AI SPV (Assignment Desk) API

Integration guide for the Mopuan app and its server

opsenseapi · Version 1.0 · 9 November 2026

1. What these services are for

AI SPV (the Assignment Desk) plans the work for a team. It takes the open work orders that nobody has taken yet and suggests which employee should do each one. You pick a **goal** (for example quality or low cost) and AI SPV calculates the best plan for that goal.

A suggestion is only a **draft**. Nothing changes in OpSense until you **commit** an assignment. The user can review the plan, lock a person to a job, skip a job, or change who is in the team, and then calculate again.

Eight services are available:

Service	Use it to
`SRVGetAiSpvDesk`	Load the desk: team, open work orders, existing work, services and places for a date range.
`SRVCalculateAiSpv`	Calculate the suggested plan for a goal. Changes nothing.
`SRVAiSpvCommitAssignment`	Assign one work order to one employee.
`SRVAiSpvCreateWorkOrder`	Create a new unassigned work order.
`SRVAiSpvTogglePool`	Take an employee out of the planning team, or put them back.
`SRVAiSpvSetQualityOverride`	Set a manual quality score for an employee.
`SRVAiSpvClearQualityOverride`	Remove one manual quality score, or all of them.
`SRVAiSpvSaveStaffCost`	Save an employee's hourly cost.

2. Goals

Send the goal in `goal`. If you leave it out, `balanced` is used. AI SPV scores every employee for every job on ten criteria (0 to 100) and adds them up with weights. Each goal uses different weights.

`goal`	What it favours
`balanced`	A mix of everything. The default. Quality and skill count most.
`quality`	The most skilled and best-rated employee for the job, and a good match between the job's priority and the employee's level. Ignores cost.
`efficiency`	Short travel, fast workers, people free soon, and staying in the same building.

`fairness`	An even load. The employee with the most free time gets the next job.
`sla`	Finishing on time: people free right now, fast and close, and reliable.
`cost`	The cheapest hourly rate. Works only when at least two employees in the team have a cost saved; otherwise cost is ignored.

A plan that would finish a job after its due time loses 25 points for that choice, whatever the goal.

3. Authentication

Every call needs two fields:

Field	Required	Notes
`api_key`	**Required**	The company's key. It decides which company the call is for.
`actor_mopuan_user_id`	**Required**	The Mopuan employee id of the person who is using the desk.

Call these services from your server only. Your server must fill `actor_mopuan_user_id` with the employee who is signed in to your session. Never read this value from the phone or from the request of the app. If the app could choose it, any user could act as any other employee. The `api_key` must never be in the mobile app.

The actor decides two things: what the call is allowed to do (see section 4), and who is recorded as the person who made the change (for example who assigned a work order).

4. Permissions

AI SPV uses the **Work Orders** module permission of the actor's role.

Services	Permission needed
`SRVGetAiSpvDesk`, `SRVCalculateAiSpv`	Work Orders: view
All other services	Work Orders: edit

`SRVAiSpvSaveStaffCost` also needs the **Payments** module: edit permission, and the Payments module must not be switched off for the company.

You can read what a role may do with `SRVGetUserRoles` (see the Mopuan Roles and Permissions guide). The desk payload also has `canEdit`, which is `true` when the actor may edit. Hide the action buttons when it is `false`.

5. Requests and responses

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

Send every parameter as a **form field**. A JSON request body is not read.

Some fields hold a list or an object. Send them as **JSON text inside one form field**: `locks`, `skips` and `pool` on `SRVCalculateAiSpv`. Limits for each of these fields:

- At most **64 KB** of text.
- At most **5000** entries.

A field over a limit, or text that is not valid JSON, is refused with `error_code` 10051.

These services answer without the `{"result": ...}` wrapper that other OpSense services use. A success is:

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

An error is:

```
{"error_code": "10059", "error_text": "Work order is no longer open for this assignment."}
```

`error_code` is a string. Always check for `error_code` first. Use `error_code`, not `error_text`, in your code.

6. Typical flow

1. **Load the desk.** Call `SRVGetAiSpvDesk` for the date range. Show the team, the open work orders and the work already assigned.
2. **Calculate.** Call `SRVCalculateAiSpv` with the same range and the goal the user chose. Show the plan: who gets which job, and the work orders that could not be assigned.
3. **Review.** The user may lock a person to a job (`locks`), skip a job (`skips`), or remove people from the team for this calculation (`pool`). Call `SRVCalculateAiSpv` again with these choices.
4. **Commit.** For each assignment the user accepts, call `SRVAiSpvCommitAssignment`, one work order per call.
5. **Recalculate.** After any commit, call `SRVCalculateAiSpv` again. The plan depends on who already has work.

7. SRVGetAiSpvDesk

Returns everything needed to draw the desk. Read-only.

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`actor_mopuan_user_id`	**Required**	See Authentication.
`date`	Optional	First day, `Y-m-d`. Default: today.
`date2`	Optional	Last day, `Y-m-d`. Default: today plus 7 days.

Date rules. These apply to `SRVGetAiSpvDesk` and `SRVCalculateAiSpv`.

- A date must be a real calendar date in `Y-m-d` format. Anything else (`2026-02-30`, `09/11/2026`, an empty value) is replaced by its default. The other date is still used.
- If `date` is after `date2`, they are swapped.
- The range is at most **31 days**. A longer range is cut to 31 days from `date`.
- `days` in the response lists every day of the range.

```
curl -X POST "https://<api-host>/v1/SRVGetAiSpvDesk" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=345" \
 -d "date=2026-11-10" \
 -d "date2=2026-11-16"
```

Trimmed response:

```
{
 "result": "success",
 "data": {
   "lang": "en",
   "canEdit": true,
   "from": "2026-11-10",
   "to": "2026-11-16",
   "days": ["2026-11-10", "2026-11-11", "..."],
   "today": "2026-11-09",
   "historyDays": 90,
   "services": [{"id": "7", "name": "Flat Cleaning", "dur": 75, "samples": 41}],
   "locs": [{"id": "12", "name": "Harbour House", "lat": 51.5072, "lng": -0.1276}],
   "staff": [{
     "id": "U20", "dbId": 20, "name": "Jane Doe", "role": "Housekeeper",
     "photo": "https://.../uploads/users/jane.jpg", "color": "#6B63E6",
     "inPool": true, "kpi": 82, "quality": 90, "qualityOverride": true,
     "cost": 14.5, "rel": 96, "jobs": 120, "done": 115, "home": "12",
     "homeGeo": {"lat": 51.507, "lng": -0.128},
     "svc": {"7": {"jobs": 30, "done": 29, "skill": 91, "speed": 110, "avg_minutes": 68}},
     "rota": {"2026-11-10": {"off": false, "label": "06:00 - 15:00", "sh": [360, 900]}}
   }],
   "tasks": [{
     "id": "WO4521", "dbId": 4521, "title": "Flat 3B", "branchName": "Harbour House",
     "service": "7", "pri": 2, "loc": "12", "date": "2026-11-10",
     "rel": 600, "due": 780, "roomId": 88
   }],
   "existing": [{
     "id": "WO4400", "dbId": 4400, "title": "Flat 1A", "branchName": "Harbour House",
     "service": "7", "pri": 1, "loc": "12", "date": "2026-11-10",
     "rel": 540, "due": 780, "roomId": 80, "assignedTo": 20
   }]
 }
}
```

Desk fields

Field	Description
`lang`	The actor's language: `en`, `tr`, `el` or `vi`. Texts built by AI SPV (the `why` texts in section 8) use it.
`canEdit`	`true` if the actor has Work Orders edit permission.

`from`, `to`	The range that was used, after the date rules.
`days`	Every day of the range, `Y-m-d`.
`today`	Today's date.
`historyDays`	How many past days of history the scores are based on (90).
`services`	Active services: `id`, `name`, `dur` (usual minutes, 60 if nobody has timed it yet), `samples` (how many timed jobs `dur` is based on).
`locs`	Places (buildings): `id`, `name`, `lat`, `lng`. `lat` and `lng` are `null` when unknown.
`staff`	The team. See below.
`tasks`	Open, unassigned work orders in the range, at most 2000. These are what AI SPV plans. See below.
`existing`	Work orders already assigned in the range, at most 5000. They count as load but are not changed. Same fields as `tasks`, plus `assignedTo`.

Staff fields

The team is the active employees below Supervisor level.

Field	Description
`id`	Employee key, for example `U20`. Use it in `locks` and `pool`.
`dbId`	OpSense user id, for example `20`. Use it in `user_id` fields and `assignedTo`.
`name`, `role`	Full name and role name.
`photo`	Photo URL, or an empty string.
`color`	A colour for this employee, as a hex code.
`inPool`	`true` if the employee is in the planning team. Only people in the team get jobs.
`kpi`	The employee's KPI score this month (0 to 99), or `null` if there is no data.
`quality`	The quality score the plan uses (0 to 99). A manual override wins over `kpi`. `null` if neither exists.
`qualityOverride`	`true` if `quality` is a manual value.
`cost`	Hourly cost, or `null`.
`rel`	Reliability: the share of assigned jobs that were finished (0 to 100), or `null` with fewer than 3 jobs.

`jobs`, `done`	Jobs assigned and jobs finished in the history period.
`home`	The building where the employee usually works (a `locs` id), or `null`.
`homeGeo`	Where the employee's day starts, `{lat, lng}` rounded to about 100 m, or `null`.
`svc`	Per service id: `jobs`, `done`, `skill` (0 to 100), `speed` (60 to 140, 100 is the company average, or `null`), `avg_minutes`.
`rota`	Per day: `off` (`true` on a day off), `label` (rota text), `sh` (shift as `[start, end]` minutes after midnight, or `null`). A day that is not listed is a normal working day.

Task fields

Field	Description
`id`	Task key, for example `WO4521`. Use it in `locks` and `skips`.
`dbId`	The work order id. Use it as `work_order_id` when you commit.
`title`	Room name, or building name, or `#id`.
`branchName`	Building name.
`service`	Service id (a `services` id).
`pri`	Priority, 1 (normal) to 4 (urgent).
`loc`	Building id (a `locs` id).
`date`	Day of the appointment, `Y-m-d`.
`rel`	Earliest start, in minutes after midnight (the appointment time).
`due`	Latest finish, in minutes after midnight: `rel` plus 240, 180, 120 or 60 for priority 1 to 4.
`roomId`	Room id, or 0.

8. SRVCalculateAiSpv

Calculates the suggested plan. It changes nothing.

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`actor_mopuan_user_id`	**Required**	See Authentication.

`date`, `date2`	Optional	The range. Same rules as section 7.
`goal`	Optional	One of `balanced`, `quality`, `efficiency`, `fairness`, `sla`, `cost`. Default `balanced`.
`locks`	Optional	JSON object: task key to employee key, for example `{"WO4521":"U20"}`. Forces that person on that job.
`skips`	Optional	JSON array of task keys, for example `["WO4530"]`. These jobs are left unassigned.
`pool`	Optional	JSON array of employee keys, for example `["U20","U31"]`. Plans with only these people, instead of the saved team. Nothing is saved. If you leave it out, the saved team is used.

Task keys must look like `WO` and a number. Employee keys must look like `U` and a number. Anything else is refused with `error_code` 10051. An unknown key is not an error: it is ignored.

A lock is followed even when the employee would normally be blocked for that job (for example a day off). The assignment is then marked `manual`. `locks` and `skips` apply to `smart` only.

```
curl -X POST "https://<api-host>/v1/SRVCalculateAiSpv" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=345" \
 -d "date=2026-11-10" \
 -d "date2=2026-11-16" \
 -d "goal=quality" \
 --data-urlencode 'locks={"WO4521":"U20"}' \
 --data-urlencode 'skips=["WO4530"]' \
 --data-urlencode 'pool=["U20","U31","U44"]'
```

Response

`data` has the goal and two results:

Field	Description
`goal`	The goal that was used.
`smart`	The AI SPV plan for the goal. `mode` is `smart`.
`baseline`	A simple plan for comparison: the employee who is free earliest gets the job. `mode` is `simple`. It ignores `locks` and `skips`. Use it to show what AI SPV improves.

Each result has the same fields:

Field	Description
`mode`	`smart` or `simple`.
`assignments`	The jobs that got an employee.

`unassigned`	The jobs that could not be assigned, or that you skipped.
`days`	The team's day, per date (see below).
`kpi`	Summary figures for the whole plan.


```
{
 "result": "success",
 "data": {
   "goal": "quality",
   "smart": {
     "mode": "smart",
     "assignments": [{
       "t": {"id": "WO4521", "dbId": 4521, "title": "Flat 3B", "service": "7", "pri": 2, "loc": "12", "date": "2
       "day": "2026-11-10",
       "staffId": "U20",
       "manual": true,
       "score": 88.4, "base": 88.4, "pen": 0,
       "c": {"skill": 91, "quality": 90, "speed": 62.5, "proximity": 100, "avail": 100,
             "balance": 87.5, "reliability": 96, "continuity": 100, "fit": 86, "cost": 70},
       "reasons": {"skill": 91, "quality": 90, "...": "..."},
       "tr": 0, "trInfo": {"m": 0, "km": 0, "mode": "same"},
       "start": 600, "finish": 675, "dur": 75, "late": false,
       "start_time": "10:00", "end_time": "11:15",
       "cands": [{"id": "U20", "score": 88.4}, {"id": "U31", "score": 80.1}],
       "blocked": [{"id": "U44", "why": "Fully booked that day"}],
       "loadPct": 16
     }],
     "unassigned": [{
       "t": {"id": "WO4530", "dbId": 4530, "title": "Flat 9C", "...": "..."},
       "day": "2026-11-10",
       "blocked": [{"staffId": "U31", "why": "Fully booked that day", "load": 480, "cap": 480, "freeAt": 1020, "
       "blockedTotal": 12,
       "whyCounts": {"Fully booked that day": 9, "Off: Annual leave": 3},
       "manual": true
     }],
     "days": {
       "2026-11-10": [{"staffId": "U20", "load": 75, "cap": 540, "travel": 0, "jobs": 1,
                       "steps": [{"t": {"id": "WO4521", "...": "..."}, "start": 600, "finish": 675, "tr": 0, "l
     },
     "kpi": {"assigned": 52, "total": 58, "ontime": 98, "travel": 310, "match": 81.2,
             "quality": 86, "balance": 74, "cost": 612.5}
   },
   "baseline": { "mode": "simple", "...": "..." }
 }
}
```


Assignment fields

Field	Description
`t`	The task (same fields as in the desk).
`day`	The day of the job.
`staffId`	The suggested employee (a `staff` `id`). Look the person up in the desk payload.
`manual`	`true` if the person was forced by a lock.

`score`	The final score: `base` minus `pen`.
`base`	The weighted score of the ten criteria (0 to 100).
`pen`	The penalty: 25 if the job would finish after `due`, otherwise 0.
`c`	The ten criteria, each 0 to 100: `skill`, `quality`, `speed`, `proximity`, `avail`, `balance`, `reliability`, `continuity`, `fit`, `cost`. `cost` is `null` when cost is not used. `reasons` repeats the same values.
`tr`	Travel minutes before the job.
`trInfo`	Travel details: `m` (minutes), `km`, `mode` (`same`, `walk`, `transit` or `unknown`), or `null` if the start place is not known.
`start`, `finish`	Start and end, in minutes after midnight.
`start_time`, `end_time`	The same times as `HH:MM`.
`dur`	Job duration in minutes. It depends on the service and the employee's speed.
`late`	`true` if `finish` is after the task's `due`.
`cands`	The employees who could take the job, with their `score`, in the order the plan ranked them.
`blocked`	The employees who could not take the job, with `why`.
`loadPct`	The employee's load for that day after this job, in percent of their shift.

Unassigned fields

Field	Description
`t`, `day`	The task and its day.
`manual`	`true` if you skipped it. Only present then.
`blocked`	**At most 5** people who could not take the job: `staffId`, `why`, `load`, `cap` (minutes), `freeAt` (minutes after midnight), `off` (day-off text or `null`). The state shown is the one at the end of that day.
`blockedTotal`	How many people were blocked in all.
`whyCounts`	Reason text to count, over all blocked people. Use it for a summary such as "9 fully booked, 3 on leave".

The lists are capped at 5 per job to keep the response small. Use `blockedTotal` and `whyCounts` for the whole picture. A skipped job has an empty `blocked` list and a `blockedTotal` of 0.

`why` is one of: `Off: <label>`, `Cannot finish within the shift`, `Fully booked that day`, `Usually does not finish this service`. It is in the actor's language (`lang`).

Day and KPI fields

`days[<date>]` has one entry for each person in the team: `staffId`, `load` and `cap` (minutes), `travel` (minutes), `jobs` (jobs added by this plan) and `steps` (the jobs in order: `t`, `start`, `finish`, `tr`, `late`).

`kpi`:

Field	Description
`assigned`, `total`	Jobs assigned, and jobs in the range.
`ontime`	Percent of assigned jobs that finish on time.
`travel`	Total travel minutes.
`match`	Average `base` score.
`quality`	Quality of the match, weighted by priority.
`balance`	How even the load is (0 to 100, higher is more even).
`cost`	Estimated labour cost, or `null` if no employee has a cost.

9. SRVAiSpvCommitAssignment

Assigns one work order to one employee. Use it for every accepted suggestion.

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`actor_mopuan_user_id`	**Required**	See Authentication.
`work_order_id`	**Required**	The task's `dbId`.
`user_id`	**Required**	The employee's `dbId`. Must be an active employee of the company.

```
curl -X POST "https://<api-host>/v1/SRVAiSpvCommitAssignment" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=345" \
 -d "work_order_id=4521" \
 -d "user_id=20"
```

```
{"result": "success", "data": {"work_order_id": 4521, "user_id": 20}}
```

What a commit does:

1. It assigns the work order to the employee. The actor is recorded as the person who assigned it.
2. It notifies the assignee, when the assignee is new and has an email address: an email, a notification in OpSense, and a Mopuan push if the employee is linked to Mopuan.
3. It starts the work order if the company's "start when assigned" rule says so.
4. It updates linked package work: the stage, the delivery order status and the delivery work order, with their notifications.

If steps 2 to 4 fail, the assignment is still saved and the response is still a success.

Conflict refusal (10059). A commit is accepted only if the work order is still **open** (status 1) and is **unassigned or already assigned to the same employee**. If someone else took it, or it was started, finished or closed in the meantime, the call fails with `error_code` 10059 and nothing changes. Committing the same employee again is accepted and does not notify again.

When you get 10059: do not retry. Load the desk and calculate again.

10. SRVAiSpvCreateWorkOrder

Creates a new work order with no assignee. It then appears in `tasks` if its date is in the range.

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`actor_mopuan_user_id`	**Required**	See Authentication.
`branch_id`	**Required**	A building (`locs` `id`) of the company.
`service_id`	**Required**	A service (`services` `id`) of the company.
`room_id`	Optional	A room of that building. If it does not belong to the building, it is ignored.
`priority`	Optional	1 to 4. Default and fallback: 1.
`appoinment_date`	Optional	Appointment date and time, for example `2026-11-12 09:30:00`. Default (and fallback if it cannot be read): now. The field name is spelled `appoinment_date`; send it exactly like this.

```
curl -X POST "https://<api-host>/v1/SRVAiSpvCreateWorkOrder" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=345" \
 -d "branch_id=12" \
 -d "service_id=7" \
 -d "room_id=88" \
 -d "priority=2" \
 -d "appoinment_date=2026-11-12 09:30:00"
```

```
{"result": "success", "data": {"id": 4600}}
```

`id` is the new work order id. Errors: 10052 (unknown building or service), 10055 (the work order could not be created; nothing was saved).

11. SRVAiSpvTogglePool

Takes an employee out of the planning team, or puts them back. The choice is saved for the company and is used by every later desk and calculation, until you change it. To leave someone out for one calculation only, use `pool` on `SRVCalculateAiSpv`.

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`actor_mopuan_user_id`	**Required**	See Authentication.
`user_id`	**Required**	The employee's `dbId`. Must be an active employee of the company.
`excluded`	**Required**	`1` takes the employee out of the team. `0` puts them back.

```
curl -X POST "https://<api-host>/v1/SRVAiSpvTogglePool" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=345" \
 -d "user_id=20" \
 -d "excluded=1"
```

```
{"result": "success", "data": {"user_id": 20, "excluded": true}}
```

12. Quality override

A manual quality score (0 to 99) replaces the KPI score for that employee in the plan. The desk shows it as `qualityOverride: true`.

SRVAiSpvSetQualityOverride

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`actor_mopuan_user_id`	**Required**	See Authentication.
`user_id`	**Required**	The employee's `dbId`. Must be an active employee of the company.
`quality`	**Required**	A whole number from 0 to 99.

```
curl -X POST "https://<api-host>/v1/SRVAiSpvSetQualityOverride" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=345" \
 -d "user_id=20" \
 -d "quality=90"
```

```
{"result": "success", "data": {"user_id": 20, "quality": 90}}
```

SRVAiSpvClearQualityOverride

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`actor_mopuan_user_id`	**Required**	See Authentication.
`user_id`	Optional	The employee's `dbId`. **If you leave it out, or send 0, every manual quality score of the company is removed.**

```
{"result": "success", "data": {"user_id": 20}}
```

When all scores are removed, `data` is an empty list.

13. SRVAiSpvSaveStaffCost

Saves an employee's hourly cost. The cost is used by the `cost` goal and in the `kpi.cost` figure. It is the same hourly rate the Payments module uses.

Field	Required	Notes
`api_key`	**Required**	See Authentication.
`actor_mopuan_user_id`	**Required**	See Authentication.
`user_id`	**Required**	The employee's `dbId`. Must be an employee of the company.
`value`	**Required**	The hourly cost, from 0 to 9999, rounded to 2 decimals. An empty value removes the cost.

```
curl -X POST "https://<api-host>/v1/SRVAiSpvSaveStaffCost" \
 -d "api_key=<api_key>" \
 -d "actor_mopuan_user_id=345" \
 -d "user_id=20" \
 -d "value=14.50"
```

```
{"result": "success", "data": {"user_id": 20, "value": 14.5}}
```

`value` is `null` when the cost was removed. Needs Payments edit permission (see section 4). Error 10056 means the company's Payments rates are not available, so costs cannot be saved.

14. Errors

error_c	Meaning	Retry?
10015	`api_key` missing	No. Fix the request.

10016	`api_key` invalid or inactive	No. Check the key.
10030	`actor_mopuan_user_id` missing, not a number, or not an active employee of the company	No. Fix the value your server sends.
10031	The actor's role has no permission for this call (section 4)	No.
10050	`goal` is not one of the six goals	No.
10051	`locks`, `skips` or `pool` is not valid JSON, is over 64 KB or 5000 entries, or has a wrong key	No.
10052	Unknown or foreign employee, work order, building or service, or an invalid `work_order_id` or cost field	No.
10053	`excluded` is not a valid value	No.
10054	`quality` is not a whole number from 0 to 99	No.
10055	The work order could not be created	Yes, once. If it repeats, contact us.
10056	Payments rates are not available for this company	No.
10057	`value` is not a number from 0 to 9999	No.
10059	The work order is no longer open for this assignment: not open any more, or assigned to someone else	No. Refresh the desk.
10099	The request could not be completed on our side	Yes

15. Recommendations

- **Recalculate after every commit.** Each commit changes who has work, so the rest of the plan changes. Call `SRVCalculateAiSpv` again before you show the next suggestion.
- **Commit one work order per request.** Do not queue many commits in a batch. Show each result to the user.
- **Handle 10059 by refreshing.** Someone else may have changed the work order. Tell the user, load the desk and calculate again. Do not retry the same commit.
- **Do not cache the desk for long.** Staff, rotas and work orders change in the portal. Load it when the user opens the desk and after every commit.
- **Keep ranges short.** A week is usually enough. The maximum is 31 days.
- **Treat a plan as a draft.** Nothing is saved until a commit succeeds.
- **Fill `actor_mopuan_user_id` on your server,** from the signed-in session. Never from the phone.