" means a peer. "Lower level" means a higher number, so less authority.

3. Authentication and the acting employee

Every call needs two fields:

Field	Meaning
`api_key`	The company API key. It decides which company the call is for.
`actor_mopuan_user_id`	The Mopuan id of the employee who does the action. OpSense finds the employee with this id and applies that employee's role, level and permissions.

Both fields come from your server. Your server fills `actor_mopuan_user_id` with the employee it has authenticated. Never take this value from the phone or from any request the user can change. Never put the `api_key` in the mobile app or in a web page.

The employee must be active, not deleted, and linked to Mopuan (the user has a `mopuan_user_id`).

Situation	Response
`api_key` missing	`error_code` 10015
`api_key` invalid or inactive	`error_code` 10016
`actor_mopuan_user_id` missing, not a number, or no active employee has it	`error_code` 10030
The employee's role lacks the permission for the action	`error_code` 10031

4. Which permission each action needs

Permissions are set per role and per module. Each module has four flags: view, create, edit and delete. The three modules used here are **Role Access** (`role-access`), **Permissions** (`permissions`) and **Users list** (`users-list`). The check uses the role of the acting employee. Without the permission the call fails with 10031.

Service	Module	Flag needed
`SRVGetPermissionMatrix`	Permissions	view
`SRVGetRole`	Role Access	view
`SRVGetUsers`	Users list	view
`SRVGetUser`	Users list	view (and the user must be visible to the actor)
`SRVSaveRole` (create)	Role Access	create
`SRVSaveRole` (change)	Role Access	edit
`SRVDeleteRole`	Role Access	delete
`SRVSaveRolePermissions`	Permissions	edit
`SRVSaveUser` (create)	Users list	create
`SRVSaveUser` (change)	Users list	edit. Without it, an employee can change only their own name, surname, e-mail, phone and profile image.
`SRVDeleteUser`	Users list	delete
`SRVUpdateUserSupervisor`	Users list	edit, and the actor's level must be 0 or 1
`SRVRemoveUserSupervisor`	Users list	edit, and the actor's level must be 0 or 1

A change is a call with `edit` set to a user or role id. A create is a call without `edit`.

5. Rules the API enforces

The API checks these rules on every call, whatever your app shows. If a rule fails, nothing is saved.

Roles and permissions

- **Shared roles are read-only.** A role with `company_id` 0 is shared by all companies. You can read it and assign it to a user. You cannot change it, change its permissions or delete it (10040).
- **Role 1 is protected.** It cannot be changed, re-permissioned or deleted (10031).
- **An employee can grant only permissions they hold.** A flag can be set to on only if the acting employee's own role has that flag on that module (10041).
- **Grants go only to roles at or below the actor's level.** The target role's level number must be equal to or higher than the actor's (10041).
- **A flag the actor does not hold stays as it is.** If the target role already has such a flag, a permission save keeps it. It does not fail and it does not remove it.

- **A role cannot be created or moved above the actor.** The level of a role you create or change must be equal to or higher than your own level (10031). A role with a lower level number than yours cannot be changed (10031).
- **A role in use cannot be deleted.** If any user has the role, the delete fails (10043). To delete a role, first move its users to another role. A role can be deleted only if its level number is higher than the actor's (10031).

Assigning a role to a user

- The role must exist, belong to the company or be shared, and have a level equal to or higher (number) than the actor's (10031).
- **A role with more permissions than the actor holds is refused** (10041). This applies when you create a user with that role, when you change a user's role (also your own), and when you change the hierarchy level of a role that has users.

Supervisor links

- The user and the supervisor must be in the company, active and not deleted.
- The supervisor's role level must be 0, 1 or 2.
- A user cannot be their own supervisor.
- No cycles. If A reports to B, then B cannot report to A, directly or through other users.
- A user who is saved as inactive cannot get new supervisors.
- These checks apply to links you add. Removing a link only needs the link to exist.
- Any violation gives 10042.

Users

- **Visibility limits editing and deleting.** The acting employee can edit or delete only users they can see (section 2). Otherwise the call fails with 10031.
- **Editing.** An employee can edit users at their own level or lower. A user with more authority cannot be edited (10031). Without the edit permission, an employee can edit only themselves.
- **Password and e-mail of another user** can be changed only if that user's level number is strictly higher than the actor's, meaning strictly less authority (10031). A peer's password and e-mail cannot be changed. You can always change your own.
- **Deleting.** A user can be deleted only if their level number is strictly higher than the actor's. Peers and users with more authority cannot be deleted (10031). Nobody can delete themselves this way.
- **User 1 is protected.** Nobody can edit or delete user 1 except user 1 editing themselves (10031).
- **User level** (`form_userlevel`) can be set only with the edit permission. Its highest value cannot be higher than the actor's own (10031).
- The e-mail address must be unique (10045).
- Deleted users are treated as not found (10020).

6. Request format

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

Send the parameters as **form fields**. A JSON request body is not read. Send `api_key` and `actor_mopuan_user_id` as form fields too, not as headers.

- **Flags** are `0` or `1`: `form_status`, `form_active`, `form_only_assigned`.
- **Lists** use repeated fields: `form_supervisors[]=3&form_supervisors[]=7`.
- **Fields you leave out are not changed** on an update.
- **Numbers** in responses can arrive as numbers or as text. Convert them before you compare.

Profile image (multipart)

To send a profile image to `SRVSaveUser`, use `multipart/form-data` and put the file in the field `pp`. All other fields are sent as normal form parts.

Rule	Value
Allowed types	JPEG (`.jpg`, `.jpeg`), PNG (`.png`), WebP (`.webp`)
Maximum size	5 MB (5 x 1024 x 1024 bytes)
Checks	The file extension and the real file content must both be the same image type. A renamed file is refused.
File name	Any name. OpSense creates its own name.
Size of the image	The image is stored as sent. It is not resized.
Error	10044 for an invalid image.

Permission fields: `yetki[...]`

`SRVSaveRolePermissions` reads the permissions from nested fields:

```
yetki[<role_id>][<section_id>][gorme]
yetki[<role_id>][<section_id>][olusturma]
yetki[<role_id>][<section_id>][duzenleme]
yetki[<role_id>][<section_id>][silme]
```

Name	Meaning
`gorme`	view
`olusturma`	create
`duzenleme`	edit
`silme`	delete

- `<role_id>` must be the same id as the `role_id` field.
- `<section_id>` is the `id` of a module from `SRVGetPermissionMatrix`.

- **If a flag field is present, the flag is on. Its value is ignored.** Even `=0` turns it on. To turn a flag off, do not send it.
- **Send the full set you want**, not only the changes. Flags you do not send are turned off, except flags the actor does not hold that the role already has (section 5).

```
-d "yetki[7][12][gorme]=1" \
-d "yetki[7][12][olusturma]=1" \
-d "yetki[7][12][duzenleme]=1" \
-d "yetki[7][15][gorme]=1"
```

This gives role 7 view, create and edit on module 12, and view on module 15. Every other flag of role 7 is off.

7. Response format

Responses are JSON. They are **not** wrapped in an extra `result` object.

Success:

```
{"result": "success", "data": { ... }}
```

Error:

```
{"error_code": "10031", "error_text": "You do not have permission to edit this user."}
```

Check `error_code`. The `error_text` is in English, and it can change. Use it for logs, not for logic. Passwords and password hashes are never returned by any service.

8. Services

All services use POST and the format in section 6. Each one needs `api_key` and `actor_mopuan_user_id`.

8.1 SRVGetPermissionMatrix

Returns the role and the permission flags of the role for every module. Use it to show a permission screen and to get the module ids for `SRVSaveRolePermissions`.

Needs: Permissions, view.

Field	Required	Notes
`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`role_id`	**Required**	The role. A company role or a shared role.

```
curl -X POST "https://<api-host>/v1/SRVGetPermissionMatrix" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "role_id=7"
```

```
{"result": "success", "data": {
 "role": {"id": 7, "company_id": 3, "role": "Floor Manager", "status": 1,
          "hierarchy_level": 2, "added_date": "2026-10-01 09:00:00"},
 "sections": [
   {"id": 12, "url": "work-orders", "sayfaAdi": "Work Orders", "subId": 0,
    "gorme": 1, "olusturma": 1, "duzenleme": 1, "silme": 0}
 ]}}
```

`sections` lists the visible modules in menu order. `url` is the module key. `sayfaAdi` is its name. `gorme`, `olusturma`, `duzenleme` and `silme` are view, create, edit and delete (1 or 0). A module the role has no access to is listed with all four at 0.

error_code	When
10020	`role_id` is missing or not a role of this company or a shared role.
10030, 10031	The acting employee was not found, or has no permission (sections 3 and 4).

8.2 SRVGetRole

Returns one role. Needs: Role Access, view.

Field	Required	Notes
`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`role_id`	**Required**	The role. A company role or a shared role.

```
curl -X POST "https://<api-host>/v1/SRVGetRole" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "role_id=7"
```

```
{"result": "success", "data": {"id": 7, "company_id": 3, "role": "Floor Manager",
 "status": 1, "hierarchy_level": 2, "added_date": "2026-10-01 09:00:00"}}
```

`company_id` 0 means a shared role (read-only).

error_code	When
10020	The role was not found.

8.3 SRVGetUsers

Lists users. The list shows only users the acting employee can see (section 2), 9 per page, newest first. Needs: Users list, view.

Field	Required	Notes

`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`name`	Optional	Search text. Matches name, surname, e-mail or full name. Not case sensitive.
`role_id`	Optional	Only users with this role.
`page`	Optional	Page number, from 1. Default 1.

```
curl -X POST "https://<api-host>/v1/SRVGetUsers" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "name=jane" \
 -d "page=1"
```

```
{"result": "success", "data": {
 "users": [
   {"id": 42, "name": "Jane", "surname": "Doe", "email": "jane@example.com",
    "phone": "5551234567", "role": 7, "active": 1, "only_assigned": 0,
    "mopuan_user_id": 345, "supervisors": [12, 15]}
 ],
 "total": 1, "page": 1, "per_page": 9,
 "roles": [{"id": 7, "role": "Floor Manager", "hierarchy_level": 2}],
 "supervisor_choices": [{"id": 12, "name": "Sam", "surname": "Lee"}]
}}
```

- `users` has more fields than shown (the user record). Treat unknown fields as optional. Passwords are never included.
- `supervisors` is the list of supervisor user ids.
- The fields `address`, `address_lat` and `address_lng` are returned only if the acting employee has the Users list edit permission.
- `profile_picture` is a full address, as in `SRVGetUser`.
- `roles` is the role list for a role picker. `supervisor_choices` lists active users with level 0, 1 or 2, for a supervisor picker.

error_code	When
10030, 10031	The acting employee was not found, or has no permission (sections 3 and 4).

8.4 SRVGetUser

Returns one user with the supervisor ids. The user must be visible to the acting employee. Needs: Users list, view.

Field	Required	Notes
`api_key`	**Required**	The company API key.

`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`user_id`	**Required**	The user.

```
curl -X POST "https://<api-host>/v1/SRVGetUser" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "user_id=42"
```

```
{"result": "success", "data": {
 "id": 42, "name": "Jane", "surname": "Doe", "email": "jane@example.com",
 "phone": "5551234567", "role": 7, "active": 1, "only_assigned": 0,
 "mopuan_user_id": 345,
 "profile_picture": "https://<panel-host>/uploads/users/pp_65f1.png",
 "supervisors": [12, 15]
}}
```

`profile_picture` is a full address. When the user has no image, it is the address of a default image. `address`, `address_lat` and `address_lng` are returned only with the Users list edit permission.

error_code	When
10020	The user was not found in this company.
10031	No permission, or the user is not visible to the actor.

8.5 SRVSaveRole

Creates a role, or changes one. Needs: Role Access, create (new role) or edit (change).

Field	Required	Notes
`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`edit`	Optional	The id of the role to change. Leave it out to create a role. A value that is not a positive whole number gives 10045.
`form_role`	**Required (create)**	The role name. Cannot be empty.
`form_status`	Optional	`1` active, `0` inactive. Default for a new role: `1`.
`form_hierarchy_level`	Optional	0, 1, 2 or 3. Default for a new role: 3. Cannot be lower than the actor's own level.

```
curl -X POST "https://<api-host>/v1/SRVSaveRole" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "form_role=Floor Manager" \
```

```
 -d "form_status=1" \
 -d "form_hierarchy_level=2"
```

```
{"result": "success", "data": {"id": 7}}
```

A new role has no permissions. Set them with `SRVSaveRolePermissions`.

error_code	When
10020	The role was not found.
10031	Role 1; the role has more authority than the actor; or the level is lower than the actor's level.
10040	The role is shared (read-only).
10041	You change the level of a role in use, and the role holds permissions the actor does not hold.
10045	`edit` is not a valid id, the name is missing or empty, `form_hierarchy_level` is not 0 to 3, or `form_status` is not 0 or 1.

8.6 SRVDeleteRole

Deletes a role and its permissions. Needs: Role Access, delete.

Field	Required	Notes
`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`role_id`	**Required**	The role to delete.

```
curl -X POST "https://<api-host>/v1/SRVDeleteRole" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "role_id=7"
```

```
{"result": "success", "data": {"id": 7}}
```

error_code	When
10020	The role was not found.
10031	Role 1, or the role does not have a higher level number than the actor.
10040	The role is shared.
10043	A user has this role.

8.7 SRVSaveRolePermissions

Replaces the permissions of a role with the set you send. See `yetki[...]` in section 6. Needs: Permissions, edit.

Field	Required	Notes
`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`role_id`	**Required**	The role.
`yetki[<role_id>][<section_id>][<flag>]`	Optional	The flags to turn on. Flags you do not send are turned off (with the exception in section 5).

```
curl -X POST "https://<api-host>/v1/SRVSaveRolePermissions" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "role_id=7" \
 -d "yetki[7][12][gorme]=1" \
 -d "yetki[7][12][duzenleme]=1"
```

```
{"result": "success", "data": {"role_id": 7}}
```

If a call fails, the permissions may be partly saved but the role never ends with fewer permissions than before. Read the matrix again to see the result.

error_code	When
10020	The role was not found.
10031	Role 1.
10040	The role is shared.
10041	The role has more authority than the actor, you try to turn on a flag the actor does not hold, or the `yetki` field is malformed.
10099	The permissions of this role are being saved by another call. Try again.

8.8 SRVSaveUser

Creates a user, or changes one. Needs: Users list, create (new user) or edit (change). Without the edit permission, an employee can change only their own name, surname, e-mail, phone and profile image. Other fields in that call are ignored.

Use `multipart/form-data` if you send `pp`. Otherwise use the normal form format.

Field	Required	Notes

`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`edit`	Optional	The id of the user to change. Leave it out to create a user. A value that is not a positive whole number gives 10045.
`form_name`	**Required (create)**	Name. Cannot be empty. Spaces at the ends are removed.
`form_surname`	**Required (create)**	Surname. Same rules as the name.
`form_email`	**Required (create)**	A valid e-mail address. It must not be used by another user.
`password`	**Required (create)**	The password, as text. On a change, send it only to set a new one. See section 5 for who can change another user's password.
`form_role`	**Required (create)**	The role id. See "Assigning a role" in section 5.
`form_phone`	Optional	Phone number.
`phone_country_code`	Optional	Country code of the phone, for example `90`. Used for Mopuan. It is not saved in OpSense. Without both a phone and a country code, no phone is sent to Mopuan.
`form_only_assigned`	Optional	`1` to limit the user to their own data. Default for a new user: `0`.
`form_active`	Optional	`1` active, `0` inactive. Default for a new user: `1`.
`form_weekly_route_user`	Optional	Saved as sent.
`form_contract`	Optional	Saved as sent.
`form_duty`	Optional	Saved as sent.
`form_mopuan_beacon`	Optional	The beacon (card) number of the employee. It is sent to Mopuan. If you clear it, the old card is removed in Mopuan.
`form_mopuan_beacon_type`	Optional	Saved as sent.
`form_userlevel`	Optional	The user level. Needs the edit permission, and cannot be higher than the actor's own.
`form_supervisors[]`	Optional	The full list of supervisor user ids. Links not in the list are removed. An empty list is not sent as a field. To remove all supervisors, use `SRVUpdateUserSupervisor` with `supervisor_id=0`. See "Supervisor links" in section 5.

`pp`	Optional	The profile image file (multipart). See section 6.
`user_address`	Optional	The address. Spaces at the ends are removed. Only the first 255 characters are saved. Needs the edit permission.
`user_address_lat`, `user_address_lng`	Optional	The coordinates of the address. Latitude from -90 to 90, longitude from -180 to 180. If the address is empty, or a coordinate is missing or out of range, or both are 0, both coordinates are cleared. Only read together with `user_address`.

```
curl -X POST "https://<api-host>/v1/SRVSaveUser" \
 -H "Content-Type: multipart/form-data" \
 -F "api_key=<api_key>" \
 -F "actor_mopuan_user_id=101" \
 -F "form_name=Jane" \
 -F "form_surname=Doe" \
 -F "form_email=jane@example.com" \
 -F "password=<password>" \
 -F "form_role=7" \
 -F "phone_country_code=90" \
 -F "form_phone=905551234567" \
 -F "form_supervisors[]=12" \
 -F "pp=@/path/photo.jpg"
```

```
{"result": "success", "data": {"id": 42, "mopuan": {"status": "success", "user_id": 345}}}
```

`id` is the OpSense user id. `mopuan` shows the result of the Mopuan update (section 9).

error_code	When
10020	The user was not found; the e-mail is already used; or `user_address` is not text.
10031	User 1 (other than by user 1); the user is not visible; the actor has no edit permission and edits another user; the user has more authority than the actor; invalid role; invalid `form_userlevel`; password or e-mail of a peer or a higher user.
10041	The role holds permissions the actor does not hold.
10042	A supervisor link is not allowed, or `form_supervisors` is not a list of valid ids.
10044	The image is not valid, or image storage is not available.
10045	`edit` is not a valid id; `form_active` or `form_only_assigned` is not 0 or 1; name, surname or e-mail is empty or invalid; `password` is not text; or a required field of a new user is missing.

8.9 SRVDeleteUser

Deletes a user and the user's supervisor links, both as employee and as supervisor. Needs: Users list, delete.

Field	Required	Notes
`api_key`	**Required**	The company API key.

`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`user_id`	**Required**	The user to delete.

```
curl -X POST "https://<api-host>/v1/SRVDeleteUser" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "user_id=42"
```

```
{"result": "success", "data": {"id": 42, "mopuan": {"status": "success"}}}
```

error_code	When
10020	The user was not found.
10031	User 1; the user is not visible; or the user is a peer or has more authority.

8.10 SRVUpdateUserSupervisor

Adds one supervisor to a user, or clears all supervisors of the user. Needs: Users list, edit, and the actor's level 0 or 1.

Field	Required	Notes
`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`user_id`	**Required**	The user.
`supervisor_id`	**Required**	The supervisor to add. `0` **removes all supervisors of the user.** If you leave the field out, all supervisors are removed too.

```
curl -X POST "https://<api-host>/v1/SRVUpdateUserSupervisor" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "user_id=42" \
 -d "supervisor_id=12"
```

```
{"result": "success", "data": {"user_id": 42}}
```

error_code	When
10031	The actor's level is not 0 or 1, or no edit permission.
10042	The user was not found, or the link is not allowed (section 5).

8.11 SRVRemoveUserSupervisor

Removes one supervisor from a user. Needs: Users list, edit, and the actor's level 0 or 1.

Field	Required	Notes
`api_key`	**Required**	The company API key.
`actor_mopuan_user_id`	**Required**	The Mopuan id of the employee who does the action. See section 3.
`user_id`	**Required**	The user.
`supervisor_id`	**Required**	The supervisor to remove.

```
curl -X POST "https://<api-host>/v1/SRVRemoveUserSupervisor" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=101" \
 -d "user_id=42" \
 -d "supervisor_id=12"
```

```
{"result": "success", "data": {"user_id": 42}}
```

error_code	When
10031	The actor's level is not 0 or 1, or no edit permission.
10042	This link does not exist.

9. What happens in Mopuan

When you save or delete a user, OpSense also updates the employee in Mopuan. Your server does not need to create or update the Mopuan employee itself.

- **User created:** the employee is created in Mopuan. OpSense stores the new Mopuan id as the user's `mopuan_user_id`. If you sent a password, it is set in Mopuan. If you sent a beacon number, the card is saved.
- **User changed:** the employee is updated in Mopuan with the new name, e-mail and phone. A new password and a new or cleared beacon are sent too. A user that has no Mopuan employee yet gets one.
- **User deleted:** the employee is deleted in Mopuan, if the user has a `mopuan_user_id`.

The result is in `data.mopuan.status`:

status	Meaning
`success`	Mopuan was updated. After a create or change, `user_id` is the Mopuan employee id.
`skipped`	The company has no active Mopuan connection. Nothing was sent.
`failed`	Mopuan could not be updated. `message` says so.

The OpSense change is kept even when Mopuan fails. The user is saved or deleted in OpSense and the response is still a success. Always read `data.mopuan.status`. After `failed`, save the user again with `SRVSaveUser` to repeat the Mopuan update. A new user whose Mopuan update failed has no `mopuan_user_id` yet, so it cannot be used as `actor_mopuan_user_id` until the update works.

10. Error codes

error_code	Meaning	Retry?
10015	`api_key` missing.	No. Fix the request.
10016	`api_key` invalid or inactive.	No. Check the key.
10020	A role or user was not found, or a required field is missing or not allowed (role name, e-mail already used, address).	No.
10030	The acting employee was not found: `actor_mopuan_user_id` is missing or invalid, or the employee is not active.	No.
10031	Not allowed: missing permission, hierarchy rule, visibility rule, or protection of role 1 and user 1.	No.
10040	The role is shared and read-only.	No.
10041	The grant is above what the actor holds, or the target role has more authority than the actor.	No.
10042	Invalid supervisor link (not allowed, cycle, wrong level, inactive or missing user, or bad list).	No.
10043	The role is in use. A user has this role.	No. Move the users first.
10044	The profile image is not valid, or image storage is not available.	Only with a corrected image.
10045	A field value is not valid: `edit` id, level not 0 to 3, a flag not 0 or 1, empty or invalid name, surname or e-mail, `password` not text, or a required field of a new user missing.	No. Fix the request.
10099	The request could not be completed on our side.	Yes

11. Recommendations

- **Call these services from your server only.** The phone talks to your server. Your server authenticates the employee and sends `actor_mopuan_user_id` itself.
- **Hide what the employee cannot do, but do not rely on it.** Use the permissions from `SRVGetUserRoles` to hide buttons. The API checks every call anyway and answers 10031 or 10041.

- **Build the forms from the API.** Use `roles` and `supervisor_choices` from `SRVGetUsers` for the user form, and the sections from `SRVGetPermissionMatrix` for the permission screen.
- **Send the full permission set** to `SRVSaveRolePermissions`. Remember that a flag field turns the flag on, whatever its value.
- **Read again after a save.** Call `SRVGetUser` or `SRVGetPermissionMatrix` to show what was saved.
- **Check `data.mopuan.status` after every user save or delete** (section 9).
- **Decide on `error_code`, not on `error_text`.** Show your own message to the employee. Retry only after 10099.
- **Check before you delete a role.** Use `SRVGetUsers` with `role_id` to see if users have it.
- **Keep your copy of roles and permissions fresh.** Use the read services in **MopuanRoles_API_Guide** after changes.