HELP CENTER · AI INTEGRATIONS

VOGSY MCP Server — Tool Reference

What is this?

The VOGSY MCP server connects VOGSY to AI assistants that support the Model Context Protocol (MCP), such as Claude, ChatGPT, and Microsoft Copilot. Once connected, you can ask the assistant to look things up or take actions in VOGSY directly from a chat conversation — for example checking your planning, submitting a timesheet, creating a task, or drafting a new project — without needing to open the VOGSY application yourself.

The server is available at the following URL, which you'll need when setting up the connection in your AI assistant: `https://mcp.vogsy.io/`

Behind the scenes, the assistant does this by calling a set of predefined "tools." Each tool corresponds to a specific piece of functionality in VOGSY, such as reading a person's availability or creating a new deliverable. This page lists every tool that is currently available, grouped by topic, so you know what you can ask the assistant to help with.

A few things worth knowing before you start:

- **You stay in control.** Before the assistant creates or changes anything in VOGSY (a timesheet, a project, a resource allocation, and so on), it will first summarize what it's about to do and ask you to confirm.
- **The assistant looks things up first.** For most actions, it will retrieve the relevant record or form from VOGSY before making a change, so it's working with your current data and the correct options rather than guesses.
- **You don't need to know the tool names.** Just describe what you want in plain language — e.g. "log 8 hours on the Acme project for yesterday" or "who's available next week for the DiUS project?" The assistant will figure out which tools to use.
- **This list is a reference, not a menu.** You don't need to invoke tools by name; they're listed here mainly so you can see the full range of what's possible, and so more technical readers can understand how requests are handled under the hood.

The rest of this page is a more detailed, technical reference of each tool, intended for anyone who wants to understand exactly what data is read or written.

Documentation of all tools exposed by the VOGSY MCP server (RC environment). The server exposes 93 tools, organized below by functional domain. For each tool the purpose and key parameters are listed. Required parameters are marked in **bold**.

General conventions across the server:

- **Form-first pattern:** Write tools (insert/update) are paired with a `get_*_form` tool that must be called first to retrieve required fields, available options, and current values. IDs returned by forms and search tools must be passed on unchanged.
- **Confirmation:** Write tools expect an explicit user confirmation before being called.
- **Irreversible actions:** Completing an account statement and registering a payment cannot be undone through the MCP server. The assistant always asks for explicit confirmation first.
- **Dates** use `YYYY-MM-DD` format; planning tools work with `year` + `week_nr`.
- **Pagination** is cursor-based where applicable (`cursor`, `max_results`, `more` flag in responses).

---

1. Search & discovery

Tool	Purpose
`search_vogsy`	Search a single entity kind (**entity_kind**: Organization, Project, Person, Opportunity, Deliverable). Rich filtering: departments, project managers, statuses, tags, pipelines, account managers, market segments, revenue types, date ranges, contact/user/archived flags. Sort options: recentlyUpdated, orderValueHigh/Low, recentlyCreated, weatherGood. Primary entry point for finding IDs.
`search_suggest_vogsy`	Lightweight suggest/typeahead search for **entity_kind** Resource, Person or LedgerAccount. Supports start/finish date context for resource searches on a project. LedgerAccount search (by number and/or name) is used to find the ledger account for journal entries and is available to back-office users only.
`search_suggest_vogsy_employees`	Suggest search specifically for employees (e.g. to find assignees or timesheet owners).
`get_search_filters`	Returns the available filter values (departments, statuses, tags, etc.) for a given **entity_kind**, mirroring the `search_vogsy` parameter set. Use to translate names to filter IDs.

2. Organizations (CRM)

Tool	Purpose
`get_vogsy_organization_form`	Retrieves the organization form: required fields, options, and current values (when updating). Always call before insert/update.
`insert_vogsy_organization`	Creates an organization. Only **name** is required; supports full customer/supplier profile: invoice address block, invoice emails/notes, VAT number & type, IBAN/BIC, debtor/supplier number, payment condition, price list, industry, market segment, language, level of detail for invoice lines, daily rates, PDF combining.
`update_vogsy_organization`	Updates an organization (**organization_id**, **name** + any fields to change). Same field set as insert, plus `account_manager_id` (update only).
`add_contact_to_vogsy_organization`	Links an existing person (**person_id**) as contact to an organization (**organization_id**).
`get_projects_for_organization`	Active projects of an organization (max 10). Used together with the tool below for organization health reports.
`get_opportunities_for_organization`	Active opportunities of an organization (max 10).

3. Persons & leads

Tool	Purpose
`get_vogsy_person_form`	Retrieves the person form with required fields and options. Call before insert/update.
`insert_vogsy_person`	Creates a person (**first_name**, **last_name**, **email**). Supports contact details, job role/title, employee flag, auto-submit time entries, placeholder flag, and lead fields (lead stage,

	description, source, organization).
`update_vogsy_person`	Updates a person (**person_id** + name/email required). Same field set as insert.

4. Opportunities

Tool	Purpose
`get_vogsy_opportunity_form`	Retrieves the opportunity form with options (stages, pipelines, types, sources) and current values.
`insert_vogsy_opportunity`	Creates an opportunity. Required: **name**, **organization_id**, **contact_id**, **account_manager_id**, **department_id**, **pipeline_id**, **stage_id**, **revenue**, **sales_probability**, **date**, **expected_order_date**. Optional: recurring revenue, template project, type, source, expected start/completion dates.
`update_vogsy_opportunity`	Updates an opportunity (**opportunity_id** + the same required set). Follows the search → form → merge → confirm flow.
`get_opportunity_bookings`	Lists the resource bookings (expected capacity demand) of an opportunity (**opportunity_id**).
`get_opportunity_bookings_form`	Retrieves the booking form for an opportunity (**opportunity_id**): available activities, seniorities, skills and demand types. Pass `opportunity_bookings_id` to load an existing booking. Always call before insert/update.
`insert_opportunity_booking`	Adds a booking to an opportunity: **opportunity_id**, **activity_id**, **start_date**, **finish_date**. Demand is expressed in `fte` or `hours` (`demand_type` , `total_fte` , `total_hours`). Optional filters on seniorities and skills, and optional resource allocations per person with fixed hours and plan flags per weekday (Monday to Friday are planned by default).
`update_opportunity_booking`	Updates a booking (**opportunity_bookings_id** + the full required set incl. **demand_type**, **total_fte**, **total_hours**).

5. Projects & project team

Tool	Purpose
`get_vogsy_project_form`	Retrieves the project form: department options, project status options, currency options.
`insert_vogsy_project`	Creates a project. Required: **name**, **customer_id** (organization), **client_id** (contact), **department_id**, **project_status_id**, **currency_id**, **start_date**, **finish_date**. Optional `opportunity_id` to create from an opportunity.
`update_vogsy_project`	Updates an existing project (same field set).
`get_project_team`	Lists team members of a project, including the `member_id` needed for team mutations.
`insert_project_team_member`	Adds a person to a project team (**project_id**, **person_id**, **role_id**: “member” or “manager”).
`update_project_team_member`	Changes a team member’s role (**member_id** from get_project_team, not the person ID).

`delete_project_team_member`	Removes a team member (**member_id**). Requires explicit confirmation.

6. Deliverables

Tool	Purpose
`get_vogsy_deliverable_form`	Retrieves the deliverable form (**project_id** required): departments, invoice methods, revenue types.
`insert_vogsy_deliverable`	Creates a deliverable. Required: **project_id**, **name**, **department_id**, **invoice_method_id**, **revenue_type_id**, **start_date**, **finish_date**. For Fixed Price ('fp') supply `fixed_price_other_currency`. Multiple deliverables: sequential calls in chronological order.
`update_vogsy_deliverable`	Updates a deliverable (same field set + **deliverable_id**).
`move_deliverable_up` / `move_deliverable_down`	Changes the display order of a deliverable within a project.

7. Calculation lines & resource allocations

The WBS financial structure of a deliverable: activity lines (hours × rates), cost lines (amounts), and resource allocations on activity lines.

Tool	Purpose
`list_calculation_lines`	Lists all calculation lines of a deliverable, split into `activities` and `costs`. Source of `calculation_line_id`.
`get_vogsy_calculated_activity_line_form`	Form for an activity line (**deliverable_id**): exact `activity_id` values and default rates.
`insert_vogsy_calculated_activity_line`	Creates an activity line: **deliverable_id**, **activity_id**, **desc**, **start_date**, **finish_date**, plus hours, cost_rate, sales_rate, remark.
`update_vogsy_calculated_activity_line`	Updates an activity line (**calculation_line_id** + same fields).
`get_vogsy_calculated_cost_line_form`	Form for a cost line: available `cost_id` values.
`insert_vogsy_calculated_cost_line`	Creates a cost line: **cost_id**, **deliverable_id**, **desc**, **remark**, **start_date**, **finish_date**, plus cost_amount and sales_amount.
`update_vogsy_calculated_cost_line`	Updates a cost line (**calculation_line_id** + same fields).
`list_resource_allocations`	Lists resource allocations for a calculation line. Source of `resource_allocation_id`.
`insert_vogsy_resource_allocation`	Allocates a person to an activity line: **calculation_line_id**, **person_id**, **hours**, **start_date**, **finish_date**, **has_custom_sales_rate**, **generate_new_planning**. Optional custom sales/cost rates (incl. invoice/employee currency variants), per-weekday plan flags (`plan_monday` … `plan_sunday`) and fixed hours per weekday.
`update_vogsy_resource_allocation`	Updates an allocation (**resource_allocation_id** + full field set; unchanged fields keep original values).

`get_allocated_activities_in_deliverable`	Retrieves the activities of a deliverable where the (current) user is allocated. Used to find valid `calculated_activity_id` values for time entries.

8. Planning & availability

Tool	Purpose
`get_project_planning`	Planning per project (**week_nr**, **year**). Filters: departments, project managers, statuses, tags, my_projects. View in weeks or days, `group_size` weeks ahead, ordered by start date, project name or resource name.
`get_project_planning_deliverables_drilldown`	Planning per deliverable within one project. Key source of `amountToPlan` / active resources for redistribution analysis and project delay impact.
`get_project_planning_activities_drilldown`	Planning per activity within a deliverable, incl. `hoursToPlan` per resource.
`get_resource_planning`	Planning per resource across the organization. `plannings_view` : actualsAndPlanned, planned, available (for overplanned/negative availability), timeOff. Filters: resource type, departments, seniorities, skills (+ levels), tags, activities, opportunity bookings.
`get_resource_planning_drilldown`	Full planning drilldown for one person across all projects and absences. Use for availability questions; contains existing `planningId` values needed for updates. `group_size=52` for a year ahead.
`get_availability_for_resources`	Availability per resource.
`get_resource_suggest_filter_options`	Returns the valid filter options (skills, seniorities, departments, tags, and so on) for `suggest_resources`. Call this first to obtain IDs.
`suggest_resources`	Suggests employees or persons that are available for a period, based on filters such as skills (with levels), seniority, department, activities and tags. Parameters: **deliverable_id**, **start_date**, **finish_date**, plus optional filters, `is_employee` , `is_active` and `max_results` .
`get_planning_form`	Retrieves the planning form (“template”) for a resource/week. Base object for `update_planning` — do not change `id` , `version` , or `resourceAllocationId` . Context: project/deliverable/resource allocation (activity planning) or `absence_type_id` (absence planning).
`update_planning`	Inserts or updates a week planning. **days** must be exactly the 7 days from `get_planning_form` with only hours/notes changed. Supply `planning_id` when updating.

9. Timesheets

Tool	Purpose
`list_timesheets`	Lists timesheets for the current user (default) or another person via `person_id` . `max_results=1` returns the current/latest timesheet.

`get_timesheet_details`	Details of one timesheet, including `setId` values per time entry set (needed for updates).
`get_time_entry_form`	Form for creating/updating a time entry: available options (e.g. absence types) and the 7-day week structure. Always call before insert/update. Pass `set_id` when updating.
`insert_timesheet_entry`	Adds a time entry. **type**: 'activity' (requires project_id, deliverable_id, calculated_activity_id) or 'absence' (requires absence_type_id). **days**: exactly 7 day objects with date/hours, optional reference (note) per entry or per day.
`update_timesheet_entry`	Updates an existing entry (**set_id**, **timesheet_id**, **days**). Requires exact IDs (resource_allocation_id / calculated_activity_id / absence_type_id).
`submit_timesheet`	Submits a timesheet for the logged-in user. Requires explicit confirmation.

10. Invoicing, billing periods & subscriptions

Tool	Purpose
`list_draft_invoices`	Overview of all draft invoices, paginated (cursor + `more` flag).
`get_invoices_for_project`	All invoices of a project; useful for financial reports and outstanding/unpaid invoice checks.
`get_vogsy_invoice_schedule_form`	Invoicing schedule of a deliverable with existing billing periods.
`insert_vogsy_billing_period`	Adds a billing period: **deliverable_id**, **invoice_amount**, **invoice_date**, **invoice_desc**. Sequential calls when creating multiple.
`update_vogsy_billing_period`	Updates a billing period (**billing_period_id** + same fields).
`list_subscription_lines`	All subscription lines of a deliverable.
`get_subscription_line_form`	Form/current values for a subscription line.
`update_subscription_line`	Creates or updates a subscription line: **deliverable_id**, **desc**, **quantity**, **amount_per_term**, **start_date**, **finish_date**. Omit `deliverable_subscription_id` to create, supply it to update.

11. Tasks

Tool	Purpose
`list_tasks`	Search tasks by assignees, entity association (project/organization/opportunity + IDs), and due-date range (from/to). Cursor pagination.
`get_task_details`	Task details plus form options (e.g. `taskStatusOptions`). Empty `task_id` returns a blank new-task form.
`get_task_statusses`	Available task statuses per **entity_kind** (Project, Organization, Opportunity).
`insert_task`	Creates a task: **title**, **due_date**, **status_id**, **entity_id**,

	entity_kind. Optional description, assignee, linked activity, file attachments. Entity association is mandatory.
`update_task`	Updates an existing task.

12. Feed & subscriptions (following)

Tool	Purpose
`get_feed_items`	The user's chronological feed. Filters: only_actions, only_marked_as_done, only_snoozed, `until_date`. For timeframe summaries: combine an unfiltered call with a done-only call.
`mark_feed_item_as_done`	Marks one feed item as done (**feed_id**).
`mark_all_feed_items_as_done`	Marks all feed items as done. Requires explicit confirmation.
`get_entity_subscriptions`	Entities the user follows, optionally filtered by kind (Organization, Project, Person, Opportunity).

13. Entity context (activity stream per entity)

These four tools take any `entity_id` (organization, project, person, opportunity) and are designed to be called in parallel to build a health summary or report.

Tool	Purpose
`get_notes_for_entity`	Notes logged on the entity.
`get_emails_for_entity`	Emails logged on the entity.
`get_conversations_for_entity`	Conversations logged on the entity.
`get_history_for_entity`	Historic events logged on the entity.

14. Accounting: account statements & journal entries

Account statements (book entries) hold the journal entries of a journal, such as a bank journal or a general journal. Some of these tools are available to back-office users only.

Tool	Purpose
`list_account_statements`	Lists all open (not yet completed) account statements, grouped per journal. Shows which statements still need journal entries.
`get_account_statement_form`	Retrieves the form for an account statement: journal, department and accounting year/period options and, when **book_entry_id** is given, its journal entries and balances. Omit the ID to get an empty form for a new statement. Always call before insert/update.
`insert_account_statement`	Creates an account statement: **journal_id**, **date**, **department_id** (department, company or holding). Optional `accounting_year` and `accounting_period` (default to the current year and month). `start_balance` and `finish_balance` apply to bank journals only.
`update_account_statement`	Updates an open statement (**book_entry_id**, **date**). The date

	is always required, even when unchanged. Other fields are only changed when provided.
`get_journal_entry_form`	Retrieves the form for a journal entry (**book_entry_id**): ledger account and currency options and, for settlements, the prefilled outstanding amount and rate. Pass `journal_entry_id` to load an existing entry. Always call before insert/update.
`insert_journal_entry`	Books a journal entry (**book_entry_id**, **entry_type**). Type `memo` is a free-form ledger booking (ledger account, description, department and exactly one of debit or credit). Type `settlement` settles an outstanding receivable or payable (`account_receivable_id` or `account_payable_id` ; for a foreign currency also `rate` and `amount_other_currency`).
`update_journal_entry`	Updates a journal entry (**book_entry_id**, **journal_entry_id**, **entry_type**). For memo entries, description, debit and credit are all required on every update. For settlements, `rate` and `amount_other_currency` are required.
`delete_journal_entry`	Deletes a journal entry (**book_entry_id**, **journal_entry_id**). A settled amount is added back to the outstanding balance of the receivable or payable. Requires explicit confirmation.
`complete_account_statement`	Marks a statement as done (**book_entry_id**). Its journal entries are finalized and, when external accounting is configured, synced to it. The statement must be in balance (total debit equals total credit and, for a bank journal, the bank entries match the difference between start and finish balance). Cannot be undone. Requires explicit confirmation.

15. Accounts receivable & payable

Tool	Purpose
`list_account_receivables`	Lists sales invoices owed by customers, with amounts, outstanding balance, due date and currency. Filters: `status` (`out` outstanding, `rec` received), `invoice_number` , `modified_after` . Cursor pagination.
`add_account_receivable_payment`	Registers a payment received on an outstanding receivable (**account_receivable_id**, **amount**). Optional `paid_at_date` (defaults to today). `rate` is required when the invoice is not in the default currency. Cannot be undone. Requires explicit confirmation.
`list_account_payables`	Lists purchase invoices, expenses and opening balances owed to creditors, with outstanding balance, due date and approval status. Filters: `status` (`out` , `pay`), `invoice_number` , `modified_after` . Cursor pagination.
`add_account_payable_payment`	Registers a payment made on an outstanding payable (**account_payable_id**, **amount**). Optional `paid_at_date` and `rate` (required for a foreign currency). The assistant checks the approval status first and never registers a payment on a rejected payable. Cannot be undone. Requires explicit confirmation.

Common workflows

Log time: `list_timesheets` (max_results=1) → `get_time_entry_form` → `get_allocated_activities_in_deliverable` (for activity IDs) → `insert_timesheet_entry` → `submit_timesheet`.

Build a project WBS: `search_vogsy` (Project) → `get_vogsy_deliverable_form` → `insert_vogsy_deliverable` → `get_vogsy_calculated_activity_line_form` → `insert_vogsy_calculated_activity_line` → `insert_vogsy_resource_allocation`.

Redistribute unplanned work: `get_project_planning_deliverables_drilldown` + `get_project_planning_activities_drilldown` (find `hoursToPlan` / `amountToPlan`) → `get_resource_planning_drilldown` per person (availability) → `get_planning_form` → `update_planning`.

Organization health report: `search_vogsy` (Organization) → in parallel: `get_projects_for_organization`, `get_opportunities_for_organization`, `get_notes_for_entity`, `get_emails_for_entity`, `get_conversations_for_entity`, `get_history_for_entity` → drill into projects with `get_invoices_for_project`.

Book a journal entry: `list_account_statements` or `get_account_statement_form` → `search_suggest_vogsy` (LedgerAccount) → `get_journal_entry_form` → `insert_journal_entry` → `complete_account_statement`.

Register a customer payment: `list_account_receivables` (by invoice number or status) → `add_account_receivable_payment`.

Find a person for a project: `get_resource_suggest_filter_options` → `suggest_resources` → `insert_vogsy_resource_allocation`.