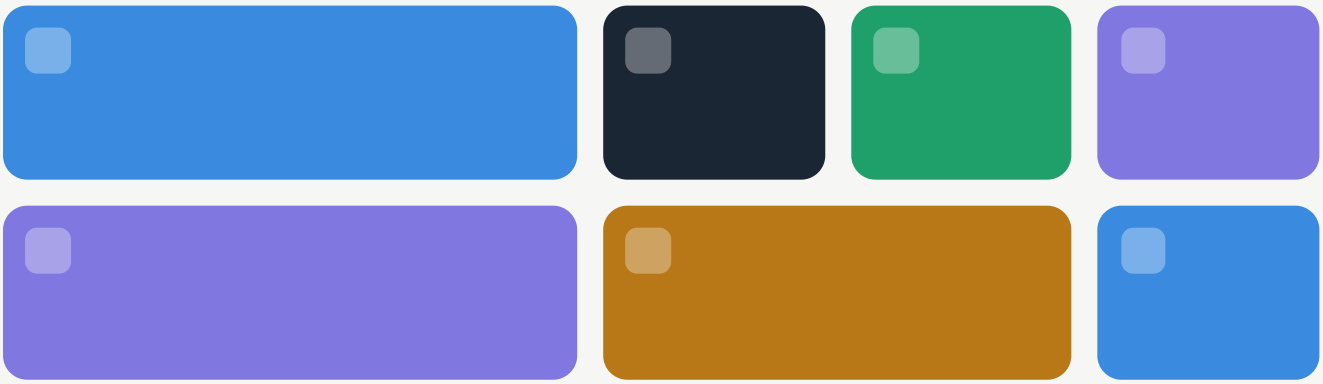
BPMSquare

CUSTOMER EDITION

Ask and MCP

How to connect other systems to BPMSquare: the REST API, the change feed and webhooks, connectors, the integration middleware and Data Workbench.

New way of customer relationship



Ask and MCP

Provided by **Snasquare**

Powered by **BPMSquare** · Snapshot 2026-10-29

CHAPTER 4

Ask and MCP

> Screen names and button words may be refined as the product improves. If a name here is slightly different on your screen, look for the same idea nearby.

About this chapter

This chapter is for developers who want people, scripts or AI assistants to ask questions of BPMSquare data without writing filters by hand. There are two tools.

- **Ask** (`POST /ask`) turns a plain-English question about one object into a query, runs it and returns the rows.
- **MCP tools** let an AI assistant call the read endpoints of the REST API as named tools.

Neither one can change data, with one exception: the MCP tool that raises a draft project invoice. Both obey the same keys, scopes and modules as the REST API (Chapter 1).

Contents

1. Ask: questions in plain English
2. What you can ask about
3. Asking well
4. MCP: tools for an AI assistant
5. The tools
6. Connect an assistant
7. Set up the key safely

---

1. Ask: questions in plain English

`POST /api/v1/ask` takes a question about one object. An AI model turns the question into a structured query. BPMSquare checks that query against the same list of fields as the normal list endpoints, runs it with your key's read scope, and returns the rows. It also returns the exact query string it used, so you can reuse it without the AI the next time.

Three things to know:

- **The model never sees your data.** It sees only your question and the names and types of the fields of the object.
- **It never writes SQL.** It chooses fields, operators and values. A field that is not on the object's list is a `422`. It is never run.

- **It can only read.** There is no write path.

Before you use it

- The key needs read access to the object you ask about (Chapter 1).
- Your workspace must own the module behind that object, or the answer is `404`.
- The AI service must be switched on for your platform. Ask your BPMSquare contact. Until it is, the answer is `503 {"error":"Not configured"}`.

Request

```
POST /api/v1/ask
Content-Type: application/json
```

Field	Type	Notes
`object`	string, required	One of the objects in section 2.
`question`	string, required	Plain English.

A ranked list

```
curl -s -X POST -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" "$BASE/ask" \
  -d '{ "object": "quotations", "question": "top 3 draft quotes over 10,000 by value" }'
```

```
{
  "question": "top 3 draft quotes over 10,000 by value",
  "object": "quotations",
  "answerable": true,
  "compiled": "/api/v1/quotations?filter=status%3Aeq%3Adraft%3Btotal%3Agt%3A10000&sort=-total&limit=3",
  "data": [
    {
      "id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
      "ref": "QT-2026-0042",
      "status": "draft",
      "total": 41050,
      "revision": 1,
      "outcome": "open",
      "loss_reason": null,
      "created_at": "2026-10-26T06:12:31.118820+00:00",
      "quote_date": "2026-10-26",
      "valid_until": "2026-11-25",
      "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
      "line_count": 2,
      "invoice_count": 0,
      "invoiced_total": 0,
      "paid_total": 0,
      "balance_due": 0,
      "_links": { "self": "/api/v1/quotations/b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03", "pdf":
"/quotations/b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03/print", "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-
9a51-0c6d2e8b1a01" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 3, "has_more": false, "generated_at": "2026-10-
28T08:36:00.000Z" }
}
```


A total

```
curl -s -X POST -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" "$BASE/ask" \
  -d '{ "object": "quotations", "question": "total value of won quotes by customer, biggest first" }'
```


```
{
  "question": "total value of won quotes by customer, biggest first",
  "object": "quotations",
  "answerable": true,
  "compiled": "/api/v1/quotations?
filter=outcome%3Aeq%3Awon&aggregate=sum%3Atotal&group_by=account.name&group_sort=sum_total",
  "data": [ { "id": "b2e4f6a8-1c3d-4e5f-8a7b-9c0d1e2f3a4b", "ref": "QT-2026-0031", "outcome": "won",
"total": 96000, "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" } }
],
  "meta": {
    "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false,
    "aggregates": { "sum_total": 96000 },
    "groups": [ { "key": "Example Trading LLC", "count": 1, "sum_total": 96000 } ],
    "generated_at": "2026-10-28T08:37:00.000Z"
  }
}
```

`data` holds the matching rows (the first page). The answer to a totals question is in `meta.aggregates` and `meta.groups`.

The `compiled` text always starts with `/api/v1/<object>`, and that is not a live path for every object. For example the sales orders are served at `/sales-orders`, not `/sales_orders`. Take the part after the `?` and put it on the real endpoint.

A question that cannot be answered

```
{
  "question": "which customers are most likely to churn",
  "object": "quotations",
  "answerable": false,
  "message": "That can't be answered from the available quotations fields. Queryable fields: id, ref,
status, total, …"
}
```

Errors

Status	Meaning
422	`Unknown object`, `Missing question`, `Could not interpret the question`, or a validation error when the AI produced a query the engine does not accept.
403	The key is not scoped to read this object.
404	Your workspace does not own the module.
502	`{"error":"AI service unavailable"}`. Try again shortly.
503	`{"error":"Not configured"}`. The AI service is not switched on. Ask your BPMSquare contact.

The call stops after 30 seconds.

2. What you can ask about

`object` can be any of these. The fields you can ask about are the object's queryable fields, which are listed with each endpoint in Chapter 2.

`object`	Fields are listed in	Scope the key needs
`quotations`	Chapter 2, section 19	`quotations`
`accounts`	section 7	`accounts`
`cases`	section 8	`cases`

object	Fields are listed in	Scope the key needs
`inventory`	section 11	`inventory`
`products`	section 15	`products`
`opportunities`	section 13	`opportunities`
`rfqs`	section 20	`rfqs` (named)
`standard_quotes`	section 22	`standard_quotes`
`sales_orders`	section 21	`sales_orders`
`abm_programs`	section 5	`abm_programs`
`abm_targets`	section 6	`abm_targets` (named)
`estimates`	section 10	`estimates` (named)
`projects`	section 17	`projects`
`project_hours`	section 16	`projects`. The `employee.*` fields also need `employees`.
`employees`	section 9	`employees` (named)
`invoices`	section 12	`invoices`
`purchase-orders`	section 18	`purchase-orders`

Four more objects can be asked about but have no endpoint of their own. They need a key with **All objects**, because a key cannot name them:

object	Fields
`assets`	`id`, `ref`, `name`, `kind`, `make`, `model`, `rating`, `serial`, `is_loaner`, `loaner_status`, `account.*`
`suppliers`	`id`, `ref`, `name`, `type`, `city`, `status`, `phone`, `email`, `created_at`
`work-orders`	`id`, `ref`, `status`, `auth_kind`, `scheduled_for`, `account.*`, `asset_name`, `technician_name`, `case_ref`
`leads`	`id`, `title`, `source`, `status`, `created_at`, `account.*`

Contacts, tenders and contract projects are not available through keys (Chapter 1, section 4).

3. Asking well

- **Name the object that holds the answer.** "Invoices overdue by more than 30 days" belongs to `invoices`, not `accounts`.
- **Use words that match fields.** "Status", "total", "customer", "due date" map to fields. "Likely to churn" does not.
- **Ask for one thing.** "Top five", "how many", "total by month" work well. A question with two unrelated parts does not.
- **Say the order and the size.** "Top 10 by value" gives a sort and a limit.
- **Check `compiled`.** If the filter is not what you meant, rewrite the question or call the list endpoint directly with the corrected query.
- **Reuse `compiled` in production.** Once a question gives the right query, save the query string and call the list endpoint. It is faster, it costs nothing extra and it does not change from day to day.
- **Do not use Ask for writes or for personal-data hunting.** It follows the key's scope like any other call.

---

4. MCP: tools for an AI assistant

MCP (Model Context Protocol) is a standard way to give an AI assistant a list of tools it can call. BPMSquare describes 34 tools. Each one is a thin wrapper around one REST endpoint from Chapter 2. A tool does nothing the REST API cannot do. The result of a tool call is the JSON of that endpoint.

BPMSquare supplies the tool list as a manifest file named `mcp.json`. It gives each tool a name, a description, an HTTP method, a path under `/api/v1` and the inputs the tool accepts. Your BPMSquare contact gives you the manifest.

To use the tools, an assistant needs a small bridge program between the assistant and BPMSquare. The bridge reads the manifest and, for each tool call, makes the matching REST call with your key. Ask your BPMSquare contact for the bridge to use with your assistant.

What the bridge does with the manifest:

1. It calls `https://<your-workspace-domain>/api/v1`.
2. It sends `Authorization: Bearer <key>` on every call.
3. For a `GET` tool, it puts `{id}` in the path from the `id` input and sends every other input as a query parameter.
4. For the one `POST` tool, it puts `{id}` in the path and sends the other inputs as the JSON body.
5. It returns the answer body to the assistant. A failure is returned as the error body from Chapter 2, section 4.

Any assistant that can send an HTTP request with a bearer token can also use the REST API directly, with the file from `GET /openapi.json` as its guide. MCP only saves you the wiring.

---

5. The tools

Read tools need a key with read access to the object. A tool for an object marked "named" works only with a key that names that object.

Tool	Calls	Inputs (besides `id`)	Scope
`list_accounts`	`GET /accounts`	None	`accounts`
`get_account`	`GET /accounts/{id}`		`accounts`
`list_cases`	`GET /cases`	`status`, `account_id`	`cases`
`list_quotations`	`GET /quotations`	`status`, `account_id`	`quotations`
`list_inventory`	`GET /inventory`	`low_stock`	`inventory`
`get_inventory_item`	`GET /inventory/{id}`		`inventory`
`list_projects`	`GET /projects`	`status`, `level`, `parent_id`, `account_id`	`projects`
`get_project`	`GET /projects/{id}`		`projects`
`get_project_hours`	`GET /projects/{id}/hours`	`from`, `to` (required)	`projects`
`get_project_invoices`	`GET /projects/{id}/invoices`		`projects`
`create_project_invoice`	`POST /projects/{id}/invoices`	`from`, `to` (required), `granularity`, `dry_run`, `top_up`	`projects`, **write**
`query_project_hours`	`GET /project-hours`	`project_id`, `account_id`, `filter`, `group_by`, `aggregates`	`projects`
`list_products`	`GET /products`	`category`, `status`, `sub_category`, `item_type`	`products`
`get_product`	`GET /products/{id}`		`products`
`list_opportunities`	`GET /opportunities`	`stage`, `outcome`, `account_id`, `owner_id`	`opportunities`
`get_opportunity`	`GET /opportunities/{id}`		`opportunities`
`list_rfqs`	`GET /rfqs`	`status`, `origin_type`, `origin_id`	`rfqs` (named)

Tool	Calls	Inputs (besides `id`)	Scope
`get_rfq`	`GET /rfqs/{id}`		`rfqs` (named)
`list_standard_quotes`	`GET /standard-quotes`	`status`, `account_id`, `opportunity_id`	`standard_quotes`
`get_standard_quote`	`GET /standard-quotes/{id}`		`standard_quotes`
`list_sales_orders`	`GET /sales-orders`	`status`, `account_id`, `source_id`	`sales_orders`
`get_sales_order`	`GET /sales-orders/{id}`		`sales_orders`
`list_abm_programs`	`GET /abm-programs`	`status`, `kind`	`abm_programs`
`get_abm_program`	`GET /abm-programs/{id}`		`abm_programs`
`list_abm_targets`	`GET /abm-targets`	`progress`, `tier`, `program_id`	`abm_targets` (named)
`get_abm_target`	`GET /abm-targets/{id}`		`abm_targets` (named)
`list_estimates`	`GET /estimates`	`status`, `account_id`	`estimates` (named)
`get_estimate`	`GET /estimates/{id}`		`estimates` (named)
`list_purchase_orders`	`GET /purchase-orders`	`status`, `supplier_id`	`purchase-orders`
`get_purchase_order`	`GET /purchase-orders/{id}`		`purchase-orders`
`list_invoices`	`GET /invoices`	`status`, `account_id`	`invoices`
`get_invoice`	`GET /invoices/{id}`		`invoices`
`list_employees`	`GET /employees`	`status`, `department`	`employees` (named)
`get_employee`	`GET /employees/{id}`		`employees` (named)

`get_*` tools and `get_project_hours`, `get_project_invoices` and `create_project_invoice` need the record's `id` as an input.

Things to know:

- **List tools return the first page only.** They accept the shortcut filters in the table and return up to 50 rows. They do not take `page`, `limit`, `sort` or `select`. For a full list, or for totals, use the REST API.
- `query_project_hours` **is the one tool with totals.** Use `group_by` for the grouping and `aggregates` for the sums, for example `sum:hours`. In the REST API the same option is called `aggregate`. The `filter` input takes the REST filter syntax: `field:op:value` clauses joined by `;`, such as `date:gte:2026-10-01;date:lte:2026-10-31`.
- `create_project_invoice` **writes.** It creates a draft invoice, or with `dry_run` only previews one. It is the only tool that needs a key with **write**. Think before you give an assistant a key that can write.
- **Not available as tools.** Creating, changing or deleting quotations, `POST /price`, `POST /ask`, the change feed, a single quotation or contact, and a single case. Use the REST API for those.

A sample call and what the bridge sends:

```
{ "name": "list_cases", "arguments": { "status": "in_repair", "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01" } }
```

```
GET /api/v1/cases?status=in_repair&account_id=3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01
```

The answer is the JSON from Chapter 2, section 8.

---

6. Connect an assistant

The steps are the same whichever assistant you use.

1. **Make a key.** In **Settings > Integrations > API keys & developer**, make a scoped key for the assistant (next section). Copy it once.
2. **Get the manifest and the bridge** from your BPMSquare contact.
3. **Give the bridge two values:** your workspace address (`https://<your-workspace-domain>/api/v1`) and the key. Put the key in the assistant host's secret store or its protected settings. Do not paste it into a chat.
4. **Restart the assistant** so it reads the tool list.
5. **Try a harmless question**, such as "How many accounts do we have?" The assistant should call `list_accounts`.
6. **Try a question the key must not answer.** Ask for employees with a key that does not name `employees`. You should get a `403`, which the assistant reports as an error.

The settings the bridge needs, in plain terms:

Setting	Value
API base	`https://<your-workspace-domain>/api/v1`
Key	The bearer token from the key screen
Tool list	The `mcp.json` file you were given

7. Set up the key safely

An assistant is a new kind of user. Give it the narrowest key that does the job.

- **Make a key just for the assistant.** Name it after the assistant, for example "Sales assistant". If you revoke it, nothing else breaks.
- **Read only.** Leave **Write** off. Tick **Write** only if you want the assistant to raise draft project invoices, and then accept that it can.
- **Only the objects it needs.** Untick **All objects** and tick, for example, Accounts, Quotations and Cases. Do not tick Employees unless the assistant truly needs staff data.
- **Set an expiry.** Ninety days is a good habit. Replace the key before it ends.
- **Do not reuse the workspace key.** It reads and writes everything a wildcard reaches.
- **Remember what the assistant can see, it can repeat.** An assistant with the `accounts` scope can read contact people's phones and emails (Chapter 1, section 7). Keep that in mind where the conversation is stored.
- **Revoke at once on a leak.** In **API keys & developer**, press **Revoke** on the key. A revoked key stops working on the next call.
- **Regenerating the workspace key stops the old one immediately.** If an assistant uses that key, it needs the new one.