BPMSquare

CUSTOMER EDITION

Integration Guide

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



Overview and security · REST API · Change feed and webhooks · Ask and MCP · Connectors and the Integration middleware · Data Workbench · Recipes

Provided by **Snasquare**

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

Contents

1 Overview and security 5

About this chapter 5
1. What you can integrate 5
2. Base URL 7
3. Keys and scopes 7
4. What a key can never reach 9
5. Modules decide what you can see 10
6. Record identity: ids, not business keys 11
7. Data protection 11
8. The demo sandbox is not for integrations 12
9. Limits 12
10. Where to go next 00

2 REST API 13

About this chapter 00
1. Conventions 00
2. Authentication and scopes 00
3. Lists, filters and totals 00
4. Errors 00
5. ABM programs 00
6. ABM targets 00
7. Accounts 00
8. Cases 00
9. Employees 00
10. Estimates 00
11. Inventory 00
12. Invoices 00
13. Opportunities (deals) 00
14. Pricing 00
15. Products 00
16. Project hours 00
17. Projects 00
18. Purchase orders 00
19. Quotations 00
20. RFQs 00
21. Sales orders 00
22. Standard quotes 00

23. API index, metadata and OpenAPI 00

3 **Change feed and webhooks** 13

About this chapter 00
1. Ask or be told 00
2. The change feed 00
3. Event webhooks 00
4. Verify the signature 00
5. Delivery, retries and idempotency 00
6. Call webhooks: one record when something happens 00
7. Push to ERP (the older way) 00
8. Telling BPMSquare that an ERP order changed 00
9. Check that it works 00

4 **Ask and MCP** 13

About this chapter 00
1. Ask: questions in plain English 00
2. What you can ask about 00
3. Asking well 00
4. MCP: tools for an AI assistant 00
5. The tools 00
6. Connect an assistant 00
7. Set up the key safely 00

5 **Connectors and the Integration middleware** 13

About this chapter 00
1. How the pieces fit 00
2. The Connectors screen 00
3. Slack 00
4. Google Calendar 00
5. Gmail 00
6. Mail server (SMTP and IMAP) 00
7. Microsoft Dynamics 365 00
8. SAP S/4HANA 00
9. ePROMIS 00
10. The Integration middleware page 00
11. What is two-way and what is read-only 00
12. The ERP id line on records 00
13. Email output and Inbox channels 00

14. A checklist for connecting an ERP 00

6 **Data Workbench** **13**

About this chapter 00

1. Where it is and who can use it 00

2. Objects available 00

3. Import 00

4. Export 00

5. Update 00

6. Custom fields 00

7. Limits and rules 00

8. A checklist for a migration 00

7 **Recipes** **13**

About this chapter 14

1. Keep an ERP in step with accounts 108

2. Verify and process a webhook safely 111

3. Create a quotation and read its status back 115

4. Load 5,000 products with Data Workbench 117

Common questions 119

CHAPTER 1

Overview and security

> 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 guide is for integration developers and consultants at a company that uses BPMSquare. You may be connecting an ERP, feeding a data warehouse, loading a catalogue, or letting an AI assistant read your records. You do not need to work at BPMSquare. You do need a workspace administrator to help with the first step, which is making a key.

This chapter tells you which door to use for which job. It also explains how access, modules, record ids and personal data work. Read it before the others. The rules here apply to every endpoint in Chapter 2.

The examples use made-up data: a customer called Example Trading LLC, addresses at `example.com`, and keys that look like `bpm_live_xxxxxxxx`. Replace them with your own.

Contents

1. What you can integrate
2. Base URL
3. Keys and scopes
4. What a key can never reach
5. Modules decide what you can see
6. Record identity: ids, not business keys
7. Data protection
8. The demo sandbox is not for integrations
9. Limits
10. Where to go next

---

1. What you can integrate

BPMSquare has eight ways in and out. They overlap on purpose. Pick the one that matches the job, not the one you already know.

You want to	Use	Direction	Chapter
Read records, filter them, total them, page through them	**REST API** (`GET`)	Out of BPMSquare	2
Create, change or delete a quotation from your own system	**REST API** (`POST`, `PATCH`, `DELETE`)	Into BPMSquare	2
Price a document against your published pricing set-up	**REST API** (`POST /price`)	Both	2
Keep another system in step with every change, by asking at your own pace	**Change feed** (`GET /changes`)	Out of BPMSquare	3
Have BPMSquare send every change to you, in signed batches	**Event webhook**	Out of BPMSquare	3
Send one record to another system when something happens (a quote is awarded, a button is pressed)	**Call webhook**, or **Push to ERP**	Out of BPMSquare	3
Tell BPMSquare that an order changed in your ERP	**Inbound order call**	Into BPMSquare	3
Ask a question in plain English and get rows back	**Ask** (`POST /ask`)	Out of BPMSquare	4
Let an AI assistant read your records	**MCP tools**	Out of BPMSquare	4
Send mail from your own mailbox, or connect Slack or a calendar	**Connectors**	Both	5
Link BPMSquare to Microsoft Dynamics 365, SAP S/4HANA or ePROMIS, map fields, watch every message, retry failures	**Connectors and Integration middleware**	Both	5
Load, extract or correct thousands of records from a spreadsheet	**Data Workbench**	Both	6

Three rules of thumb:

- If a person can do it on a screen with a spreadsheet, use **Data Workbench**. It is the safe way to load or fix data in bulk.
- If a system must do it every day without a person, use the **REST API**, the **change feed** or **webhooks**.
- If you run Dynamics 365 or SAP S/4HANA, start with **Connectors and Integration middleware**. They already know how to move customers, products and quotes. You then use the REST API only for the gaps.

Only quotations can be written through the REST API. Every other object is read-only there. To create or correct other objects in bulk, use Data Workbench.

2. Base URL

Every workspace is served on its own web address. Use that address for every call:

```
https://<your-workspace-domain>/api/v1
```

You never put a workspace id in the path. The key decides which workspace a call belongs to. A key made in one workspace cannot see another.

The samples in this guide use:

```
BASE=https://bpm.example.com/api/v1
KEY=bpm_live_xxxxxxxx
```

All responses are JSON. Cross-origin calls from a browser are allowed, and the pre-flight `OPTIONS` request is answered on every route. Even so, keep your key on a server. Never put it in a web page or a mobile app.

The webhook and connector features that call out of BPMSquare need public `https` addresses. BPMSquare refuses to call an address on a private or internal network.

---

3. Keys and scopes

A key is a bearer token. Send it on every call:

```
Authorization: Bearer bpm_live_xxxxxxxx
```

Only a workspace **administrator** can create, see or revoke keys. Go to **Settings > Integrations > API keys & developer**.

The two kinds of key

Kind	How it is made	What it can do
Scoped key	**Scoped API keys**, **+ New key**. You give a name, tick **Read** and/or **Write**, choose **All objects** or a list of objects, and optionally set an **Expires** date (a day in the future). Press **Create key**.	Exactly what you ticked.
Workspace key	The **API Key** row at the top. Press **Generate key**. There is one per workspace. **Regenerate** replaces it and the old one stops at once.	Read and write on every object that a wildcard reaches (see below).

The token of a scoped key is shown once, right after you create it. Copy it then. BPMSquare keeps only a one-way hash, so a lost key cannot be recovered. Revoke it and make another.

Prefer scoped keys. Give each integration its own key, name it after the system that uses it, give it only the objects it needs, and set an expiry. Then you can revoke one system without breaking the others.

A real scoped token starts with `bpm_` and is followed by 48 letters and digits. The list on the screen shows the first characters, when the key was last used, and its expiry. An expired or revoked key behaves like a key that does not exist: the answer is `401`.

What a scope is

A scope has three parts:

- **Read**: on by default.
- **Write**: off by default. It only matters for quotations (and for raising a project invoice, see Chapter 2).
- **Objects**: `All objects`, or a named list.

A read-only key that tries to write gets `403`. A key limited to some objects gets `403` for any other object.

The objects you can name

The key screen offers these objects: Deals, Quotations, Accounts, Cases, Inventory, Invoices, Purchase Orders, Pricing (engine), Products, Projects (workforce), RFQs (supplier prices), Estimates (quoted prices, no costs), ABM programs, Standard Quotes, Sales Orders, ABM target accounts (buying-group roles) and Employees (staff data).

Objects that "All objects" does not reach

Some objects hold sensitive data. **All objects** does not include them. A key reaches them only if you tick them by name.

Object	Why it is held back
Employees	Staff personal data.
RFQs	Supplier prices.
Estimates	The price a customer will be quoted, before it is quoted.
ABM target accounts	A buying group names people and their role.

This also protects you. If you made a wildcard key last year, a new endpoint added today does not quietly widen it to this data.

Which scope each endpoint needs

Endpoint family	Scope name	Write needed for
`/quotations`	`quotations`	`POST`, `PATCH`, `DELETE`
`/accounts`	`accounts`	
`/cases`	`cases`	
`/inventory`	`inventory`	

Endpoint family	Scope name	Write needed for
`/invoices`	`invoices`	
`/purchase-orders`	`purchase-orders`	
`/products`	`products`	
`/opportunities`	`opportunities`	
`/projects`, `/project-hours`	`projects`	`POST /projects/:id/invoices`
`/employees`	`employees` (named)	
`/rfqs`	`rfqs` (named)	
`/estimates`	`estimates` (named)	
`/standard-quotes`	`standard_quotes`	
`/sales-orders`	`sales_orders`	
`/abm-programs`	`abm_programs`	
`/abm-targets`, `/abm-programs/:id/targets`	`abm_targets` (named)	
`POST /price`	`pricing`	A read scope is enough
`POST /ask`	The scope of the object you ask about	A read scope is enough
`/changes`	One per object type, see Chapter 3	
`/`, `/openapi.json`, `/metadata`, `/metadata/:entity`	Any key with read access	

`/project-hours` rows include the person (`employee.*`) only when the key also names `employees`. Otherwise those columns are removed.

4. What a key can never reach

Some limits cannot be changed by any scope. Design your integration around them.

- **Another workspace.** A key sees one workspace only.
- **Settings and people.** There is no endpoint for users, roles, settings, keys or webhooks. You manage those on screens.

- **Contacts, tenders and contract projects.** They are not available through the API. A key cannot be made for them, and there is no endpoint. (Contact people do appear inside `GET /accounts/:id`, see Chapter 2.)
- **Product cost price, and internal cost and margin.** The cost price of a product is never returned: not in a list, a detail or the change feed. Estimates and sales orders return no cost or margin fields. The project-hours endpoint has no cost column. (Inventory items do return their unit cost, so give the `inventory` scope only where that is acceptable.)
- **Personal identity numbers.** An individual's ID number is never returned.
- **Internal notes and private fields.** Private ABM notes, RFQ internal notes and reply-link tokens are not returned.
- **Writes outside quotations.** Apart from drafting a project invoice from hours, nothing except quotations can be created, changed or deleted.
- **Bypassing a module.** If your workspace does not own the module behind an object, the endpoint answers as if it did not exist (next section).

---

5. Modules decide what you can see

Each object belongs to a module. Your BPMSquare contact switches modules on for your workspace. If a module is off, its endpoints answer `404 {"error":"Not found"}`, exactly as if the address did not exist. This is the same on the Data Workbench screens and in the menu.

Object	Module
Quotations	Quotations
Accounts	Accounts
Cases	Cases
Inventory, purchase orders	Purchasing
Invoices	Invoices
Products	Products
Deals (opportunities)	Pipeline
Projects, project hours	Workforce projects
Employees	Business roles
RFQs	RFQ
Estimates	Estimator
Standard Quotes	Standard Quotes
Sales Orders	Sales Orders

Object	Module
ABM programs, ABM target accounts	ABM
Price	Pricing engine

`POST /projects/:id/invoices` also needs the Invoices module.

If you get `404` on an endpoint that this guide documents, check two things: the module is on, and the id is one in your workspace.

---

6. Record identity: ids, not business keys

Every record BPMSquare creates gets a permanent id the moment it exists. The id is a UUID, such as `3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01`. It never changes and it is never reused. That id is the only safe thing to build on.

Business references such as `QT-2026-0042` or `ACC-0007` are for people. They are shown on screens and PDFs. Treat them as labels. A name or a reference can be renamed, can collide with a similar one, and means nothing to a different workspace.

How this works in practice:

- **When you create a record**, you cannot give it an id. It does not exist yet. You describe it by business keys instead: an account name, an account id you looked up earlier, a template name. In Data Workbench the file uses names such as `account_name` and references such as `quote_ref` to link a new record to an existing one.
- **When you read, change or delete a record**, always use its `id`. Every `GET /…/:id` and every `PATCH` and `DELETE` takes the id.
- **When you store a link in your own system**, store the BPMSquare `id` next to your own key. Do not store the reference.
- **When you update in bulk**, Data Workbench needs a `Record ID` column. Only an Export fills it in. A file with names or references alone is refused. Do not type ids by hand.
- **A wrong id fails safely.** An id that does not belong to your workspace is never matched. It answers `404`, or it fails that one row in a bulk update. It never changes someone else's data.
- **When BPMSquare links to an ERP record**, the ERP number is shown next to the BPMSquare record as a read-only line. It is the number people use in the ERP, not an internal code. See Chapter 5.

The change feed and webhooks give you `object_type` and `object_id` for every event. Match on `object_id`.

---

7. Data protection

- **Personal data is encrypted at rest.** An account's phone, second phone, email, second email and tax number are stored encrypted. So are a contact's phones and emails. They are decrypted only when the application returns them to an authorised caller.

- **The API returns what the screen shows.** `GET /accounts` returns an account's phone and email. `GET /accounts/:id` also returns the account's contact people, with their phone and email. They come under the `accounts` scope. Give a key the `accounts` scope only if the integration may see this data.
- **The change feed redacts personal fields.** In `GET /changes` and in webhook batches, changes to encrypted fields show `"[redacted]"` with `"redacted": true`. Staff phone and email are redacted the same way.
- **Webhooks send no personal data by default.** A Call webhook sends personal fields only if you switch on **Send personal data** for that webhook.
- **Cost stays internal.** The cost price of a product is never returned by the API. Change-feed items for products leave it out. Event webhooks send the change history as recorded, so choose which objects an Event webhook receives with care (Chapter 3).
- **Keys are secrets.** Scoped keys are stored as one-way hashes. Webhook signing secrets and ERP passwords are stored encrypted. They are shown once and cannot be read back. Anyone with the plain workspace key can use it, so keep it out of source code and chat messages.
- **Everything you do is traceable.** A quotation created through the API is recorded in **Change history** with the actor `api:v1`. Every Data Workbench change is recorded there with the person's name. In the Integration middleware, every message is recorded with its status. The request and answer of a message are kept for two days, with personal data and tokens removed. Bodies of inbound pulls and API calls are kept only if you switch that on.
- **Exports contain real data.** A Data Workbench export of accounts or contacts contains their personal fields. Treat the file like the data it holds.

---

8. The demo sandbox is not for integrations

BPMSquare runs a shared demo workspace at `app.bpmsquare.com`. It exists for demonstrations and training. Do not build an integration against it.

- It is shared, and its data is made up.
- It never sends real email, real webhooks or real ERP messages. A demo workspace records what it would have sent and marks it **Redirected (demo)**.

Your own workspace has its own address. While you are building, point **Settings > Integrations > Email output** at your own inbox. Then a test cannot reach a real customer (Chapter 5).

---

9. Limits

These are honest limits of the product at this snapshot. Design for them.

- **Quotations are the only writable object.** Accounts, cases, invoices, products and the rest are read-only through the API. Use Data Workbench to create or correct them in bulk.
- **No rate limit is promised.** The application does not count your calls and never answers `429` itself. That is not a promise that your hosting never will. Back off and retry if you see `429`, `502` or `503`. Do not poll faster than you need.

- **Lists are filtered in memory.** A list call loads the workspace's rows for that object and then applies your filter, sort and page. It is fast for thousands of rows and slower for hundreds of thousands. Use narrow filters, `select`, and pages of at most 200.
- **Time limits.** `POST /ask` and `POST /price` stop after 30 seconds.
- **Quotation size.** A quotation takes at most 200 lines per request.
- **Webhooks arrive in batches, not instantly.** An Event webhook is delivered on a schedule, in signed batches of up to 100 events. Ask your BPMSquare contact how often deliveries run for your workspace. If you need to know the moment something changes, poll the change feed yourself.
- **A batch can arrive again.** If your receiver answers anything but a success, the same events are sent again later. Your receiver must cope with a repeat (Chapter 3).
- **Replacing quotation lines is not atomic.** `PATCH` with `lines` updates the header, then replaces the lines. If the second step fails, the quotation can be left without lines. Read it back after a failure and repeat the call.
- **The API description is not complete.** The OpenAPI file at `GET /openapi.json` is generated by BPMSquare, but it does not list every endpoint. This guide is the complete list. See Chapter 2.
- **Some links in responses are not endpoints.** A few `_links` values, for example the `self` link on a case, point at an address that does not exist. Use the list endpoint and the filter instead.

10. Where to go next

If you are going to…	Read
Call the API	Chapter 2
Mirror data into another system, or receive events	Chapter 3
Use plain-English questions or an AI assistant	Chapter 4
Connect Dynamics 365, SAP S/4HANA, mail, Slack or a calendar	Chapter 5
Load or correct data from a spreadsheet	Chapter 6
Follow a worked example from end to end	Chapter 7

CHAPTER 2

REST API

> 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 the reference for the REST API, version 1. It lists every endpoint, what it needs, what you can filter on and what comes back. Each endpoint family has a real request and a real-shaped answer.

Read Chapter 1 first. It explains keys, scopes, modules and the limits that apply here.

All samples use made-up data. Example Trading LLC is the customer. The workspace address is `bpm.example.com`. Ids, references and amounts are invented, but the field names are exactly what the API returns. A few long answers are shortened and say so.

```
BASE=https://bpm.example.com/api/v1
KEY=bpm_live_xxxxxxxx
```

Contents

1. Conventions
2. Authentication and scopes
3. Lists, filters and totals
4. Errors
5. ABM programs
6. ABM targets
7. Accounts
8. Cases
9. Employees
10. Estimates
11. Inventory
12. Invoices
13. Opportunities (deals)
14. Pricing
15. Products
16. Project hours
17. Projects

18. Purchase orders
19. Quotations
20. RFQs
21. Sales orders
22. Standard quotes
23. API index, metadata and OpenAPI

The change feed (`/changes`) and the plain-English query (`/ask`) have their own chapters: Chapter 3 and Chapter 4.

---

1. Conventions

- **Format.** Requests and answers are JSON. Send `Content-Type: application/json` with a body.
- **Ids.** Every record has a UUID `id` . Use it in paths and links. Business references such as `QT-2026-0042` are labels (Chapter 1, section 6).
- **Dates and times.** A date is `YYYY-MM-DD` . A time stamp is ISO 8601. Stored time stamps carry microseconds and a `+00:00` offset.
- **Money.** A plain JSON number in your workspace currency. Totals of quotations are before tax.
- **Missing values.** An absent value is `null` .
- **Links.** Rows and answers carry `_links` . Follow the ones that are endpoints in this guide. A few point at addresses that do not exist, for example the `self` link on a case (section 8).
- **Detail answers.** `GET /…/:id` returns `{ "data": { … }, "_links": { … } }` .
- **Cross-origin calls** are allowed. Keep your key on a server anyway.

2. Authentication and scopes

Send `Authorization: Bearer <key>` on every call. Create keys under **Settings > Integrations > API keys & developer**. Chapter 1, section 3 explains scoped keys, read and write, the objects a key can name, and which objects "All objects" does not reach.

An unknown, expired or revoked key gets `401` . A key without the right scope gets `403` . A module your workspace does not own gets `404` .

3. Lists, filters and totals

Every list endpoint answers in the same envelope:

```
{
  "data": [ { "…": "rows" } ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-28T08:15:30.412Z" },
  "_links": { "self": "/api/v1/accounts", "next": null, "prev": null }
}
```

`meta.aggregates` and `meta.groups` appear only when you ask for them. `next` and `prev` are relative links that repeat your query string. A list of quotations also has `_links.metadata`.

Query parameters

These work on every list endpoint. All are optional and they combine.

Parameter	What it does
`select`	Comma list of fields to return. Use dots for nested fields, such as `account.name`. Only the endpoint's queryable fields can be named.
`filter`	Clauses written `field:op:value`, joined with `;` (all must match). Operators: `eq`, `ne`, `gt`, `gte`, `lt`, `lte`, `like`, `in`, `isnull`. `like` means "contains", ignoring case. `in` takes a comma list. `isnull` takes `true` or `false`. Numbers compare as numbers, dates as ISO text. URL-encode the value (`;` is `%3B`).
`search`	Case-insensitive "contains" across the endpoint's searchable fields. Combined with `filter` by AND.
`sort`	Comma list of fields. A leading `-` means descending, for example `-total,ref`.
`page`, `limit`	1-based page number. `limit` defaults to 50 and is capped at 200.
`aggregate`	`count`, `sum:field`, `avg:field`, `min:field` or `max:field`, comma separated, over the whole filtered set. The answer is in `meta.aggregates` under keys such as `count` and `sum_total`. Numeric fields only.
`group_by`	Totals per distinct value of a field, in `meta.groups` as `[{ "key", "count", … }]`. A date field is grouped by `group_period`.
`group_period`	`day`, `week`, `month` (default), `quarter` or `year`. Only with a date `group_by`. Keys look like `2026-10`, `2026-Q4`, `2026-W44`. Dates that cannot be read group under `(none)`.
`group_limit`	Keep the top N groups (maximum 100). The rest are summed into one group called `Other`.
`having`	A filter on group totals, written `key:op:value`, for example `sum_total:gt:50000`. Operators: `eq`, `ne`, `gt`, `gte`, `lt`, `lte`. Needs `group_by`.
`group_sort`	`count`, `key` or an aggregate key. Descending unless you prefix `+`.
`count`	`only` returns `meta` and no rows.

A field name the endpoint does not list is a `422` that names the accepted fields.

Each endpoint also accepts a few shortcut parameters, such as `?status=`. They are shown with the endpoint and combine with `filter` by AND.

Some row fields cannot be used in `select`, `filter` or `sort`: `_links`, `custom_data` and `counts`. They appear in an unprojected row only.

Examples

Won quotations over 10,000, biggest first, only three columns:

```
curl -s -G -H "Authorization: Bearer $KEY" "$BASE/quotations" \
  --data-urlencode "filter=outcome:eq:won;total:gte:10000" \
  --data-urlencode "sort=-total" \
  --data-urlencode "select=ref,total,account.name"
```

Total value of won quotations by month, without the rows:

```
curl -s -G -H "Authorization: Bearer $KEY" "$BASE/quotations" \
  --data-urlencode "filter=outcome:eq:won" \
  --data-urlencode "group_by=created_at" --data-urlencode "group_period=month" \
  --data-urlencode "aggregate=count,sum:total" --data-urlencode "count=only"
```

```
{
  "data": [],
  "meta": {
    "count": 0, "total": 14, "page": 1, "limit": 50, "has_more": false,
    "aggregates": { "count": 14, "sum_total": 612400 },
    "groups": [
      { "key": "2026-09", "count": 6, "sum_total": 233900 },
      { "key": "2026-10", "count": 8, "sum_total": 378500 }
    ],
    "generated_at": "2026-10-28T08:15:30.412Z"
  },
  "_links": { "self": "/api/v1/quotations", "next": null, "prev": null }
}
```

4. Errors

Status	Body	When
400	`{"error":"Malformed JSON","message":"The request body could not be parsed as JSON."}` (some routes use `{"error":"Invalid JSON body"}` or a plain message)	The body is not JSON, or a required value in the query or body is badly formed.
401	`{"error":"Unauthorized","message":"Include header: Authorization: Bearer <tenant API key>. …"}`	No key, an unknown key, a revoked or expired key. Also on the metadata endpoints for a write-only key.

Status	Body	When
403	`{"error":"Forbidden","message":"This API key is not scoped for read access to \"accounts\"."}` (`write access` for a write)	The scope does not allow that object or that direction.
404	`{"error":"Not found"}` (quotations: `"Quotation not found"`; accounts also echo `"id"`)	An unknown id, an id that is not in your workspace, or a module you do not own.
409	`{"error":"…"}`	A business conflict, such as editing a superseded quotation, or billing a period that is already billed.
422	`{"error":"Validation failed","message":"2 fields rejected. See \"details\". Full field reference: GET /api/v1/metadata/quotations","details":[{"field":"…","message":"…"}]}`	An invalid body or an invalid list query. The message always names the quotations metadata address, whichever endpoint answered.
500	`{"error":"<message>"}`	A server or database error. Repeat later.
502, 503	`{"error":"AI service unavailable"}`, `{"error":"Not configured"}`	`POST /ask` only.

Treat `5xx` and `429` as "try again later". Do not repeat a `4xx` without changing the request.

5. ABM programs

Account-based marketing programs. Scope `abm_programs` (included in "All objects"). Module ABM. A program holds no personal data. The companies inside it are a separate object (section 6).

GET /abm-programs

Shortcut parameters: `status` (`draft`, `active`, `paused`, `completed`, `archived`), `kind` (`new_business`, `expansion`).

Queryable fields: `id`, `ref`, `name`, `kind`, `status`, `goal`, `goal_metric`, `goal_target`, `default_tier`, `starts_on`, `ends_on`, `targets_count`, `created_at`. Searchable: `ref`, `name`, `goal`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/abm-programs?status=active"
```

```
{
  "data": [
    {
      "id": "6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13",
      "ref": "ABM-0002",
      "name": "Gulf contractors 2026",
      "kind": "new_business",
      "goal": "Open 5 deals with top contractors",
      "goal_metric": "deals_opened",
      "goal_target": 5,
      "starts_on": "2026-09-01",
      "ends_on": "2026-12-31",
      "status": "active",
      "default_tier": "one_to_few",
      "owner_user_id": "11111111-2222-4333-8444-555555555555",
      "custom_data": null,
      "created_at": "2026-08-30T10:02:11.482113+00:00",
      "updated_at": "2026-10-20T07:44:09.113002+00:00",
      "targets_count": 12,
      "_links": {
        "self": "/api/v1/abm-programs/6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13",
        "targets": "/api/v1/abm-programs/6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13/targets"
      }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:15:30.412Z" },
  "_links": { "self": "/api/v1/abm-programs", "next": null, "prev": null }
}
```


`goal_metric` is one of `deals_won`, `deals_opened`, `accounts_engaged`, `revenue`.

GET /abm-programs/:id

One program with its funnel, counted by company and never by person.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/abm-programs/6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13"
```

```
{
  "data": {
    "id": "6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13",
    "ref": "ABM-0002",
    "name": "Gulf contractors 2026",
    "kind": "new_business",
    "status": "active",
    "default_tier": "one_to_few",
    "funnel": { "targets": 12, "reached": 9, "engaged": 6, "inTalks": 3, "deals": 2, "won": 1, "lost": 0,
"dropped": 1 },
    "by_tier": [
      { "tier": "one_to_one", "targets": 2, "engaged": 2, "deals": 1, "won": 1 },
      { "tier": "one_to_few", "targets": 6, "engaged": 3, "deals": 1, "won": 0 },
      { "tier": "one_to_many", "targets": 4, "engaged": 1, "deals": 0, "won": 0 }
    ],
    "money": { "openValue": 480000, "wonValue": 150000, "openDeals": 1, "wonDeals": 1 },
    "_links": {
      "self": "/api/v1/abm-programs/6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13",
      "targets": "/api/v1/abm-programs/6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13/targets"
    }
  },
  "_links": { "self": "/api/v1/abm-programs/6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13" }
}
```


The answer also carries every field of the list row (shortened here). An unknown id is `404 {"error":"Not found"}`.

---

6. ABM targets

One company inside one program. Scope `abm_targets`, which you must name (Chapter 1, section 3), because a buying group names people and their roles. Module ABM.

GET /abm-targets

Shortcut parameters: `progress` (`planned`, `reached`, `engaged`, `in_talks`, `deal_open`, `won`, `lost`, `dropped`), `tier` (`one_to_one`, `one_to_few`, `one_to_many`), `program_id`.

Queryable fields: `id`, `program_id`, `program_ref`, `program_name`, `account.id`, `account.name`, `tier`, `progress`, `progress_source`, `progress_changed_at`, `next_step`, `next_step_due`, `opportunity_id`, `people_count`, `created_at`. Searchable: `program_ref`, `program_name`, `account.name`, `next_step`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/abm-targets?progress=engaged"
```

```
{
  "data": [
    {
      "id": "701c5e29-3b68-4fa4-b7d1-0c2f6b4a8d14",
      "program_id": "6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13",
      "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
      "tier": "one_to_one",
      "progress": "engaged",
      "progress_source": "rule",
      "progress_changed_at": "2026-10-12T09:30:00+00:00",
      "owner_user_id": "11111111-2222-4333-8444-555555555555",
      "next_step": "Site visit with procurement",
      "next_step_due": "2026-11-05",
      "opportunity_id": null,
      "custom_data": null,
      "created_at": "2026-09-02T06:10:44.221903+00:00",
      "updated_at": "2026-10-12T09:30:00.118322+00:00",
      "program_ref": "ABM-0002",
      "program_name": "Gulf contractors 2026",
      "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
      "people_count": 3,
      "_links": {
        "self": "/api/v1/abm-targets/701c5e29-3b68-4fa4-b7d1-0c2f6b4a8d14",
        "program": "/api/v1/abm-programs/6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13",
        "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
      }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:16:02.004Z" },
  "_links": { "self": "/api/v1/abm-targets", "next": null, "prev": null }
}
```


`progress_source` is `rule` (set by an automatic rule) or `manual` (set by a person). Private notes and the reasons for progress or dropping a target are never returned.

GET /abm-targets/:id

The same row, plus `buying_group`: the contact id and role of each person, with no names, contact details or notes.

```
{
  "data": {
    "id": "701c5e29-3b68-4fa4-b7d1-0c2f6b4a8d14",
    "tier": "one_to_one",
    "progress": "engaged",
    "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
    "people_count": 1,
    "buying_group": [
      {
        "id": "81d2f6a3-4c79-4e05-98b1-1d3a7c5b9e20",
        "contact_id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02",
        "role": "decision_maker",
        "is_primary": true,
        "_links": { "contact": "/api/v1/contacts/5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02" }
      }
    ],
    "_links": { "self": "/api/v1/abm-targets/701c5e29-3b68-4fa4-b7d1-0c2f6b4a8d14" }
  },
  "_links": { "self": "/api/v1/abm-targets/701c5e29-3b68-4fa4-b7d1-0c2f6b4a8d14" }
}
```


The other row fields are present too (shortened here). `role` is one of `decision_maker`, `influencer`, `user`, `blocker`, `unknown`. The `contact` link is not an endpoint. To read the person, use `GET /accounts/:id`.

GET /abm-programs/:id/targets

The companies of one program. Same rows, parameters and scope as `GET /abm-targets` (scope `abm_targets`). The `self` link is `/api/v1/abm-programs/:id/targets`. An unknown program id gives an empty list, not `404`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/abm-programs/6f0b4d18-2a57-4e93-a6c0-9b1e5a3f7c13/targets?sort=-next_step_due"
```

---

7. Accounts

Companies you sell to. Scope `accounts`. Module Accounts. Read only.

GET /accounts

Shortcut parameter: `type` (`prospect`, `oem`, `direct`, `end_customer`).

Queryable fields: `id`, `name`, `type`, `city`, `phone`, `email`, `created_at`, `referred_by.name`. Searchable: `name`, `city`, `email`. Phone and email are stored encrypted and are decrypted for you.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/accounts?type=end_customer&search=example"
```

```
{
  "data": [
    {
      "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
      "name": "Example Trading LLC",
      "type": "end_customer",
      "city": "Dubai",
      "phone": "+971 4 555 0147",
      "email": "procurement@example.com",
      "referred_by": null,
      "created_at": "2026-03-04T05:22:10.331042+00:00",
      "counts": { "contacts": 2, "cases": 1, "assets": 3, "contracts": 0, "quotes": 4, "workOrders": 1,
"invoices": 2 },
      "_links": { "self": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:17:40.220Z" },
  "_links": { "self": "/api/v1/accounts", "next": null, "prev": null }
}
```


`referred_by` is `{ "id", "name" }` or `null`.

GET /accounts/:id

The whole account with what is linked to it. The fields follow the account record. In a workspace with e-invoicing the buyer fields appear too. An individual's ID number is never returned.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
```

```
{
  "data": {
    "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
    "tenant_id": "00000000-0000-4000-8000-000000000001",
    "ref": "ACC-0007",
    "name": "Example Trading LLC",
    "type": "end_customer",
    "address_line1": "Plot 14, Industrial Area 3",
    "address_line2": null,
    "city": "Dubai",
    "state": null,
    "postal_code": null,
    "country": "AE",
    "phone": "+971 4 555 0147",
    "phone2": null,
    "email": "procurement@example.com",
    "email2": null,
    "website": "https://www.example.com",
    "industry": "Trading",
    "employee_count": null,
    "annual_revenue": null,
    "gstin": null,
    "notes": null,
    "referred_by_account_id": null,
    "created_at": "2026-03-04T05:22:10.331042+00:00",
    "custom_data": null,
    "marketing_opt_out": false,
    "owner_user_id": null,
    "referred_by": null,
    "contacts": [
      {
        "id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02",
        "ref": "CON-0003",
        "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
        "name": "Sara Khan",
        "role": "Procurement manager",
        "department": "Purchasing",
        "phone": "+971 50 555 0182",
        "email": "sara.khan@example.com",
        "custom_data": null
      }
    ],
    "sites": [],
    "assets": [],
    "cases": [],
    "quotes": [
      { "id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03", "ref": "QT-2026-0042", "status": "draft", "total":
41050, "created_at": "2026-10-26T06:12:31.118820+00:00" }
    ],
    "contracts": [],
    "work_orders": [
      { "id": "ee3b1c52-7a94-4c06-8d1f-2b5a9e7c0d21", "ref": "WO-0011", "status": "scheduled",
"scheduled_for": "2026-11-03", "authorized_by": { "kind": "quote", "id": "b7d1e0c2-5a3f-4e18-8c92-
6f4a1d9e2b03" } }
    ],
    "invoices": []
  },
```

```
  "meta": { "generated_at": "2026-10-28T08:18:12.907Z" },
  "_links": { "self": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "collection":
"/api/v1/accounts" }
}
```

Some arrays are shown empty to keep the sample short. The `contacts` rows carry the contact record: phone and email are decrypted, and fields such as `phone2`, `website`, address fields, notes and birthday are present too. The `assets`, `cases`, `contracts` and `invoices` rows carry their own records. `sites` is always `[]` today.

Contact people are returned under the `accounts` scope. Give an integration that scope only if it may see them.

Errors: `404 {"error":"Not found","id":"…"}`.

---

8. Cases

Service cases (repair jobs). Scope `cases`. Module Cases. Read only. There is a list and no detail endpoint.

GET /cases

Shortcut parameters: `status` (`intake`, `inspection`, `report_sent`, `report_approved`, `quote_sent`, `quote_approved`, `in_repair`, `qa`, `ready`, `closed`, `buyback`, `scrapped`), `account_id`.

Queryable fields: `id`, `ref`, `type`, `status`, `equipment_label`, `complaint`, `disposition`, `has_loaner`, `intake_at`, `closed_at`, `account.id`, `account.name`, `technician_name`. Searchable: `ref`, `equipment_label`, `complaint`, `account.name`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/cases?status=in_repair&account_id=3f2a9c1e-7b44-4d0a-9a51-
0c6d2e8b1a01"
```

```
{
  "data": [
    {
      "id": "c41d8f06-9e2b-47a3-b5c1-3a7e0d9f6b06",
      "ref": "CASE-0031",
      "type": "adhoc",
      "status": "in_repair",
      "equipment_label": "22 kW 3-ph induction motor, serial EX-2291",
      "complaint": "Overheating under load",
      "disposition": "repair",
      "has_loaner": false,
      "intake_at": "2026-10-20T07:05:00+00:00",
      "closed_at": null,
      "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
      "technician_name": "R. Menon",
      "_links": {
        "self": "/api/v1/cases/c41d8f06-9e2b-47a3-b5c1-3a7e0d9f6b06",
        "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
      }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-28T08:19:01.511Z" },
  "_links": { "self": "/api/v1/cases", "next": null, "prev": null }
}
```


`type` is `amc`, `adhoc` or `direct`. `disposition` is `repair`, `buyback`, `scrap` or `null`. The row's `self` link points at `/api/v1/cases/:id`, which is not an endpoint. To find one case, filter the list: `?filter=ref:eq:CASE-0031`.

---

9. Employees

Staff records. These are personal data. Scope `employees`, which you must name. The workspace also needs the Business roles module. Without it the answer is `404` with the message "The employees module isn't enabled for this workspace."

GET /employees

Shortcut parameters: `status` (`active`, `inactive`), `department`.

Queryable fields: `id`, `employee_code`, `first_name`, `last_name`, `email`, `phone`, `department`, `designation`, `status`, `employment_type`, `wfm_role`, `valid_from`, `valid_to`, `created_at`. Searchable: `employee_code`, `first_name`, `last_name`, `email`, `department`, `designation`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/employees?status=active&department=Workshop"
```

```
{
  "data": [
    {
      "id": "923e7a4b-5d8a-4b26-99f3-2e4b8d6c0f16",
      "employee_code": "EMP-0042",
      "first_name": "Ravi",
      "last_name": "Menon",
      "email": "ravi.menon@example.com",
      "phone": "+971 50 555 0120",
      "department": "Workshop",
      "designation": "Technician",
      "status": "active",
      "employment_type": "full_time",
      "wfm_role": "employee",
      "valid_from": "2025-01-01",
      "valid_to": null,
      "created_at": "2025-01-02T06:00:00.000000+00:00",
      "custom_data": null,
      "_links": { "self": "/api/v1/employees/923e7a4b-5d8a-4b26-99f3-2e4b8d6c0f16" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:20:11.003Z" },
  "_links": { "self": "/api/v1/employees", "next": null, "prev": null }
}
```


GET /employees/:id

The detail answer is the complete employee record, not the shorter list shape. Besides the list fields it carries internal columns such as `tenant_id`, `updated_at`, `shift_id`, `site_id`, `technician_id`, `self_service_override`, `remote_mode`, `enrolled_photo_path` and `consent_recorded_at`.

```
{
  "data": {
    "id": "923e7a4b-5d8a-4b26-99f3-2e4b8d6c0f16",
    "tenant_id": "00000000-0000-4000-8000-000000000001",
    "employee_code": "EMP-0042",
    "first_name": "Ravi",
    "last_name": "Menon",
    "email": "ravi.menon@example.com",
    "phone": "+971 50 555 0120",
    "department": "Workshop",
    "designation": "Technician",
    "valid_from": "2025-01-01",
    "valid_to": null,
    "status": "active",
    "employment_type": "full_time",
    "shift_id": null,
    "site_id": null,
    "wfm_role": "employee",
    "technician_id": null,
    "custom_data": null,
    "created_at": "2025-01-02T06:00:00.000000+00:00",
    "updated_at": "2026-09-01T06:00:00.000000+00:00",
    "_links": { "self": "/api/v1/employees/923e7a4b-5d8a-4b26-99f3-2e4b8d6c0f16" }
  },
  "_links": { "self": "/api/v1/employees/923e7a4b-5d8a-4b26-99f3-2e4b8d6c0f16" }
}
```


The sample is shortened. Do not build on the internal columns. They can change.

---

10. Estimates

Project estimates built from a template. Scope `estimates`, which you must name. Module Estimator. Only customer-safe fields are returned. Costs, margin, parameters, the model and the bill of materials are never returned.

GET /estimates

Shortcut parameters: `status` (`draft`, `quoted`, `won`, `lost`), `account_id`.

Queryable fields: `id`, `ref`, `name`, `status`, `currency`, `quoted_total`, `quoted_total_incl_vat`, `account.id`, `account.ref`, `account.name`, `contact.id`, `contact.ref`, `contact.name`, `template.name`, `created_at`, `updated_at`. Searchable: `ref`, `name`, `account.name`, `contact.name`, `template.name`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/estimates?status=quoted"
```

```
{
  "data": [
    {
      "id": "812d6f3a-4c79-4a15-88e2-1d3a7c5b9e15",
      "ref": "EST-0009",
      "name": "Warehouse mezzanine, 400 m2",
      "status": "quoted",
      "currency": "AED",
      "quoted_total": 182500,
      "quoted_total_incl_vat": 191625,
      "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "ref": "ACC-0007", "name": "Example Trading
LLC" },
      "contact": { "id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02", "ref": "CON-0003", "name": "Sara Khan" },
      "template": { "name": "Steel structure" },
      "custom_data": null,
      "created_at": "2026-10-10T06:00:00.000000+00:00",
      "updated_at": "2026-10-14T11:20:00.000000+00:00",
      "_links": { "self": "/api/v1/estimates/812d6f3a-4c79-4a15-88e2-1d3a7c5b9e15" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:21:30.700Z" },
  "_links": { "self": "/api/v1/estimates", "next": null, "prev": null }
}
```


`quoted_total` and `quoted_total_incl_vat` are `null` until the estimate has a price. `account`, `contact` and `template` are `null` when not set.

GET /estimates/:id

The same object as one list row, wrapped as `{ "data": { … }, "_links": { "self": "/api/v1/estimates/:id" } }`. An id that is not a UUID is `404`.

---

11. Inventory

Stocked spare parts and materials. Scope `inventory`. Module Purchasing. Read only.

GET /inventory

Shortcut parameters: `status` (`active`, `inactive`), `low_stock=true` (only items at or below their reorder level).

Queryable fields: `id`, `sku`, `name`, `description`, `category`, `uom`, `qty_on_hand`, `reorder_level`, `unit_cost`, `supplier_id`, `status`. Searchable: `sku`, `name`, `description`, `category`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/inventory?low_stock=true&sort=qty_on_hand"
```

```
{
  "data": [
    {
      "id": "7d2b9e40-1c85-4a6f-93d8-e5a0b1c2d307",
      "sku": "BRG-6310-2RS",
      "name": "Deep groove ball bearing 6310-2RS",
      "description": "50 x 110 x 27 mm, sealed",
      "category": "Bearings",
      "uom": "Nos",
      "qty_on_hand": 4,
      "reorder_level": 6,
      "unit_cost": 38.5,
      "supplier_id": "2e6a4c18-8f03-4b97-a1e5-d9c7b3f0a508",
      "status": "active",
      "custom_data": null,
      "_links": { "self": "/api/v1/inventory/7d2b9e40-1c85-4a6f-93d8-e5a0b1c2d307" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:22:44.100Z" },
  "_links": { "self": "/api/v1/inventory", "next": null, "prev": null }
}
```


GET /inventory/:id

The complete inventory record (all columns) plus `_links`. It does not include stock movements.


```
{
  "data": {
    "id": "7d2b9e40-1c85-4a6f-93d8-e5a0b1c2d307",
    "ref": "INV-0018",
    "tenant_id": "00000000-0000-4000-8000-000000000001",
    "sku": "BRG-6310-2RS",
    "name": "Deep groove ball bearing 6310-2RS",
    "description": "50 x 110 x 27 mm, sealed",
    "category": "Bearings",
    "uom": "Nos",
    "supplier_id": "2e6a4c18-8f03-4b97-a1e5-d9c7b3f0a508",
    "qty_on_hand": 4,
    "reorder_level": 6,
    "unit_cost": 38.5,
    "status": "active",
    "notes": null,
    "custom_data": null,
    "created_at": "2026-02-11T09:00:00.000000+00:00",
    "_links": { "self": "/api/v1/inventory/7d2b9e40-1c85-4a6f-93d8-e5a0b1c2d307" }
  },
  "_links": { "self": "/api/v1/inventory/7d2b9e40-1c85-4a6f-93d8-e5a0b1c2d307" }
}
```


Errors: `404 {"error":"Not found"}`.

12. Invoices

Customer invoices. Scope `invoices`. Module Invoices. Read only. (A draft invoice can be raised from project hours, section 17.)

GET /invoices

Shortcut parameters: `status` (`draft`, `sent`, `partial`, `paid`, `overdue`, `cancelled`), `account_id`.

Queryable fields: `id`, `ref`, `status`, `account.id`, `account.name`, `quote_id`, `work_order_id`, `due_date`, `total`, `paid_amount`, `issued_at`, `einvoice_doc_type`, `einvoice_ready`. Searchable: `ref`, `account.name`. The two e-invoice fields exist only in a workspace with Saudi e-invoicing switched on.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/invoices?status=overdue&aggregate=count,sum:total"
```

```
{
  "data": [
    {
      "id": "61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709",
      "ref": "INV-2026-0113",
      "status": "overdue",
      "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
      "quote_id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
      "work_order_id": null,
      "due_date": "2026-10-15",
      "total": 41050,
      "paid_amount": 10000,
      "issued_at": "2026-09-15T08:00:00+00:00",
      "_links": { "self": "/api/v1/invoices/61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "aggregates": { "count": 1,
"sum_total": 41050 }, "generated_at": "2026-10-28T08:23:30.900Z" },
  "_links": { "self": "/api/v1/invoices", "next": null, "prev": null }
}
```

GET /invoices/:id


```
{
  "data": {
    "id": "61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709",
    "ref": "INV-2026-0113",
    "status": "overdue",
    "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
    "quote_id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
    "work_order_id": null,
    "case_id": null,
    "contract_id": null,
    "due_date": "2026-10-15",
    "notes": null,
    "terms": "Net 30",
    "total": 41050,
    "paid_amount": 10000,
    "balance_due": 31050,
    "custom_data": null,
    "issued_at": "2026-09-15T08:00:00+00:00",
    "lines": [
      { "id": "a1c3e5b7-2d49-4f60-8b1a-3c5e7a9b1d22", "description": "Supply of 3-phase induction motor 22
kW", "uom": "Nos", "qty": 2, "rate": 18500, "amount": 37000 },
      { "id": "b2d4f6c8-3e5a-4071-9c2b-4d6f8b0c2e33", "description": "Installation and commissioning",
"uom": "Lot", "qty": 1, "rate": 4500, "amount": 4050 }
    ],
    "payments": [
      { "amount": 10000, "paid_on": "2026-10-01", "method": "bank_transfer" }
    ],
    "_links": { "self": "/api/v1/invoices/61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709" }
  },
  "_links": { "self": "/api/v1/invoices/61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709" }
}
```


A payment's `method` is the text recorded with the payment. In a workspace with e-invoicing the invoice also carries `einvoice_doc_type`, `einvoice_ready`, `einvoice_uuid`, `einvoice_seq`, `original_invoice_id`, `credit_reason` and `supply_date`, and each line carries `vat_category`, `vat_rate` and `vat_exemption_reason`. Errors: `404`.

---

13. Opportunities (deals)

The sales pipeline. Scope `opportunities`. Module Pipeline.

GET /opportunities

Shortcut parameters: `stage`, `outcome` (`open`, `won`, `lost`, `dropped`), `account_id`, `owner_id`. The default stages are `qualify`, `propose`, `negotiate`, `won`, `lost`. Each workspace can change the list.

Queryable fields: `id`, `ref`, `title`, `account_id`, `account_name`, `stage`, `outcome`, `amount`, `probability`, `expected_close`, `owner_id`, `source`, `competitor`, `created_at`, `closed_at`. Searchable: `ref`, `title`, `account_name`, `competitor`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/opportunities?outcome=open&sort=-amount"
```


```
{
  "data": [
    {
      "id": "d3b7f1a4-62e8-45c9-9d0b-7a2e4c6f8b0b",
      "ref": "OPP-0021",
      "title": "22 kW motor replacement programme",
      "description": null,
      "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
      "contact_id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02",
      "stage": "propose",
      "outcome": "open",
      "loss_reason": null,
      "amount": 41050,
      "currency": "AED",
      "probability": 50,
      "expected_close": "2026-12-01",
      "owner_id": "11111111-2222-4333-8444-555555555555",
      "source": "referral",
      "competitor": null,
      "custom_data": null,
      "created_at": "2026-10-05T07:00:00.000000+00:00",
      "updated_at": "2026-10-26T06:13:00.000000+00:00",
      "closed_at": null,
      "account_name": "Example Trading LLC",
      "_links": {
        "self": "/api/v1/opportunities/d3b7f1a4-62e8-45c9-9d0b-7a2e4c6f8b0b",
        "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
      }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:24:20.000Z" },
  "_links": { "self": "/api/v1/opportunities", "next": null, "prev": null }
}
```


Internal columns (team and the reason for a probability override) are never returned.

GET /opportunities/:id

The list row plus `loss_note`, `lines`, and `quotes` (the standard quotes and quotations made from the deal, newest first).

```
{
  "data": {
    "id": "d3b7f1a4-62e8-45c9-9d0b-7a2e4c6f8b0b",
    "ref": "OPP-0021",
    "title": "22 kW motor replacement programme",
    "stage": "propose",
    "outcome": "open",
    "loss_reason": null,
    "loss_note": null,
    "amount": 41050,
    "currency": "AED",
    "probability": 50,
    "account_name": "Example Trading LLC",
    "lines": [
      {
        "id": "c3e5a7d9-4f5b-4182-8d3c-5e7a9c1d3f44",
        "sl_no": "1",
        "description": "22 kW IE3 motor",
        "uom": "Nos",
        "qty": 2,
        "rate": 18500,
        "discount_pct": 0,
        "amount": 37000,
        "product_id": "a85c2e71-0d94-4f3b-b6a8-1e7d3c9f5a0a",
        "group_id": null,
        "group_label": null,
        "break_of": null,
        "break_qty": null,
        "is_selected": true
      }
    ],
    "quotes": [
      { "object": "quotation", "id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03", "ref": "QT-2026-0042",
"status": "draft", "subtotal": 41050, "total": 41050, "created_at": "2026-10-26T06:12:31.118820+00:00" }
    ],
    "_links": {
      "self": "/api/v1/opportunities/d3b7f1a4-62e8-45c9-9d0b-7a2e4c6f8b0b",
      "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
    }
  },
  "_links": { "self": "/api/v1/opportunities/d3b7f1a4-62e8-45c9-9d0b-7a2e4c6f8b0b" }
}
```


The other list fields are present too (shortened here). `quotes[].object` is `quotation` or `standard_quote`.

---

14. Pricing

Prices a document against your workspace's published pricing set-up. Scope `pricing`. A read scope is enough, although the method is `POST`. Module Pricing engine.

POST /price

Body:

Field	Type	Notes
`document.attributes`	object	Header attributes the rules match on, such as `document_type`, `customer.*`, `region`. Optional.
`document.lines`	array, required	1 to 200 lines.
`document.lines[].line_no`	integer	Defaults to `(index + 1) × 10`.
`document.lines[].attributes`	object	Product and line attributes the rules match on.
`document.lines[].quantity`, `.weight_kg`	number	Must be numbers.
`document.lines[].cost_items`	array	`[{ "path": string, "qty": number }]`.
`document.lines[].manual`	object	`{ CODE: { value, reason? } }`. Allowed only where a component's override policy permits it.
`options.trace`	`full` (default), `summary`, `none`	How much of the working to return.
`options.pricing_date`	`YYYY-MM-DD`	Defaults to today.
`options.config_version`	integer	Price against a specific version of the set-up. Defaults to the published one.
`options.pricing_area`, `options.procedure`, `options.currency`	string	`procedure` is required when the version has more than one.
`options.replay_of`	UUID	Re-price a stored pricing document instead of sending `document.lines`.

```
curl -s -X POST -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" "$BASE/price" -d '{
  "document": {
    "attributes": { "document_type": "quotation", "customer": { "group": "contractor" }, "region": "AE" },
    "lines": [
      { "line_no": 10, "quantity": 2, "attributes": { "product": { "category": "Motors" } } }
    ]
  },
  "options": { "trace": "summary", "pricing_date": "2026-10-28" }
}'
```

```
{
  "data": {
    "pricing_date": "2026-10-28",
    "currency": "AED",
    "totals": { "net": 36000, "subtotals": { "NET": 36000 } },
    "lines": [
      {
        "line_no": 10,
        "net": 36000,
        "subtotals": { "NET": 36000 },
        "components": { "LIST": 37000, "DISC": -1000 },
        "trace": [
          { "step": 10, "component": "LIST", "status": "APPLIED", "calc_type": "PER_UNIT", "value": 18500,
"qty": 2, "result": 37000 },
          { "step": 20, "component": "DISC", "status": "APPLIED", "calc_type": "PERCENT", "value": -2.7,
"basis": 37000, "result": -1000 },
          { "step": 30, "subtotal": "NET", "status": "SUBTOTAL", "result": 36000 }
        ]
      }
    ]
  },
  "meta": {
    "config_version": 7,
    "procedure": "STANDARD",
    "calc_ms": 12,
    "trace": "summary",
    "generated_at": "2026-10-28T08:26:00.000Z",
    "document_id": "9b8a7c6d-5e4f-4031-b2a1-0f9e8d7c6b5a"
  },
  "_links": { "self": "/api/v1/price" }
}
```


Component codes, subtotal names, the procedure code and the amounts come from your own pricing set-up. The values above are only an illustration. A trace step can carry `step`, `component`, `subtotal`, `status`, `reason`, `rule_id`, `matched_on`, `specificity`, `calc_type`, `basis`, `value`, `qty`, `formula`, `values_used` and `result`.

With `trace: "full"`, every step is returned, including those marked `EXCLUDED` and `SKIPPED`. `summary` returns only `APPLIED` and `SUBTOTAL` steps. `none` leaves `trace` out.

Every call is stored. Its id is in `meta.document_id`. Send `"options": { "replay_of": "<document_id>" }` to price it again. The answer then echoes `meta.replay_of`.

Errors:

Status	Body
422	`{"error":"Invalid document","message":"document.lines must be a non-empty array."}`, and likewise `Invalid options`
422	`{"error":"AMBIGUOUS_RULE","message":"…"}`, `MISSING_REQUIRED_COMPONENT`, `FORMULA_ERROR` and the other pricing codes in `error`
409	`{"error":"Pricing configuration","message":"…"}`, for example when no pricing set-up is published

Status	Body
500	`{"error":"Pricing failed","message":"Unexpected error while pricing the document."}`

15. Products

The sellable catalogue. Scope `products`. Module Products. Read only. The cost price and the preferred supplier are internal and are never returned.

GET /products

Shortcut parameters: `status` (`active`, `inactive`), `category`, `sub_category`, `item_type` (a stable key such as `finished_product`, `raw_material`, `bom_kit`).

Queryable fields: `id`, `ref`, `sku`, `name`, `description`, `category`, `sub_category`, `item_type`, `uom`, `list_price`, `qty_breaks`, `tax_percent`, `status`. Searchable: `ref`, `sku`, `name`, `description`, `category`, `sub_category`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/products?category=Motors"
```

```
{
  "data": [
    {
      "id": "a85c2e71-0d94-4f3b-b6a8-1e7d3c9f5a0a",
      "ref": "PRD-0014",
      "sku": "MOT-22-IE3",
      "name": "22 kW IE3 induction motor",
      "description": "415 V, 4 pole, B3 foot mount",
      "category": "Motors",
      "sub_category": "Induction",
      "item_type": "finished_product",
      "uom": "Nos",
      "list_price": 18500,
      "qty_breaks": [ { "from": 5, "rate": 17800 }, { "from": 10, "rate": null } ],
      "tax_percent": 5,
      "status": "active",
      "custom_data": null,
      "_links": { "self": "/api/v1/products/a85c2e71-0d94-4f3b-b6a8-1e7d3c9f5a0a" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-28T08:27:10.000Z" },
  "_links": { "self": "/api/v1/products", "next": null, "prev": null }
}
```

`qty_breaks` is a list of `{ from, rate }`. A `null` rate means that quantity is priced at the normal price.

GET /products/:id

The list row plus `created_at` and `updated_at`.

16. Project hours

One row for each worked session over the last 12 months, for project costing. Scope `projects`. Module Workforce projects. Read only.

GET /project-hours

Shortcut parameters: `project_id`, `account_id`.

Queryable fields: `date`, `project.id`, `project.ref`, `project.name`, `project.level`, `project.status`, `top_project.id`, `top_project.name`, `account.id`, `account.name`, `employee.id`, `employee.name`, `employment_type`, `minutes`, `hours`, `break_minutes`, `bill_rate`, `billable_amount`, `invoiced`, `assigned`. Searchable: `project.ref`, `project.name`, `top_project.name`, `account.name`, `employee.name`. The `employee.*` fields exist only for a key that also names `employees`. A session that is not assigned to a project shows `project.name = "Unassigned"`.

Hours per project for October, totals only:

```
curl -s -G -H "Authorization: Bearer $KEY" "$BASE/project-hours" \
  --data-urlencode "filter=date:gte:2026-10-01;date:lte:2026-10-31" \
  --data-urlencode "group_by=project.name" \
  --data-urlencode "aggregate=sum:hours,sum:billable_amount" \
  --data-urlencode "count=only"
```

```
{
  "data": [],
  "meta": {
    "count": 0, "total": 118, "page": 1, "limit": 50, "has_more": false,
    "aggregates": { "sum_hours": 912.5, "sum_billable_amount": 54750 },
    "groups": [
      { "key": "Plant retrofit", "count": 84, "sum_hours": 655, "sum_billable_amount": 39300 },
      { "key": "Unassigned", "count": 34, "sum_hours": 257.5, "sum_billable_amount": 15450 }
    ],
    "generated_at": "2026-10-28T08:28:00.000Z"
  },
  "_links": { "self": "/api/v1/project-hours", "next": null, "prev": null }
}
```

One row, when you leave out `count=only`:

```
{
  "id": "923e7a4b-5d8a-4b26-99f3-2e4b8d6c0f16:2026-10-14T04:00:00.000Z",
  "date": "2026-10-14",
  "project": { "id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c", "ref": "PRJ-0003", "name": "Plant retrofit",
"level": 0, "status": "active" },
  "top_project": { "id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c", "name": "Plant retrofit" },
  "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
  "employee": { "id": "923e7a4b-5d8a-4b26-99f3-2e4b8d6c0f16", "name": "Ravi Menon", "code": "EMP-0042" },
  "employment_type": "full_time",
  "minutes": 450,
  "hours": 7.5,
  "break_minutes": 30,
  "bill_rate": 60,
  "billable_amount": 450,
  "invoiced": false,
  "assigned": true
}
```


Rows have no `_links`. Treat the row `id` as opaque. Cost is never a column.

---

17. Projects

Workforce projects and their sub-projects. Scope `projects` (write for the invoice call). Module Workforce projects.

GET /projects

Shortcut parameters: `status` (`planned`, `active`, `on_hold`, `completed`, `cancelled`), `level` (`0` for a project, `1` to `3` for a sub-project), `parent_id`, `account_id`.

Queryable fields: `id`, `ref`, `name`, `code`, `level`, `parent_id`, `account_id`, `status`, `start_date`, `end_date`, `budget_hours`, `bill_rate`. Searchable: `ref`, `name`, `code`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/projects?level=0&status=active"
```

```
{
  "data": [
    {
      "id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c",
      "ref": "PRJ-0003",
      "name": "Plant retrofit",
      "code": "PO-77812",
      "parent_id": null,
      "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
      "status": "active",
      "start_date": "2026-09-01",
      "end_date": "2026-12-31",
      "budget_hours": 1200,
      "bill_rate": 60,
      "custom_data": null,
      "created_at": "2026-08-25T06:00:00.000000+00:00",
      "updated_at": "2026-10-01T06:00:00.000000+00:00",
      "level": 0,
      "_links": {
        "self": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c",
        "hours": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c/hours?from=YYYY-MM-DD&to=YYYY-MM-DD"
      }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:29:00.000Z" },
  "_links": { "self": "/api/v1/projects", "next": null, "prev": null }
}
```

GET /projects/:id


```
{
  "data": {
    "id": "f2d8b6c3-7a41-4e95-8b0d-4c9a1e7f2b0d",
    "ref": "PRJ-0003.1",
    "name": "Electrical works",
    "code": null,
    "parent_id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c",
    "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
    "status": "active",
    "start_date": "2026-09-01",
    "end_date": null,
    "budget_hours": 400,
    "bill_rate": null,
    "custom_data": null,
    "created_at": "2026-08-25T06:05:00.000000+00:00",
    "updated_at": "2026-08-25T06:05:00.000000+00:00",
    "level": 1,
    "parent": { "id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c", "ref": "PRJ-0003", "name": "Plant retrofit",
"_links": { "self": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c" } },
    "sub_projects": [],
    "links": { "site_ids": [], "employee_ids": ["923e7a4b-5d8a-4b26-99f3-2e4b8d6c0f16"], "shift_ids": [] },
    "_links": {
      "self": "/api/v1/projects/f2d8b6c3-7a41-4e95-8b0d-4c9a1e7f2b0d",
      "hours": "/api/v1/projects/f2d8b6c3-7a41-4e95-8b0d-4c9a1e7f2b0d/hours?from=YYYY-MM-DD&to=YYYY-MM-DD",
      "invoices": "/api/v1/projects/f2d8b6c3-7a41-4e95-8b0d-4c9a1e7f2b0d/invoices",
      "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
    }
  },
  "_links": { "self": "/api/v1/projects/f2d8b6c3-7a41-4e95-8b0d-4c9a1e7f2b0d" }
}
```


`sub_projects` rows are `{ id, ref, name, _links }`.

GET /projects/:id/hours

Worked hours for the project and everything under it, for a period. Totals only, with no per-person detail. Query: `from` and `to`, both `YYYY-MM-DD` and both required. `to` must be on or after `from`, and the window can be at most 366 days.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c/hours?
from=2026-10-01&to=2026-10-31"
```

```
{
  "data": {
    "project_id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c",
    "from": "2026-10-01",
    "to": "2026-10-31",
    "deduct_breaks": true,
    "total_minutes": 39300,
    "total_hours": 655,
    "breakdown": [
      {
        "project_id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c",
        "name": "Plant retrofit",
        "ref": "PRJ-0003",
        "status": "active",
        "parent_id": null,
        "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
        "depth": 0,
        "own_minutes": 12000,
        "total_minutes": 39300,
        "employees": 5,
        "employees_total": 11,
        "sessions": 40,
        "_links": { "self": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c" }
      }
    ]
  },
  "_links": {
    "self": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c/hours?from=2026-10-01&to=2026-10-31",
    "project": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c"
  }
}
```


Errors: `400 {"error":"from and to (YYYY-MM-DD) are required"}`, `{"error":"to must be on or after from"}`, `{"error":"Pick a window of 366 days or fewer"}`; `404`.

GET /projects/:id/invoices

Every invoice raised from this project's hours, newest first. That includes invoices raised for its sub-projects, or for a parent project that covered it.

```
{
  "data": [
    {
      "id": "d4f6a8c0-5b7d-4293-8e4f-6a8c0e2b4d55",
      "project_id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c",
      "project_name": "Plant retrofit",
      "invoice_id": "61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709",
      "invoice_ref": "INV-2026-0113",
      "invoice_status": "draft",
      "period_from": "2026-09-01",
      "period_to": "2026-09-30",
      "granularity": "project",
      "minutes": 37200,
      "amount": 37200,
      "created_at": "2026-10-01T00:30:00.000000+00:00",
      "_links": {
        "invoice": "/api/v1/invoices/61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709",
        "project": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c"
      }
    }
  ],
  "_links": { "self": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c/invoices", "project":
"/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c" }
}
```


Needs the Workforce projects and Invoices modules (`404` otherwise). If a database update for project billing is still waiting in your workspace, the answer is `{"data":[],"pending_migration":true}`. Ask your BPMSquare contact.

POST /projects/:id/invoices

Raises a draft invoice for a period from the project's hours. The rate comes from a ladder: the project's rate, then the parent's rate, then the rate for the employment type, then the workspace default. Needs a key with **write** on `projects`.

Body field	Type	Notes
`from`, `to`	`YYYY-MM-DD`, required	At most 366 days apart.
`granularity`	`project` (default) or `sub_project`	One line for the project, or one line per sub-project.
`dry_run`	boolean	`true` returns the preview and writes nothing.
`top_up`	boolean	Bill only the hours added since the earlier invoice for the same period.

A dry run:

```
curl -s -X POST -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" \
  "$BASE/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c/invoices" \
  -d '{ "from": "2026-10-01", "to": "2026-10-31", "dry_run": true }'
```


```
{
  "data": {
    "project": { "id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c", "ref": "PRJ-0003", "name": "Plant retrofit",
"level": 0, "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "account_name": "Example Trading LLC" },
    "from": "2026-10-01",
    "to": "2026-10-31",
    "granularity": "project",
    "lines": [
      { "group_id": "e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c", "group_label": "Plant retrofit", "description":
"Plant retrofit", "rate": 60, "source": "project", "employment_type": null, "minutes": 39300, "hours": 655,
"amount": 39300 }
    ],
    "minutes": 39300,
    "hours": 655,
    "subtotal": 39300,
    "tax": { "label": "VAT", "rate": 5, "inclusive": false, "amount": 1965 },
    "total": 41265,
    "due_date": "2026-11-27",
    "unassigned_minutes": 0,
    "conflicts": [],
    "top_up": null,
    "billed": [],
    "blockers": []
  },
  "_links": { "self": "/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c/invoices" }
}
```


`source` is `project`, `parent`, `employment_type` or `default`. `blockers` lists, in words, why an invoice cannot be raised now: no account, no hours, no rate, or the period is already billed. Cost and margin are removed from the preview.

Without `dry_run` the call creates the draft and answers `201`:


```
{
  "data": { "id": "61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709", "ref": "INV-2026-0114", "total": 41265, "status":
"draft" },
  "_links": { "invoice": "/api/v1/invoices/61f0a3d9-4b7c-4e25-8a16-c3d5e9b1f709", "project":
"/api/v1/projects/e9c4a7b2-1f58-4d36-a0e7-5b8d2c1f3a0c" }
}
```


Errors: `400` (`{"error":"Invalid JSON body"}` , the period text, `{"error":"granularity must be project or sub_project"}`), `403` (a read-only key), `404` , `409` (the first blocker, such as `Already invoiced on INV-… Bill the difference as a top-up instead.` , `Nothing to top up for that period.` or `Link this project to an account first.`), `503` (billing for projects is not ready in your workspace yet; ask your BPMSquare contact), `500` .

18. Purchase orders

Orders placed with your suppliers. Scope `purchase-orders`. Module Purchasing. Read only.

GET /purchase-orders

Shortcut parameters: `status` (`draft`, `sent`, `partially_received`, `received`, `cancelled`), `supplier_id`.

Queryable fields: `id`, `ref`, `status`, `supplier.id`, `supplier.name`, `quote_id`, `case_id`, `order_date`, `expected_date`, `total`, `created_at`. Searchable: `ref`, `supplier.name`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/purchase-orders?status=sent"
```


```
{
  "data": [
    {
      "id": "14a9e5c7-3b02-4d68-b7f1-9c6d0a2e8b0e",
      "ref": "PO-2026-0057",
      "status": "sent",
      "supplier": { "id": "2e6a4c18-8f03-4b97-a1e5-d9c7b3f0a508", "name": "Example Bearings FZE" },
      "quote_id": null,
      "case_id": "c41d8f06-9e2b-47a3-b5c1-3a7e0d9f6b06",
      "order_date": "2026-10-22",
      "expected_date": "2026-11-05",
      "total": 1540,
      "created_at": "2026-10-22T07:30:00.000000+00:00",
      "_links": { "self": "/api/v1/purchase-orders/14a9e5c7-3b02-4d68-b7f1-9c6d0a2e8b0e" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:30:10.000Z" },
  "_links": { "self": "/api/v1/purchase-orders", "next": null, "prev": null }
}
```

GET /purchase-orders/:id


```
{
  "data": {
    "id": "14a9e5c7-3b02-4d68-b7f1-9c6d0a2e8b0e",
    "ref": "PO-2026-0057",
    "status": "sent",
    "supplier": { "id": "2e6a4c18-8f03-4b97-a1e5-d9c7b3f0a508", "name": "Example Bearings FZE", "email":
"sales@example.com", "phone": "+971 6 555 0190" },
    "quote_id": null,
    "case_id": "c41d8f06-9e2b-47a3-b5c1-3a7e0d9f6b06",
    "order_date": "2026-10-22",
    "expected_date": "2026-11-05",
    "notes": null,
    "terms": null,
    "total": 1540,
    "custom_data": null,
    "created_at": "2026-10-22T07:30:00.000000+00:00",
    "lines": [
      {
        "id": "e5a7c9b1-6d8f-4304-9f5e-7a9c1e3b5d66",
        "inventory_item_id": "7d2b9e40-1c85-4a6f-93d8-e5a0b1c2d307",
        "description": "Deep groove ball bearing 6310-2RS",
        "uom": "Nos",
        "qty_ordered": 40,
        "qty_received": 0,
        "rate": 38.5,
        "amount": 1540
      }
    ],
    "_links": { "self": "/api/v1/purchase-orders/14a9e5c7-3b02-4d68-b7f1-9c6d0a2e8b0e" }
  },
  "_links": { "self": "/api/v1/purchase-orders/14a9e5c7-3b02-4d68-b7f1-9c6d0a2e8b0e" }
}
```


Errors: `404`.

---

19. Quotations

The only object you can write. A quotation is a header plus ordered lines. Lines can be grouped into additive groups or into mutually exclusive alternatives. Scope `quotations` (write for `POST`, `PATCH` and `DELETE`). Module Quotations. `GET /metadata/quotations` serves the live field reference.

GET /quotations

Shortcut parameters: `status` (a status your workspace uses; the defaults are `draft`, `sent`, `approved`, `rejected`), `account_id`.

Queryable fields: `id`, `ref`, `status`, `total`, `revision`, `outcome`, `loss_reason`, `created_at`, `quote_date`, `valid_until`, `line_count`, `account.id`, `account.name`, `invoice_count`, `invoiced_total`, `paid_total`, `balance_due`. Searchable: `ref`, `account.name`. The four invoice fields add up the quotation's own invoices that

are not cancelled, so you can follow a quotation through to cash.

```
curl -s -G -H "Authorization: Bearer $KEY" "$BASE/quotations" \
  --data-urlencode "filter=status:eq:draft;total:gte:10000" \
  --data-urlencode "sort=-total" --data-urlencode "aggregate=sum:total"
```


```
{
  "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": 50, "has_more": false, "aggregates": { "sum_total":
41050 }, "generated_at": "2026-10-28T08:31:00.000Z" },
  "_links": { "self": "/api/v1/quotations", "metadata": "/api/v1/metadata/quotations", "next": null, "prev":
null }
}
```


The `pdf` link is a page on your workspace address, not an API path.

GET /quotations/:id

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/quotations/b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03"
```

```
{
  "data": {
    "id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
    "ref": "QT-2026-0042",
    "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
    "contact_id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02",
    "entity_id": null,
    "type": "quotation",
    "status": "draft",
    "business_status": null,
    "outcome": "open",
    "name": "Supply of 22 kW motors and installation",
    "ref_no": null,
    "pr_no": null,
    "po_number": null,
    "po_amount": null,
    "quote_date": "2026-10-26",
    "valid_until": "2026-11-25",
    "inquiry_date": null,
    "submitted_at": null,
    "closed_at": null,
    "updated_at": "2026-10-26T06:12:31.118820+00:00",
    "notes": null,
    "terms": "Payment 30 days from invoice.",
    "scope_of_work": null,
    "discount_type": "pct",
    "discount_pct": 0,
    "discount_fixed": 0,
    "gst_rate": 5,
    "asset_ids": [],
    "selected_option_id": null,
    "custom_data": null,
    "meta": null,
    "total": 41050,
    "revision": 1,
    "superseded_by": null,
    "created_at": "2026-10-26T06:12:31.118820+00:00",
    "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
    "contact": { "id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02", "name": "Sara Khan" },
    "lines": [
      {
        "id": "0e4c6a91-3d27-4b85-9f10-a2c8d5e7b104",
        "sl_no": "1",
        "description": "Supply of 3-phase induction motor 22 kW",
        "uom": "Nos",
        "qty": 2,
        "rate": 18500,
        "discount_pct": 0,
        "amount": 37000,
        "group_id": null,
        "group_label": null,
        "group_type": null,
        "group_description": null,
        "category": "material",
        "deduction": 0,
        "inventory_item_id": null
      },
```

```
      {
        "id": "9a1f3b52-6c48-4d07-8e39-b1d2c4a6f205",
        "sl_no": "2",
        "description": "Installation and commissioning",
        "uom": "Lot",
        "qty": 1,
        "rate": 4500,
        "discount_pct": 10,
        "amount": 4050,
        "group_id": null,
        "group_label": null,
        "group_type": null,
        "group_description": null,
        "category": "labour",
        "deduction": 0,
        "inventory_item_id": null
      }
    ],
    "line_count": 2,
    "totals": { "subtotal": 41050, "discount": 0, "deductions": 0, "total": 41050 },
    "_links": {
      "self": "/api/v1/quotations/b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
      "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
      "pdf": "/quotations/b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03/print"
    }
  },
  "_links": { "self": "/api/v1/quotations/b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03", "metadata":
"/api/v1/metadata/quotations" }
}
```


Lines come in natural order of `sl_no` (1, 2, 10). `total` is before tax. `gst_rate` is shown for display and is not added to `total`.

POST /quotations

Creates a quotation with its lines and answers `201`. Only `account_id` is required. The reference (`ref`) is made by your workspace's quotation numbering. `GET /metadata/quotations` is the live and authoritative list of fields.

Field	Type	Notes
`account_id`	uuid, required	Must belong to your workspace, else `404`.
`contact_id`	uuid	Must belong to your workspace.
`entity_id`	string (64)	The issuing legal entity. Leave out for the default.
`type`	`quotation`, `technical`, `budgetary`, `supply`, `repair`	Defaults to `quotation`. `technical` hides pricing on the document.
`status`	string (40)	A status from your workspace's quotation statuses. Defaults to `draft`. A closed status needs `outcome` set to won, lost or dropped.

Field	Type	Notes
`business_status`	`pending`, `po_received`	
`outcome`	`open`, `won`, `lost`, `dropped`	Defaults to `open`.
`name`	string (300)	The title.
`ref_no`, `pr_no`, `po_number`	string (100)	Your own references.
`po_amount`	number, 0 or more	
`quote_date`, `valid_until`, `inquiry_date`	`YYYY-MM-DD`	
`submitted_at`, `closed_at`	ISO 8601	Stored as you give them. Never stamped for you on create.
`notes`, `terms`	text	
`scope_of_work`	rich text	Cleaned on the server.
`discount_type`	`pct`, `fixed`	Defaults to `pct`.
`discount_pct`	number, 0 to 100	
`discount_fixed`	number, 0 or more	Capped at the subtotal.
`gst_rate`	number, 0 to 100	`null` means no tax row. It is not part of `total`.
`asset_ids`	list of uuid	Every id must be in your workspace.
`selected_option_id`	string (64)	The `group_id` of the alternative group that counts.
`custom_data`	object	Keyed by your custom field keys.
`meta`	object	Free form. Stored untouched.
`lines`	list, at most 200	See below.

Line fields:

Field	Notes
`description`	Required.
`sl_no`	String (40). Your serial label.
`uom`	String (20).

Field	Notes
`qty`	0 or more. Defaults to 1.
`rate`	0 or more. Defaults to 0.
`discount_pct`	0 to 100.
`group_id`, `group_label`, `group_description`	Group a set of lines.
`group_type`	`additive` (default) or `alternative`.
`category`	`labour`, `material`, `testing`, `transport`.
`deduction`	0 or more. Honoured only on `material` lines.
`inventory_item_id`	uuid in your workspace. For reporting only.

These fields are read-only. Sending one is a `422`: `id`, `ref`, `total`, `updated_at`, `revision`, `superseded_by`, `created_at`, and on lines `id` and `amount`. An unknown field is rejected, not ignored. A number may be sent as a string that looks like a number.

How money is worked out:

- `amount = qty × rate × (1 − discount_pct / 100)`.
- `subtotal` is the sum of the effective lines. A line in an alternative group other than `selected_option_id` is left out.
- `total = subtotal − header discount − material deductions`.
- A percentage discount is rounded to a whole unit.

```
curl -s -X POST -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" "$BASE/quotations" -d '{
  "account_id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
  "contact_id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02",
  "name": "Supply of 22 kW motors and installation",
  "quote_date": "2026-10-26",
  "valid_until": "2026-11-25",
  "gst_rate": 5,
  "terms": "Payment 30 days from invoice.",
  "lines": [
    { "sl_no": "1", "description": "Supply of 3-phase induction motor 22 kW", "uom": "Nos", "qty": 2,
"rate": 18500, "category": "material" },
    { "sl_no": "2", "description": "Installation and commissioning", "uom": "Lot", "qty": 1, "rate": 4500,
"discount_pct": 10, "category": "labour" }
  ]
}'
```

The `201` answer is `{ "data": { … } }` with the same quotation object as `GET /quotations/:id` (without `account` and `contact`), plus `lines`, `line_count`, `totals` and `_links`. The new quotation has `status: "draft"`, `outcome: "open"`, `revision: 1` and `total: 41050`. The creation is recorded in Change history with the actor `api:v1`.

A `422` looks like this:

```
{
  "error": "Validation failed",
  "message": "2 fields rejected. See \"details\". Full field reference: GET /api/v1/metadata/quotations",
  "details": [
    { "field": "total", "message": "Read-only: this value is calculated by the server and cannot be set directly." },
    { "field": "lines[0].qty", "message": "must be >= 0" }
  ]
}
```

Other `422` messages: `Unknown status "x". Allowed values are tenant-configured …`, `Set an outcome (won, lost, or dropped) when creating a quotation directly in a closed status.` and `A rejected quotation can't be marked won.`

Other errors: `400` malformed JSON; `403` a read-only or wrongly scoped key; `404` with `{"error":"Account not found in this tenant"}` (also `Contact not found in this tenant`, `One or more assets not found in this tenant`, `One or more inventory items not found in this tenant`); `500`.

PATCH /quotations/:id

Send only the fields you want to change. They are the same fields as `POST`, and none is required.

- `lines`, when present, **replaces all lines**. Send the complete set. Leave `lines` out to keep them as they are.
- The total is always worked out again.
- Moving off a closed status sets `outcome` back to `open`, unless you send one.
- `submitted_at` is stamped the first time the status leaves its starting value. `closed_at` is stamped when `outcome` leaves `open`. A value you send always wins.
- A superseded version (one with `superseded_by` set) is read-only. Edit the latest version.

```
curl -s -X PATCH -H "Authorization: Bearer $KEY" -H "Content-Type: application/json" \
  "$BASE/quotations/b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03" -d '{ "status": "sent", "valid_until": "2026-12-15" }'
```

The `200` answer is `{ "data": { … } }`, the updated quotation as in `POST`, with `status: "sent"`, `valid_until: "2026-12-15"` and `submitted_at` stamped.

Errors: `404 {"error":"Quotation not found"}`; `409 {"error":"This version has been superseded and is read-only -- edit the latest version instead."}`; `422` (also `Unknown status`, the closed-status rules, the rejected-won rule, and `Set an outcome (won, lost, or dropped) before closing this quotation -- or before clearing the outcome on one that's already closed.`); `500`.

Replacing lines is not atomic with the header update. If the call fails part-way, read the quotation back and repeat it.

DELETE /quotations/:id

```
curl -s -X DELETE -H "Authorization: Bearer $KEY" "$BASE/quotations/b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03"
```

```
{ "data": { "id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03", "ref": "QT-2026-0042", "deleted": true } }
```

The deletion is kept in your workspace's deleted-records log and in Change history. Errors: `404`.

---

20. RFQs

Requests for quotation sent to suppliers. Scope `rfqs`, which you must name, because supplier prices are commercially sensitive. Module RFQ. Read only.

GET /rfqs

Shortcut parameters: `status` (`draft`, `sent`, `partially_replied`, `replied`, `awarded`, `closed`, `cancelled`), `origin_type` (`standard_quote`, `quote`, `opportunity`, `case`, `inspection_report`, `work_order`, `fence_project`, `pricing`, `manual`), `origin_id`.

Queryable fields: `id`, `ref`, `title`, `status`, `origin_type`, `origin_id`, `origin_label`, `respond_by`, `required_by`, `created_at`, `updated_at`. Searchable: `ref`, `title`, `origin_label`. Supplier prices are on the detail endpoint only.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/rfqs?status=sent"
```

```
{
  "data": [
    {
      "id": "28b3d7f9-5c16-4a40-9e82-0d1f6a3c7b0f",
      "ref": "RFQ-0012",
      "title": "Bearings for case CASE-0031",
      "status": "sent",
      "origin_type": "case",
      "origin_id": "c41d8f06-9e2b-47a3-b5c1-3a7e0d9f6b06",
      "origin_label": "CASE-0031",
      "respond_by": "2026-11-01",
      "required_by": "2026-11-10",
      "delivery_terms": "DAP Dubai",
      "currency": "AED",
      "custom_data": null,
      "created_at": "2026-10-21T08:00:00.000000+00:00",
      "updated_at": "2026-10-22T07:00:00.000000+00:00",
      "_links": { "self": "/api/v1/rfqs/28b3d7f9-5c16-4a40-9e82-0d1f6a3c7b0f" }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:32:00.000Z" },
  "_links": { "self": "/api/v1/rfqs", "next": null, "prev": null }
}
```


GET /rfqs/:id

The RFQ with its lines, its suppliers and their quoted prices. Never returned: the reply-link token, internal notes and message bodies.

```
{
  "data": {
    "id": "28b3d7f9-5c16-4a40-9e82-0d1f6a3c7b0f",
    "ref": "RFQ-0012",
    "title": "Bearings for case CASE-0031",
    "status": "sent",
    "origin_type": "case",
    "origin_id": "c41d8f06-9e2b-47a3-b5c1-3a7e0d9f6b06",
    "origin_label": "CASE-0031",
    "requested_by": "11111111-2222-4333-8444-555555555555",
    "buyer_id": null,
    "respond_by": "2026-11-01",
    "required_by": "2026-11-10",
    "delivery_terms": "DAP Dubai",
    "currency": "AED",
    "notes_to_supplier": null,
    "custom_data": null,
    "created_at": "2026-10-21T08:00:00.000000+00:00",
    "updated_at": "2026-10-22T07:00:00.000000+00:00",
    "lines": [
      {
        "id": "f6b8d0c2-7e90-4415-8a6f-8b0d2f4c6e77",
        "sl_no": 1,
        "product_id": null,
        "description": "Deep groove ball bearing 6310-2RS",
        "spec": "50 x 110 x 27 mm",
        "uom": "Nos",
        "qty": 40,
        "required_by": "2026-11-10",
        "status": "open",
        "awarded_response_id": null,
        "awarded_unit_cost": null
      }
    ],
    "suppliers": [
      {
        "id": "07c9e1d3-8fa1-4526-9b70-9c1e3a5d7f88",
        "supplier_id": "2e6a4c18-8f03-4b97-a1e5-d9c7b3f0a508",
        "supplier_name": "Example Bearings FZE",
        "status": "replied",
        "sent_at": "2026-10-21T09:00:00+00:00",
        "viewed_at": "2026-10-21T10:00:00+00:00",
        "replied_at": "2026-10-22T06:30:00+00:00",
        "supplier_quote_ref": "EB-Q-2210",
        "currency": "AED",
        "incoterm": "DAP",
        "freight_amount": 120,
        "payment_terms": "30 days",
        "valid_until": "2026-11-30"
      }
    ],
    "responses": [
      {
        "id": "18d0f2e4-90b2-4637-8c81-0d2f4b6e8a99",
        "rfq_supplier_id": "07c9e1d3-8fa1-4526-9b70-9c1e3a5d7f88",
        "rfq_line_id": "f6b8d0c2-7e90-4415-8a6f-8b0d2f4c6e77",
        "status": "quoted",
```

```
        "unit_price": 38.5,
        "currency": "AED",
        "fx_rate": 1,
        "lead_time_days": 7,
        "moq": 10,
        "offered_description": null,
        "valid_until": "2026-11-30",
        "source": "link",
        "is_awarded": false,
        "submitted_at": "2026-10-22T06:30:00+00:00"
      }
    ],
    "_links": { "self": "/api/v1/rfqs/28b3d7f9-5c16-4a40-9e82-0d1f6a3c7b0f" }
  },
  "_links": { "self": "/api/v1/rfqs/28b3d7f9-5c16-4a40-9e82-0d1f6a3c7b0f" }
}
```


Value lists:

Field	Values
`lines[].status`	`open`, `awarded`, `no_award`, `cancelled`
`suppliers[].status`	`pending`, `sent`, `viewed`, `replied`, `declined`, `expired`, `cancelled`
`responses[].status`	`quoted`, `no_quote`, `alternative`
`responses[].source`	`link`, `buyer`, `upload`, `email`

Errors: `404`.

21. Sales orders

Scope `sales_orders` (included in "All objects"). Module Sales Orders. Read only. Never returned: the creator and owner ids, the cancellation note, contact details of a party, the order type's template, attachments, and any cost or margin.

GET /sales-orders

Shortcut parameters: `status` (`draft`, `confirmed`, `cancelled` for orders made in BPMSquare; an order that came from your ERP carries the ERP's own word in `status_label`), `account_id`, `source_id` (the quote or deal it was made from).

Queryable fields: `id`, `ref`, `status`, `status_label`, `spine`, `origin`, `type.id`, `type.name`, `pricing_mode`, `currency`, `customer_po`, `source_type`, `source_id`, `opportunity_id`, `subtotal`, `discount_total`, `tax_total`, `shipping_amount`, `total`, `line_count`, `credit_check`, `price_check`, `stock_check`, `invoice_count`, `invoiced_total`, `paid_total`, `requested_date`, `expected_date`, `confirmed_at`, `sent_at`, `account.id`, `account.name`, `contact_id`, `created_at`, `updated_at`. Searchable: `ref`, `customer_po`, `account.name`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/sales-orders?status=confirmed"
```


```
{
  "data": [
    {
      "id": "3c7e1a85-9d24-4b60-a3f7-6e8b2d0c4a10",
      "ref": "SO-0004",
      "type": { "id": "4d8f2b96-0e35-4c71-b4a8-7f9c3e1d5a11", "name": "Standard order" },
      "origin": "bpmsquare",
      "status": "confirmed",
      "status_label": null,
      "spine": "open",
      "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
      "contact_id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02",
      "bill_to": { "name": "Example Trading LLC", "address": "Plot 14, Industrial Area 3, Dubai" },
      "ship_to": { "name": "Example Trading LLC, Jebel Ali store", "address": "Warehouse 6, Jebel Ali" },
      "customer_po": "PO-77812",
      "source_type": "standard_quote",
      "source_id": "5e9a3c07-1f46-4d82-95b9-8a0d4f2e6b12",
      "opportunity_id": null,
      "currency": "AED",
      "pricing_mode": "internal",
      "subtotal": 41050,
      "discount_total": 0,
      "tax_total": 2052.5,
      "shipping_amount": 0,
      "total": 43102.5,
      "requested_date": "2026-11-20",
      "expected_date": "2026-11-24",
      "confirmed_at": "2026-10-27T05:00:00+00:00",
      "sent_at": null,
      "line_count": 2,
      "credit_check": "passed",
      "price_check": null,
      "stock_check": "warning",
      "invoice_count": 0,
      "invoiced_total": 0,
      "paid_total": 0,
      "custom_data": null,
      "created_at": "2026-10-27T04:40:00.000000+00:00",
      "updated_at": "2026-10-27T05:00:00.000000+00:00",
      "_links": {
        "self": "/api/v1/sales-orders/3c7e1a85-9d24-4b60-a3f7-6e8b2d0c4a10",
        "pdf": "/sales-orders/3c7e1a85-9d24-4b60-a3f7-6e8b2d0c4a10/print",
        "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
      }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:33:00.000Z" },
  "_links": { "self": "/api/v1/sales-orders", "next": null, "prev": null }
}
```


Value lists:

Field	Values
`spine`	`open`, `in_progress`, `completed`, `cancelled`
`origin`	`bpmsquare`, `erp`
`pricing_mode`	`internal`, `external`
`source_type`	`standard_quote`, `quotation`, `opportunity`, `lead`, `email`, `order`, `direct`, `erp`
`credit_check`, `price_check`, `stock_check`	`passed`, `warning`, `failed`, `not_run`, or `null`

GET /sales-orders/:id

The list object plus `notes`, `terms`, `lines`, `erp_documents`, `erp_checks` and `erp_transfers`.

```
{
  "data": {
    "id": "3c7e1a85-9d24-4b60-a3f7-6e8b2d0c4a10",
    "ref": "SO-0004",
    "status": "confirmed",
    "total": 43102.5,
    "line_count": 2,
    "notes": "Deliver to the Jebel Ali store.",
    "terms": "Payment 30 days from invoice.",
    "lines": [
      {
        "id": "29e1a3f5-a1c3-4748-9d92-1e3a5c7f9b00",
        "line_no": 1,
        "product_id": "a85c2e71-0d94-4f3b-b6a8-1e7d3c9f5a0a",
        "description": "22 kW IE3 induction motor",
        "printed_text": "415 V, 4 pole",
        "qty": 2,
        "uom": "Nos",
        "unit_price": 18500,
        "discount_pct": 0,
        "tax_pct": 5,
        "line_total": 37000,
        "requested_date": "2026-11-20",
        "confirmed_qty": 2,
        "confirmed_date": "2026-11-24",
        "availability_source": "master",
        "availability_at": "2026-10-27T05:01:00+00:00"
      }
    ],
    "erp_documents": [
      { "kind": "invoice", "number": "90001234", "date": "2026-11-25", "due_date": "2026-12-25", "amount":
43102.5, "currency": "AED", "status": "Open", "status_spine": "open", "erp_id": "0090001234", "fetched_at":
"2026-11-26T01:00:00+00:00" }
    ],
    "erp_checks": [
      { "kind": "credit", "status": "passed", "message": "Within credit limit.", "checked_at": "2026-10-
27T05:00:30+00:00", "stale": false }
    ],
    "erp_transfers": [
      { "attempted_at": "2026-10-27T05:02:00+00:00", "result": "sent", "erp_number": "0000451122", "step":
null, "message": null }
    ],
    "_links": { "self": "/api/v1/sales-orders/3c7e1a85-9d24-4b60-a3f7-6e8b2d0c4a10" }
  },
  "_links": { "self": "/api/v1/sales-orders/3c7e1a85-9d24-4b60-a3f7-6e8b2d0c4a10" }
}
```


All the header fields of the list row are present too (shortened here). Value lists: `availability_source` is `erp` or `master`. `erp_documents[].kind` is `delivery` or `invoice`. `status_spine` is `open`, `delivered`, `paid` or `cancelled`. `erp_checks[].kind` is `credit`, `price` or `stock`. `erp_transfers[].result` is `sent`, `failed` or `already_sent`. Errors: `404`.

22. Standard quotes

Scope `standard_quotes` (included in "All objects"). Module Standard Quotes, which is separate from Quotations. Read only. Never returned: line margin checks, the supplier-request link, the template, authorship, print options and contact details.

GET /standard-quotes

Shortcut parameters: `status` (`draft`, `sent`, `accepted`, `rejected`, `expired`), `account_id`, `opportunity_id`.

Queryable fields: `id`, `ref`, `status`, `inquiry_date`, `valid_until`, `sent_at`, `closed_at`, `subtotal`, `header_discount_pct`, `tax_pct`, `shipping_amount`, `total`, `line_count`, `account.id`, `account.name`, `contact_id`, `opportunity_id`, `created_at`, `updated_at`. Searchable: `ref`, `account.name`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/standard-quotes?status=sent&sort=-total"
```


```
{
  "data": [
    {
      "id": "5e9a3c07-1f46-4d82-95b9-8a0d4f2e6b12",
      "ref": "SQ-0031",
      "status": "sent",
      "inquiry_date": "2026-10-20",
      "valid_until": "2026-11-20",
      "sent_at": "2026-10-24T08:00:00+00:00",
      "closed_at": null,
      "subtotal": 41050,
      "header_discount_pct": 0,
      "tax_pct": 5,
      "shipping_amount": 0,
      "total": 43102.5,
      "account": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC" },
      "contact_id": "5c8e7d10-2b9a-4f63-a1d4-8e0b3c7f9a02",
      "opportunity_id": "d3b7f1a4-62e8-45c9-9d0b-7a2e4c6f8b0b",
      "line_count": 2,
      "custom_data": null,
      "created_at": "2026-10-20T07:00:00.000000+00:00",
      "updated_at": "2026-10-24T08:00:00.000000+00:00",
      "_links": {
        "self": "/api/v1/standard-quotes/5e9a3c07-1f46-4d82-95b9-8a0d4f2e6b12",
        "pdf": "/standard-quotes/5e9a3c07-1f46-4d82-95b9-8a0d4f2e6b12/print",
        "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
      }
    }
  ],
  "meta": { "count": 1, "total": 1, "page": 1, "limit": 50, "has_more": false, "generated_at": "2026-10-
28T08:34:00.000Z" },
  "_links": { "self": "/api/v1/standard-quotes", "next": null, "prev": null }
}
```

GET /standard-quotes/:id

The list object plus `intro_text`, `notes`, `terms` and `lines`.


```
{
  "data": {
    "id": "5e9a3c07-1f46-4d82-95b9-8a0d4f2e6b12",
    "ref": "SQ-0031",
    "status": "sent",
    "total": 43102.5,
    "line_count": 2,
    "intro_text": "Thank you for your enquiry.",
    "notes": null,
    "terms": "Valid for 30 days.",
    "lines": [
      {
        "id": "3af2b4c6-b2d4-4859-8ea3-2f4b6d8a0c11",
        "sl_no": "1",
        "description": "22 kW IE3 induction motor",
        "uom": "Nos",
        "qty": 2,
        "rate": 18500,
        "discount_pct": 0,
        "amount": 37000,
        "product_id": "a85c2e71-0d94-4f3b-b6a8-1e7d3c9f5a0a",
        "group_id": null,
        "group_label": null,
        "group_type": null,
        "break_of": null,
        "break_qty": null,
        "is_selected": true,
        "show_on_pdf": true
      }
    ],
    "_links": {
      "self": "/api/v1/standard-quotes/5e9a3c07-1f46-4d82-95b9-8a0d4f2e6b12",
      "pdf": "/standard-quotes/5e9a3c07-1f46-4d82-95b9-8a0d4f2e6b12/print",
      "account": "/api/v1/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01"
    }
  },
  "_links": { "self": "/api/v1/standard-quotes/5e9a3c07-1f46-4d82-95b9-8a0d4f2e6b12" }
}
```


The other header fields of the list row are present too (shortened here). Errors: `404`.

---

23. API index, metadata and OpenAPI

These endpoints answer for any key with read access. A write-only key gets `401`. They are not held back by modules.

GET /

The API describes itself: name, version `1.0`, how to authenticate, `start_here`, the `endpoints` map, `conventions`, `query`, `access_control`, `webhooks`, `ask` and `coming_soon`.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE"
```

```
{
  "name": "BPMSquare REST API",
  "version": "1.0",
  "generated_at": "2026-10-28T08:35:00.000Z",
  "authentication": "Bearer token …",
  "start_here": {
    "openapi": "GET /api/v1/openapi.json …",
    "metadata": "GET /api/v1/metadata …",
    "quotation_fields": "GET /api/v1/metadata/quotations …",
    "changes": "GET /api/v1/changes?since=<cursor> …"
  },
  "endpoints": { "GET /api/v1/accounts": "List all accounts", "…": "…" },
  "conventions": { "errors": "…", "totals": "…", "children": "…", "unknown_fields": "…" }
}
```

The answer is shortened here. The index lists most endpoints but not all. This chapter is the complete list.

GET /metadata

Lists the entities that have field-level descriptions. Only `quotations` has one today.

```
{
  "description": "Machine-readable description of every entity the v1 API exposes for writing: …",
  "entities": [
    {
      "key": "quotations",
      "entity": "quotation",
      "description": "A customer quotation: a header carrying commercial terms plus an ordered set of line
items, …",
      "endpoint": "/api/v1/quotations",
      "field_count": 35,
      "children": ["quote_line"],
      "_links": { "metadata": "/api/v1/metadata/quotations", "collection": "/api/v1/quotations" }
    }
  ],
  "documented_count": 1,
  "not_yet_documented": { "message": "These endpoints are read-only and have no field-level metadata yet.
…", "endpoints": ["/api/v1/accounts", "/api/v1/cases", "…"] },
  "_links": { "self": "/api/v1/metadata", "api_index": "/api/v1" }
}
```

`field_count` is the number of quotation fields. It can grow, so do not hard-code it.

GET /metadata/:entity

`GET /metadata/quotations` returns every quotation field with `field`, `type`, `label`, `description`, `required_on_create` and `writable`. Where they apply it also gives `computed`, `read_only`, `allowed_values`, `min`, `max`, `max_length`, `default`, `relation` and `example`. The same is given for the `lines` child (with `max_items: 200`), and for `relations`. Any other entity answers `404 {"error":"Unknown entity","message":"No metadata for \"x\".","available":["quotations"]}`.


```
{
  "data": {
    "entity": "quotation",
    "plural": "quotations",
    "table": "quotes",
    "endpoint": "/api/v1/quotations",
    "fields": [
      { "field": "account_id", "type": "uuid", "label": "Account", "description": "The customer this
quotation is for. …", "required_on_create": true, "writable": true, "relation": { "entity": "account",
"endpoint": "/api/v1/accounts" } },
      { "field": "total", "type": "number", "label": "Total", "description": "subtotal - header discount -
material deductions …", "required_on_create": false, "writable": false, "computed": true }
    ],
    "children": [ { "key": "lines", "max_items": 200, "entity": "quote_line", "fields": [ { "field":
"description", "type": "text", "required_on_create": true, "writable": true } ] } ],
    "relations": [ { "field": "account_id", "entity": "account", "endpoint": "/api/v1/accounts" } ]
  },
  "usage": { "create": "POST /api/v1/quotations …", "read_one": "GET /api/v1/quotations/:id", "update":
"PATCH /api/v1/quotations/:id …", "delete": "DELETE /api/v1/quotations/:id", "notes": ["…"] },
  "_links": { "self": "/api/v1/metadata/quotations", "all_metadata": "/api/v1/metadata", "collection":
"/api/v1/quotations" }
}
```


The answer is shortened here.

GET /openapi.json

An OpenAPI 3.0.3 file for the API. Its server is `/api/v1` and it uses bearer security. Import it into Postman, Insomnia, an integration platform or a code generator. The quotation entity has a full schema with `POST`, `PATCH` and `DELETE`. Every other list endpoint gets a generic list path and detail path with the shared query parameters. `/changes`, `/price` and `/ask` are included.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/openapi.json" -o bpmsquare-openapi.json
```

The file is generated, and it does not match the API in a few places. If you generate a client from it, know these:

- It lists `GET /cases/{id}`. That endpoint does not exist.
- It does not list `/project-hours`, `/projects/{id}/hours`, `/projects/{id}/invoices` (`GET` and `POST`), `/abm-programs/{id}/targets`, `/metadata`, `/metadata/{entity}`, `/openapi.json` itself, or the index.
- Its shared query parameters leave out `group_period`, `group_limit`, `having` and `group_sort`. The API accepts them.
- Detail answers for objects other than quotations are described only as "OK", with no schema.

CHAPTER 3

Change feed and webhooks

> 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 need to keep another system in step with BPMSquare. There are two ways to learn that something changed. You can ask: that is the **change feed**. Or you can be told: that is a **webhook**. BPMSquare can also send one record to another system on demand, and an ERP can tell BPMSquare that an order changed.

Read Chapter 1 for keys and limits. The examples use made-up data: Example Trading LLC, a quotation `QT-2026-0042`, and a receiver at `hooks.example.com`.

Contents

1. Ask or be told
2. The change feed
3. Event webhooks
4. Verify the signature
5. Delivery, retries and idempotency
6. Call webhooks: one record when something happens
7. Push to ERP (the older way)
8. Telling BPMSquare that an ERP order changed
9. Check that it works

1. Ask or be told

	Change feed	Event webhook
How it works	You call `GET /changes` with a cursor.	BPMSquare calls your address with a signed batch.
Who sets the pace	You.	BPMSquare's delivery schedule.
Needs a public address	No.	Yes, an `https` address.

	Change feed	Event webhook
Access control	The key's scopes decide what you see.	An administrator chooses the objects. No key is involved.
Good for	A job that runs every few minutes, a first load, catching up after downtime.	Letting your system react without polling.
Missed changes	None. Keep your cursor and ask again.	None. A failed batch is sent again.

Both read the same record: the workspace's change history. An event in a webhook has the same shape as an item in the change feed.

Webhooks are delivered in batches on a schedule, not the moment a record changes. If you need to know within seconds, poll the change feed. Ask your BPMSquare contact how often deliveries run for your workspace.

A common pattern is to use both: a webhook as the prompt that "something changed", and the change feed as the source of truth you read after it.

---

2. The change feed

`GET /api/v1/changes` is an ordered log of every create, update and delete in your workspace. It is built for synchronisation: you keep a cursor, and each call returns what happened after it.

The cursor is based on the time and id of the last item. No item is skipped or repeated across pages, even when two changes happen in the same instant.

The key needs read access. The feed shows only the object types that the key could read through the list endpoints. A key limited to `quotations` sees quotation changes only.

Parameters

Parameter	Notes
`since`	The `cursor` from a previous answer. Leave it out to start at the beginning of the log. A cursor that was not returned by this endpoint is a `422`.
`object_type`	Only one object type, for example `quotes`. `403` if the key cannot read that type.
`limit`	Defaults to 100. Maximum 500.

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/changes?object_type=quotes&limit=100"
```

```
{
  "data": [
    {
      "id": "7a1b2c3d-4e5f-4a6b-8c7d-9e0f1a2b3c4d",
      "object_type": "quotes",
      "object_id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
      "object_label": "QT-2026-0042",
      "action": "create",
      "changes": [
        { "field": "account_id", "from": null, "to": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01" },
        { "field": "total", "from": null, "to": 41050 }
      ],
      "actor": "api:v1",
      "at": "2026-10-26T06:12:31.502114+00:00"
    },
    {
      "id": "8b2c3d4e-5f60-4b7c-9d8e-0f1a2b3c4d5e",
      "object_type": "quotes",
      "object_id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
      "object_label": "QT-2026-0042",
      "action": "update",
      "changes": [ { "field": "status", "from": "draft", "to": "sent" } ],
      "actor": "sara.admin@example.com",
      "at": "2026-10-27T09:41:12.318204+00:00"
    }
  ],
  "meta": { "count": 2, "has_more": false, "object_type": "quotes" },
  "cursor": "MjAyNi0xMC0yN1QwOTo0MToxMi4zMTgyMDQrMDA6MDB8OGIyYzNkNGUtNWY2MC00YjdjLTlkOGUtMGYxYTJiM2M0ZDVl",
  "_links": {
    "self": "/api/v1/changes",
    "next": "/api/v1/changes?
since=MjAyNi0xMC0yN1QwOTo0MToxMi4zMTgyMDQrMDA6MDB8OGIyYzNkNGUtNWY2MC00YjdjLTlkOGUtMGYxYTJiM2M0ZDVl&object_ty
pe=quotes&limit=100"
  }
}
```


What an item holds

Field	Meaning
`id`	The id of this change. Unique. Use it to remove duplicates.
`object_type`	The kind of record, such as `quotes` or `accounts`.
`object_id`	The record's id. Match on this, not on `object_label`.
`object_label`	The record's reference or name at the time.
`action`	`create`, `update`, `delete` or `reopen` (a quotation moved back into the pipeline).
`changes`	A list of `{ field, from, to }`.
`actor`	The person's email, or `api:v1` for a change made through the API.

Field	Meaning
`at`	When it happened.

Things to know about `changes`:

- Encrypted personal fields (account and contact phone, email, tax number, and staff phone and email) show `"[redacted]"` with `"redacted": true`. The change is visible. The value is not.
- Changes to lines appear as text, such as `Line 1`, `Line 2 added`, `Line 3 removed`. The order of lines is by position.
- The cost price of a product is left out of product changes.
- A `delete` tells you the record is gone. Fetch nothing. Remove it on your side.

Object types

These are the types the application logs: `accounts`, `contacts`, `quotes`, `cases`, `invoices`, `purchase_orders`, `inventory`, `products`, `employees`, `rfqs`, `estimates`, `opportunities`, `wfm_projects`, `standard_quotes`, `sales_orders`, `abm_programs`, `abm_target_accounts`, and more.

A key needs the matching scope to see a type:

Object type	Scope the key needs
`quotes`	`quotations`
`purchase_orders`	`purchase-orders`
`wfm_projects`	`projects`
`abm_target_accounts`	`abm_targets` (named)
`accounts`, `cases`, `invoices`, `inventory`, `products`, `employees` (named), `rfqs` (named), `estimates` (named), `opportunities`, `standard_quotes`, `sales_orders`, `abm_programs`	The scope of the same name

Types with no scope of their own, such as assets, suppliers, work orders and leads, are visible only to a key with **All objects**.

Reading the feed correctly

1. **First load.** Call without `since` and follow `cursor` until `meta.has_more` is false.
2. **Normal run.** Call with the `cursor` you stored last time.
3. **Save the cursor only after you have processed the page.** If your job fails half-way, you read the page again. That is safe if you remove duplicates on `id`.
4. **An empty page.** When nothing is returned, `cursor` repeats your `since` (or is `null` if you sent none). Keep it.

5. **A full last page.** `has_more` is true when the page is full (`count` equals `limit`). A final full page can say `true` and the next call can return nothing. That is normal.

Errors: `401`, `403` (an `object_type` outside the key's scope), `422 {"error":"Invalid cursor","message":"` since `must be a cursor returned by a previous /changes response."}`, `500`.

The change feed is a log, not a copy of the data. To get the current record, read it with its list or detail endpoint using `object_id`. Chapter 7 shows a full loop.

---

3. Event webhooks

An Event webhook sends the same items as the change feed, in signed batches, to an address you give.

Register one

You must be a workspace administrator.

1. Go to **Settings > Integrations > Webhooks**. If your workspace does not show that entry, go to **Settings > Integrations > Webhooks & push** instead. Both do the same job for events. The first is a newer, single list. Ask your BPMSquare contact which one your workspace has.
2. Press **New webhook**. Type a **Name**. Choose **Event** as the **Kind**.
3. Under **Changes sent**, tick **All objects**, or tick the objects you want: Quotes, Accounts, Contacts, Cases, Work orders, Invoices, Purchase orders, Inventory, Assets, Suppliers.
4. On the **Request** tab, type the **Address**. It must start with `https`. An address on a private or internal network is refused, both when you save it and each time BPMSquare calls it.
5. Press **Create**. Open the **Security** tab and copy the **Signing secret**. It looks like `whsec_` followed by 48 hexadecimal characters. It is shown once. Store it in your own secret store.

A new webhook sends only changes made after it was created. It does not replay history. To bring in history, read the change feed once.

Choosing **All objects** sends every change type recorded in your workspace's change history, including staff records. A webhook is not limited by a key's scopes. Tick only the objects the receiver should see.

In the older **Webhooks & push** page the steps are shorter: **+ New webhook**, a **Name**, an **Endpoint URL**, **All objects** or a list, then **Create webhook**. **Pause**, **Resume** and **Delete** are on each row.

What is sent

BPMSquare sends a `POST` with a JSON body. The call times out after 10 seconds.

```
Content-Type: application/json
X-BPMSquare-Signature: <hex HMAC-SHA256 of the raw body, key = your whsec_ secret>
X-BPMSquare-Delivery: <delivery_id>
```

```
{
  "delivery_id": "0f1e2d3c-4b5a-4968-8777-665544332211",
  "sent_at": "2026-10-28T20:30:04.118Z",
  "events": [
    {
      "id": "8b2c3d4e-5f60-4b7c-9d8e-0f1a2b3c4d5e",
      "object_type": "quotes",
      "object_id": "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
      "object_label": "QT-2026-0042",
      "action": "update",
      "changes": [ { "field": "status", "from": "draft", "to": "sent" } ],
      "actor": "sara.admin@example.com",
      "at": "2026-10-27T09:41:12.318204+00:00"
    }
  ]
}
```

Each item in `events` has the same shape as a change-feed item. One batch holds up to 100 events, oldest first.

The test message

Test (on the webhook's Test tab, or on the older page) sends a signed message marked as a test. The older page sends the normal batch shape with one event that has `object_type: "ping"` and `action: "ping"`. The newer page sends a flat body: `{ "event": "webhook.test", "delivery_id": …, "sent_at": …, "workspace": …, "ping": true }`. Accept it, check the signature, and do not treat it as a data change. A test on the newer page never changes the webhook's failure count.

---

4. Verify the signature

Every webhook call is signed. Check the signature before you trust the body.

- The signature is the hexadecimal HMAC-SHA256 of the **raw request body**.
- The key is the webhook's signing secret, exactly as shown, including the `whsec_` prefix.
- It is sent in the `X-BPMSquare-Signature` header.
- Compare in constant time.
- Compute it over the raw bytes. If you parse the JSON first and serialise it again, the bytes can differ and the check fails.

Node (Express)

```
import crypto from "node:crypto";
import express from "express";

const app = express();
const SECRET = readSecret("bpmsquare_webhook_secret"); // your own secret store; the whsec_… value

app.post("/hooks/bpmsquare", express.raw({ type: "application/json" }), (req, res) => {
  const given = req.get("X-BPMSquare-Signature") || "";
  const expected = crypto.createHmac("sha256", SECRET).update(req.body).digest("hex");
  const ok =
    given.length === expected.length &&
    crypto.timingSafeEqual(Buffer.from(given), Buffer.from(expected));
  if (!ok) return res.status(401).send("bad signature");

  const { delivery_id, events } = JSON.parse(req.body.toString("utf8"));
  for (const e of events) {
    // Remove duplicates on e.id, then handle e.object_type, e.action and e.changes.
  }
  res.sendStatus(200); // any 2xx acknowledges the batch
});

app.listen(3000);
```

Python (Flask)

```
import hashlib
import hmac
import json

from flask import Flask, abort, request

app = Flask(__name__)
SECRET = read_secret("bpmsquare_webhook_secret").encode("utf-8")  # your own secret store; the whsec_… value


@app.post("/hooks/bpmsquare")
def bpmsquare_hook():
    raw = request.get_data()  # the raw bytes, before any parsing
    given = request.headers.get("X-BPMSquare-Signature", "")
    expected = hmac.new(SECRET, raw, hashlib.sha256).hexdigest()
    if not hmac.compare_digest(given, expected):
        abort(401)

    batch = json.loads(raw)
    for event in batch["events"]:
        pass  # Remove duplicates on event["id"], then handle the change.
    return "", 200  # any 2xx acknowledges the batch
```

`read_secret` stands for however your company keeps secrets. Do not put the secret in source code.

If the signature is wrong, answer `401` and do nothing with the body. Do not tell the sender why.

5. Delivery, retries and idempotency

How an Event webhook is delivered:

- **Batches.** Up to 100 events per delivery, oldest first. A larger backlog is delivered over several deliveries.
- **On a schedule.** Deliveries are not instant. Ask your BPMSquare contact how often they run for your workspace.
- **Success.** Any `2xx` answer is a success. BPMSquare then moves its position forward.
- **Failure.** Any other status, a timeout (10 seconds) or a connection error is a failure. BPMSquare does not move forward. The same events are sent again at a later delivery, with a **new** `delivery_id`.
- **Pausing.** After 15 failures in a row the webhook pauses itself. The list shows it as paused. An administrator presses **Resume** (or switches it on) to start it again. It carries on from where it stopped, so nothing is lost.
- **Order.** Events within a batch are in time order. A retry sends the oldest undelivered events first.

What this means for your receiver:

1. **Answer fast.** Acknowledge within 10 seconds. Put the work on a queue and answer `200` straight away.
2. **Expect repeats.** A batch can arrive again, for example when your answer was lost. Make the handler idempotent on `events[].id`. Keep the ids you have processed, and skip any you have seen.
3. **Do not rely on `delivery_id`.** It changes on every attempt. Use the event `id`.
4. **Do not rely on one event per call.** Always loop over `events`.
5. **Be careful with order across batches.** Apply an event only if its `at` is later than the last one you applied for that `object_id`.
6. **Treat the event as a prompt.** For the current state of a record, read it with its endpoint.
7. **Answer `401` only for a bad signature.** Answer `5xx` when you are down, so the batch is sent again. Answer `2xx` for anything you accept, even if you choose to ignore it.

---

6. Call webhooks: one record when something happens

A **Call** webhook sends a single record when something fires it. It is available when **Settings > Integrations > Webhooks** is present in your workspace. A Call webhook does not read the change feed. It has no schedule.

What can fire it:

- A **Standard Quote** is awarded (the Integration middleware, Chapter 5, can choose a webhook as the place an awarded quote goes).
- A **button on Accounts**: tick **Button on Accounts** and every account page gets a **Send to this webhook** entry.
- A **Coverage** route that names the webhook for a group of accounts.
- The **Test** tab of the webhook, with a real record.

Set up

1. **Settings > Integrations > Webhooks > New webhook**.

2. On **Basics**, type a **Name** and choose **Call** as the **Kind**.
3. Choose what it **Sends**: **Accounts** or **Standard Quotes**. The kind and the object cannot be changed later.
4. On **Request**, type the **Address**. The address may hold tokens in the path or query, such as `{{account.ref}}`, but never personal data. Choose the **Method**: `GET` (no body), `POST`, `PUT` or `PATCH`.
5. Choose the **Payload**:
   - **Standard**: a signed record in a fixed shape (below).
   - **Template**: your own JSON, with tokens picked from a list.
   - **Body from mapping**: the body built on the Integration middleware Mapping tab.
6. Add the **Headers** your receiver expects, such as an API key. There can be at most 10. After you save, a header value is hidden and can only be replaced.
7. Leave **Send personal data** off unless the receiver needs email, phone or tax numbers.
8. Set **Wait for an answer (seconds)**. The default is 15.
9. Optionally set **Read the id from the answer**, such as `$.data.number`. BPMSquare keeps that number against the record.
10. Press **Create**. On **Security**, copy the **Signing secret** (shown once). On **Test**, choose a real record and press **Preview** to see the exact request without sending, or press the send button.

What is sent

The Standard payload for an account:

```
{
  "event": "account.pushed",
  "delivery_id": "6c1d9e0a-3b52-4f7e-a8d1-2e4b6c8d0f13",
  "sent_at": "2026-10-28T09:00:00.000Z",
  "workspace": "Example Trading LLC",
  "record": {
    "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
    "url": "https://bpm.example.com/accounts/3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01",
    "account": { "ref": "ACC-0007", "name": "Example Trading LLC", "type": "end_customer", "city": "Dubai",
"country": "AE" }
  }
}
```

The `record` carries the object's allowed fields. Email, phone and tax numbers are left out unless **Send personal data** is on. The event name is `account.pushed` for an account, `quote.awarded` for an awarded Standard Quote, and `record.sent` for anything else. A test uses `webhook.test`. This sample is shortened.

Headers added to every Call webhook. Your own headers cannot replace them:

Header	Value
`X-BPMSquare-Signature`	Hex HMAC-SHA256 of the raw body (an empty body for `GET`). Key: the signing secret.
`X-BPMSquare-Delivery`	A new id for each attempt.
`X-BPMSquare-Event`	The event name.

Header	Value
`X-BPMSquare-Timestamp`	Seconds since 1 January 1970, when the call was made. The signature does not cover it.
`X-BPMSquare-Idempotency-Key`	A key that stays the same for every attempt at the same firing, including a restart from the Monitor. Use it to ignore repeats.

Retries

If the receiver answers `429` or `5xx`, or the connection fails, BPMSquare waits 2 seconds and tries once more. Any other `4xx` is not repeated. If it still fails, the call is shown as **Failed** in the Integration middleware Monitor with a plain-language reason. Press **Restart** there to send it again with the same idempotency key. A success settles the earlier failed lines for that record.

A Call webhook that fails 15 times in a row pauses itself. **Resume** starts it again. A test never changes the failure count.

A webhook that a link, a button or a Coverage route uses cannot be deleted. The panel says where it is used.

The **Log** tab lists the last 20 calls. The Monitor keeps their details for two days.

---

7. Push to ERP (the older way)

If your workspace does not have the Call webhooks above, an account page has a **Push to ERP** action. It sends one account to the **Receiving URL** you set under **Settings > Integrations > Webhooks & push**.

1. Under **Push to external systems**, type the **Receiving URL**. It must be `https`.
2. Under **Signing secret**, press **Generate**. Copy the value at once with **Copy**. It is shown only then. **Regenerate** replaces it and the old one stops at once.
3. Press **Save**.

The body is:

```
{
  "event": "account.pushed",
  "data": { "id": "3f2a9c1e-7b44-4d0a-9a51-0c6d2e8b1a01", "name": "Example Trading LLC", "phone": "+971 4
555 0147", "email": "procurement@example.com" },
  "sent_at": "2026-10-28T09:00:00.000Z"
}
```

`data` is the account record, including its phone, email and tax number. It is shortened here. The call has a 15-second timeout. If a signing secret is set, the call carries `X-BPMSquare-Signature`, signed the same way as above. Without a secret the push is not signed, so always set one.

Verify it exactly as in section 4. Only the shape of the body is different: there is no `events` list. There is no automatic retry. A failure is reported on the screen, and the person presses **Push to ERP** again.

---

8. Telling BPMSquare that an ERP order changed

When your workspace reads sales orders from Dynamics 365 or SAP S/4HANA, BPMSquare refreshes them on a schedule. If you want an order refreshed now, your ERP can tell BPMSquare. This needs the **Sales Orders** and **Integration middleware** modules, and an ERP connection (Chapter 5).

```
POST https://<your-workspace-domain>/api/middleware/webhook/sales-order
Content-Type: application/json
X-BPMSquare-Signature: <hex HMAC-SHA256 of the raw body>
```

```
{ "order_ref": "SO-0004", "erp_id": "0000451122", "sent_at": "2026-10-28T09:00:00.000Z" }
```

Field	Meaning
`order_ref`	The ERP's order number, as shown on the order's ERP line.
`erp_id`	The ERP's own record id. A GUID for Dynamics 365, the sales order number (digits) for SAP.
`sent_at`	Optional. A time more than 10 minutes away from now is refused.

Send `order_ref`, `erp_id`, or both. Keep the body under 8 KB.

The signing key is the **Signing secret** under **Settings > Integrations > Webhooks & push**, the same secret that signs pushes to your ERP. The call is not signed with an API key. If your workspace shows only the newer Webhooks list and you do not see that page, ask your BPMSquare contact to help you set up the secret.

The call goes to your workspace's own address. The body never names a workspace.

Answers:

Answer	Meaning
`200 {"ok":true,"throttled":false,"invoices":1}`	The order and its documents were refreshed. `throttled: true` means it was refreshed in the last 10 seconds and nothing was done.
`400`	The body has neither `order_ref` nor `erp_id`, or is not JSON.
`401`	A bare refusal. The signature, the address or the modules are wrong. The answer does not say which.
`404 {"ok":false,"error":"That order is not known to this workspace."}`	No order matches.
`502`	The ERP could not be read. Try again later.
`503`	A database update is waiting in your workspace. Ask your BPMSquare contact.

An order that the ERP names but BPMSquare does not have yet is read in, if the **Sales orders** link is switched on (Chapter 5).

A Python call:

```
import hashlib
import hmac
import json
import requests

BASE = "https://bpm.example.com"
secret = read_secret("bpmsquare_erp_signing_secret").encode("utf-8")  # your own secret store

body = json.dumps({"order_ref": "SO-0004", "erp_id": "0000451122"}).encode("utf-8")
signature = hmac.new(secret, body, hashlib.sha256).hexdigest()
resp = requests.post(
    f"{BASE}/api/middleware/webhook/sales-order",
    data=body,
    headers={"Content-Type": "application/json", "X-BPMSquare-Signature": signature},
    timeout=30,
)
print(resp.status_code, resp.json())
```

Sign the exact bytes you send.

---

9. Check that it works

- **Event webhook.** Use **Test**. Your receiver must verify the signature and answer `2xx`. The webhook's row shows its last call, its state and its last error. Then change a record and wait for the next delivery.
- **Call webhook.** Use **Preview** to see the request, then send a real record from the **Test** tab. The **Log** tab shows the answer. The Integration middleware **Monitor** (Chapter 5) shows each call as one line, with **View request** and **View response**.
- **Change feed.** Make a change in the app, then call `GET /changes` with your stored cursor. The change appears with your name as `actor`.
- **Signature.** Send a body with one wrong byte. Your receiver must answer `401`.
- **Repeats.** Send the same batch twice. Your receiver must do the work once.

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.

CHAPTER 5

Connectors and the Integration middleware

> 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 the consultant or administrator who connects BPMSquare to other systems. You do not write code for anything in it. You fill in fields, press **Test**, and watch the results.

It covers two screens in **Settings > Integrations**:

- **Connectors**: one tile for each system you can connect. Slack, Google Calendar, Gmail, a mail server, Microsoft Dynamics 365, SAP S/4HANA and ePROMIS.
- **Integration middleware**: one page for everything about syncing with an ERP. The connection, the links between BPMSquare objects and ERP objects, field mapping, a message monitor, and resync.

It also covers the two settings that decide where email goes while you test: **Email output** and **Inbox channels**.

Only workspace administrators can open these screens. The Integration middleware page needs a module that your BPMSquare contact switches on for your workspace. Without it, the page does not appear. Ask your BPMSquare contact.

The examples use a made-up company, Example Trading LLC.

Contents

1. How the pieces fit
2. The Connectors screen
3. Slack
4. Google Calendar
5. Gmail
6. Mail server (SMTP and IMAP)
7. Microsoft Dynamics 365
8. SAP S/4HANA
9. ePROMIS
10. The Integration middleware page
11. What is two-way and what is read-only
12. The ERP id line on records

13. Email output and Inbox channels
14. A checklist for connecting an ERP

1. How the pieces fit

You want to	Connector	Where it is managed
Send quotation emails from your own Gmail, as a reply in the customer's thread	Gmail	Connectors
Use your own business mailbox for supplier RFQs and for the shared Inbox	Mail server, or Gmail	Connectors, then Inbox channels or RFQ settings
Post to Slack	Slack	Connectors
Read a calendar	Google Calendar	Connectors
Bring customers, contacts, products, projects and orders in from Dynamics 365, and send awarded quotes out	Microsoft Dynamics 365	Connectors, then Integration middleware
Bring customers, suppliers, products and orders in from SAP S/4HANA, check and send sales orders	SAP S/4HANA	Integration middleware
Link tenders and budgets to ePROMIS	ePROMIS	Its own settings page

A password, secret or key you enter on any of these screens is stored encrypted. It is never shown again. When you edit a connection and leave a secret field blank, the stored secret is kept.

Sending to or reading from another system always goes out from BPMSquare's servers, never from your browser. Every address you type must be a public `https` address. An address on a private or internal network is refused.

2. The Connectors screen

Open **Settings** > **Integrations** > **Connectors**. The screen shows a tile for each connector, with a **Connected** or **Not connected** label.

Connect a simple connector

Slack, Google Calendar, Gmail and the mail server work the same way.

1. Click the tile. It opens.
2. Press **Connect**.
3. Fill in the fields.
4. Press **Save & connect**. If the details are wrong, the screen says so in red and nothing is saved.
5. Press **Test connection**.

Change or remove one

- These four connectors have no **Edit** button. To change a detail, press **Disconnect**, then connect again.
- **Disconnect** asks you to confirm. Anything that relies on the connector stops working straight away.
- Dynamics 365, SAP S/4HANA and ePROMIS are different. They have **Edit** and keep a stored secret when you leave its field blank.

Which tiles you see

- **Dynamics 365**: always. With the Integration middleware module, the tile opens that page (**Managed in Integration middleware**). Without it, the tile opens the older Dynamics 365 page (section 7).
- **SAP S/4HANA**: only with the Integration middleware module.
- **ePROMIS**: only with the GCC tender module.

---

3. Slack

What it is. A link to a Slack channel through an Incoming Webhook.

What you need. An Incoming Webhook URL from Slack. It looks like `https://hooks.slack.com/services/…`. In Slack, make an app for your workspace, switch on Incoming Webhooks and add one for the channel.

Steps.

1. Open the **Slack** tile. Press **Connect**.
2. Paste the **Incoming Webhook URL**.
3. Press **Save & connect**.
4. Press **Test connection**. A message "BPMSquare is connected to this Slack channel." appears in the channel.

What it does today. The connection is stored and can be tested. At this snapshot no BPMSquare screen sends its notifications to Slack yet. Connect it now if you want it ready.

Disconnect. Press **Disconnect** and confirm.

---

4. Google Calendar

What it is. Read-only access to a calendar through its private iCal address.

What you need. The secret address in iCal format. In Google Calendar, open the calendar's settings, go to **Integrate calendar**, and copy the **Secret address in iCal format**. Anyone who holds that address can read the calendar, so treat it like a password.

Steps.

1. Open the **Google Calendar** tile. Press **Connect**.
2. Paste the **Secret iCal URL**.
3. Press **Save & connect**, then **Test connection**.

The test reads the calendar and says how many upcoming events it found, and what the next one is. A calendar with no upcoming events still passes: "Connected, no upcoming events found on this calendar."

What it does today. The connection is stored and can be tested. At this snapshot no BPMSquare screen shows calendar events yet.

---

5. Gmail

What it is. Quotation emails are sent from your own Gmail address instead of BPMSquare's sender. When the customer's original email is found in your inbox, with a matching subject, the quote goes out as a real reply in that thread.

What you need.

- A Gmail address.
- A 16-character **App Password**. Google issues these only when 2-Step Verification is on for the account. In your Google Account, go to Security, then 2-Step Verification, then App passwords.

Steps.

1. Open the **Gmail** tile. Press **Connect**.
2. Type the **Gmail address** and the **App password**.
3. Press **Save & connect**.
4. Press **Test connection**. A test email is sent from your address to your address.

Gmail can also be chosen as the **Mailbox** of an Inbox channel (section 13).

---

6. Mail server (SMTP and IMAP)

What it is. Your own business mailbox on any mail server: Zoho, Microsoft 365 with IMAP switched on, cPanel, or your hosting provider. BPMSquare sends supplier RFQs from this address and reads the replies back. It can also be the mailbox for an Inbox channel.

What you need. The mailbox address, the user name (usually the address), the password or an app password, and the server details from your mail provider.

Fields.

Field	Example
Email address	`purchasing@example.com`
Username (usually the email address)	`purchasing@example.com`
Password or app password	
SMTP server	`smtp.example.com`

Field	Example
SMTP port	`587` (or `465`)
IMAP server	`imap.example.com`
IMAP port	`993`

Steps.

1. Open the **Mail server (SMTP / IMAP)** tile. Press **Connect**.
2. Fill in the fields and press **Save & connect**.
3. Press **Test connection**. BPMSquare logs in to both servers. The answer names them: "Logged in to smtp.example.com (send) and imap.example.com (receive) as …".

To send RFQs from this mailbox, choose it as the mail channel in the RFQ settings.

---

7. Microsoft Dynamics 365

What it does. It brings records in from Dynamics 365 and sends awarded quotes out. It works with three products. The product you choose decides what an awarded quote becomes there.

Product	What an awarded quote becomes
Sales, Customer Service, Field Service, Project Operations (Dataverse)	A **Quote**
Finance & Operations / Supply Chain	A **Project**
Business Central	A **Project**

A **Sales Order** is also offered as the document. On Finance & Operations and Business Central that choice stays greyed out as "not ready" until its entity is set on the Mapping tab, and it has not yet been proven against live systems.

What you need before you start.

- An app registration in Microsoft Entra for BPMSquare. You will copy its **Directory (tenant) ID** and **Application (client) ID**.
- A **client secret** for that app, and the day it stops working.
- In Dynamics, an application user for that app with a security role. The role must be able to read customers, contacts and products, and to create quotes (or projects). A role that is too narrow shows up as a missing permission when you press **Test**.
- The address of your environment, for example `https://contoso.crm4.dynamics.com`. For Business Central online, the environment name, such as `Production`. For Business Central on-premises, the API base address of your server.
- For Finance & Operations, the legal entity. For Business Central, the company.

Login types. Choose one. Finance & Operations and Dataverse accept the first three. Business Central also accepts the fourth.

Login type	Use when
OAuth 2.0, app registration and client secret	The usual choice.
OAuth 2.0, app registration and certificate	Your company forbids client secrets. You paste the certificate and the private key. The private key is stored encrypted and not shown again.
OAuth 2.0, sign in as a user	Records must be read as a person. BPMSquare shows a **Redirect address**. Add it under **Web** in the app registration, then press **Connect with Microsoft**.
Basic, username and password or web service key	Business Central on-premises only.

Set up (Integration middleware page).

1. Open **Settings > Integrations > Connectors**, click **Dynamics 365**, and press **Managed in Integration middleware**. You can also go straight to **Settings > Integrations > Integration middleware**.
2. If nothing is connected yet, the page shows **Connect Dynamics 365**. Press it.
3. Choose the **Product**.
4. Choose the **Login type**.
5. Type the **Address**.
6. Type the **Directory ID** and **Client ID**.
7. Paste the **Secret** and set **Secret expires on**. The page warns you 30 days before that day.
8. For Finance & Operations or Business Central, type the **Legal entity** or **Company**.
9. Press **Connect**.
10. Press **Test**. The answer is one line, such as "The connection and the permissions are fine." If a permission is missing, the line says how many, and **Show details** lists them with what to do.

After that, the connector chip at the top of the page shows a green dot and the address. Click the chip to **Test**, **Edit** or **Disconnect**. A **Disconnect** keeps the records that were already brought in. Nothing new goes out until you connect again.

Without the Integration middleware module, the Dynamics 365 tile opens an older page named **Microsoft Dynamics 365**. It has the same connection fields and its own sync buttons. The rest of this chapter describes the middleware page.

What it reads. Customers become accounts. Contacts become contacts. Products become products. Projects become projects. With Dataverse, sales orders become sales orders, and their deliveries and invoices come in as read-only documents.

8. SAP S/4HANA

What it does. It brings records in from SAP S/4HANA and reads sales orders back. It works with S/4HANA Cloud (public edition) and with S/4HANA systems that expose the standard OData APIs. Sales orders can be checked with SAP, and sent to it, from the order page. Awarded quotes are not sent to SAP from the Links tab. The Links tab for SAP shows only what comes in.

SAP S/4HANA has no tile of its own until your workspace has the Integration middleware module. It is managed on that page only.

What you need before you start.

- The **API host** of your SAP system, for example `my426318-api.s4hana.cloud.sap`. Type it without `https://` and without a path.
- A **Communication User** of a **Communication Arrangement**. The arrangement needs scenario `SAP_COM_0008` (Business Partner, Customer and Supplier Integration), and `SAP_COM_0009` (Product Integration) for products.
- That user's password.

To try BPMSquare first without your own SAP system, choose the sandbox login. It needs an **API key** from the SAP Business Accelerator Hub (open an API and choose **Show API Key**). The sandbox gives sample data, read only. The panel shows it as "Sandbox (sample data)".

Set up.

1. Open **Settings > Integrations > Integration middleware** and click the **SAP S/4HANA** chip.
2. Press **Connect SAP S/4HANA**.
3. Choose the **Login type**: Communication user, or API key.
4. For a Communication user, type the **API host**, the **Communication user** and the **Password**. For the sandbox, paste the **API key**.
5. Press **Connect**, then **Test**.

What it reads.

From SAP	Into BPMSquare
Customers	Accounts
Suppliers	Suppliers
Products	Products
Sales orders	Sales orders (and their deliveries and invoices, as read-only documents)

Each record keeps SAP's own number, shown on the record (section 12).

Narrowing what is read. On the **Resync** tab, **Scope > Edit** limits a run to records of a given category, grouping or industry. Sales area and country are not offered as filters yet.

Sending an order to SAP. An order can be sent only when its order type, or the coverage it falls under, has a sales area chosen under **Mapping > Value mapping**. With no row there, **Send to ERP** on the order is not possible. The Sales and tenders chapter of the Solution Guide describes the order page.

9. ePROMIS

What it is for. ePROMIS is a project and cost system used with tenders. BPMSquare can read its lists (tenders, the Resource Master, cost codes and justifications) and send an awarded tender's project and budget to it.

Where it lives. Its own page: **Settings > Integrations > Connectors > ePROMIS**. It needs the GCC tender module. It is not on the Integration middleware page.

What you need. From your ePROMIS supplier: the API base address, how requests sign in (an API key in a header, a bearer token, or a user name and password), the credential, a company or database code if they ask for one, and the description of the API.

Fields.

Field	What it holds
Base URL	The ePROMIS API address, starting with `https`.
Sign-in	API key in a header, Bearer token, or User name and password.
Header name	For an API key: the header that carries it, spelled as the ePROMIS documentation spells it. The default is `X-API-Key`.
User name	For User name and password.
API key, Token or Password	The credential. Stored encrypted. Leave blank on a later save to keep it.
Company code and Sent as	A code some setups need on every request, and whether it travels in a header or in the address.
Endpoint map (JSON)	Where each ePROMIS operation lives. See below.

The endpoint map. ePROMIS does not publish a public description of its API. BPMSquare therefore does not guess one. The endpoint map says, for each operation, the path, the method and where in the answer to find the values. The operations are: Test address, Tenders list, Project read, Project create, Budget create, Budget replace, Resource Master list, Cost Code list and Justification list. Until you fill an operation in, it answers "Not configured". **Start from the empty template** loads the empty shape.

Steps.

1. Open the **ePROMIS** tile and press **Set up** (or **Open settings**).
2. Fill in the Base URL, Sign-in and the credential. Add the company code if ePROMIS asks for one.
3. Paste the endpoint map from the ePROMIS documentation. Start from the empty template.
4. Press **Connect**.
5. Press **Test**. With a Test address in the map, it calls that. Without one, it contacts the base address and says "Connected" only if that proves the credential.
6. **Edit** and **Disconnect** are on the connected line.

Limits at this snapshot. The connection, the endpoint map and the test are in place. ePROMIS is not yet on the Integration middleware page, so it has no Links, Monitor or Resync tabs. Fill in and test the endpoint map against your ePROMIS system before you rely on it.

---

10. The Integration middleware page

Open **Settings > Integrations > Integration middleware**. This is where everything about syncing with an ERP lives. The page serves every connector that has the module: **Dynamics 365** and **SAP S/4HANA**. A chip for each is at the top. The chosen connector is kept in the address (`?connector=`), so a link or a reload comes back to the same one.

Where each part of the page is.

Part	What it is for
Connector chips	One chip for each connector, with a dot for its state. Click the open chip to see or change the connection.
Setup line	A slim line: "Set up 3 of 5 · Next: …". It opens the next step. It is hidden when every step is done.
Links tab	Which BPMSquare objects are linked to which ERP objects, and which links are on.
Mapping tab	Which BPMSquare field fills which ERP field, and the value lists.
Monitor tab	Every message in and out. Restart a failed one.
Resync tab	Read data from the ERP now or on a timetable.

The tab is kept in the address (`?tab=links` , `mapping` , `monitor` , `resync`).

The setup line

For Dynamics 365 there are five checks. For SAP S/4HANA, which only reads, there are two (the first and the third).

Check	Done when
Connect	The connection works.
Choose the links	You opened Links, checked the Standard Quote link and saved once. The defaults being on does not count as a choice.
Read customers	Customers have been read from the ERP at least once. A quote needs its customer to exist in the ERP.
Send a test quote	A test quote went through.
First real quote	A real Standard Quote was sent.

Links

One table, one row for each link. Columns: **BPMSquare**, an arrow, **ERP**, **Runs**, **On**.

For Dynamics 365:

BPMSquare	Arrow	ERP	Runs
Standard Quote	goes to	Quote, Project or Sales Order (what your product creates)	When awarded
Accounts	comes from	Customers	Off, Hourly or Daily
Contacts	comes from	Contacts	Off, Hourly or Daily
Products	comes from	Products	Off, Hourly or Daily
Projects	comes from	Projects	Off, Hourly or Daily
Sales orders	comes from	Sales orders (Dataverse only)	Off, Hourly or Daily

For SAP S/4HANA, the rows are Accounts from Customers, Suppliers from Suppliers, Products from Products, and Sales orders from Sales orders.

Switch a row on with its **On** switch. A row cannot be switched on if the ERP is not connected. A row that says **Database update needed** cannot be switched on yet. Ask your BPMSquare contact.

The Standard Quote row. Click it to open its panel.

- **Create as**: what the awarded quote becomes in the ERP. A greyed choice is not ready for your ERP yet and says why.
- **Which webhook**: appears when you choose **Webhook**. The quote is sent to a Call webhook from **Settings > Webhooks** instead of the ERP (Chapter 3). The webhook list is empty until you make one.
- **If the customer has none: price list**: for a Dynamics Quote or Sales Order. A quote uses the customer's own price list in Dynamics. This one is used only when the customer has none.
- **Include**: what travels with the quote. The items are Customer prices, Summary text, Drawing link, Notes and terms, Security profile, Materials list, Project tasks and Internal costs. An item that cannot be sent to the chosen target is greyed, with the reason. A quote or sales order carries prices only. The materials list, project tasks and internal costs go to Project targets.
- **Try it**: pick a Standard Quote, then **Preview** or **Send a quote now**. Preview shows in words what would be sent, and the exact body behind **Show the exact body**, and sends nothing. **Send a quote now** is safe to repeat. A quote never lands twice.
- **Save**. Save once even if you change nothing. That is what completes the "Choose the links" check.

The row's menu (**More**) has **Send a test quote**. It builds a clearly marked sample quote whose reference starts with `ZZ`, sends it, and shows its ERP number and an **Open** link. BPMSquare never deletes a test quote. You do.

When a quote is sent. A Standard Quote goes to the ERP only when it is **awarded**. In BPMSquare, awarded is the status Accepted. A rejected quote is never sent. The send happens after the person saves. It never slows or fails the save. If it fails, the message shows in the Monitor, and you can restart it.

Mapping

Works with no changes. Open it only if your ERP is customised.

- **Mapping for** (a choice at the top): the document to map. For Dynamics 365 these are Quote to the ERP, Customers, Contacts, Products and Projects from the ERP, and one entry for each Call webhook that takes a quote. For SAP S/4HANA: Customers, Suppliers and Products.
- **Fields** or **Value mapping** (a switch at the top).
- **Fields.** Each line sends one BPMSquare field to one ERP field. A line with no use shows **Not sent**. The ERP side is a pick-list read from the ERP's own description, with its display name, technical name, type and a required mark. **Fetch fields** reads that description again, or takes a pasted sample, or a list you type. Fields that are new or gone are flagged, never dropped.
- **Convert** on a line changes a value on its way: a code from a list, a date format, a join of fields or a fixed value. **If empty** gives a value to send when the BPMSquare field is empty.
- **Propose** fills the empty lines by matching names and types. **Propose by name** fills only empty lines. **Replace all by name** starts over. **Ask AI for the empty lines** is offered only if your workspace has it, and sends field names and types, never record data. You accept each suggestion. Nothing is saved until you press **Save mapping**.
- **Add a fixed value** adds a line that sends a constant. **Remove** deletes a line you added. **Show all** opens a long list.
- **Show sample data** (for fields that come in): five sample rows as the ERP returns them.
- **Try it** (for fields that go out): the exact body the saved mapping would send for a sample quote. Nothing is sent and nothing is recorded.
- **Reset to standard** puts the standard mapping back. The page says "Using the standard mapping" until someone edits it.
- **Value mapping.** Lists that turn BPMSquare values into the ERP's codes, such as `Mtr` into `M`. One list serves every document of this connection. The inbound side of a list says what BPMSquare value a returned code becomes. Exactly one row answers a code. If a code is unknown, a default can be set. If you leave the default empty, the code is kept as it arrived and counted.
- For SAP, the only list is the **sales area** (and, for the company, the ERP company) of an order type or a coverage. The order type's row wins.

Save with **Save mapping**. If you try to leave with unsaved changes, the page asks.

Monitor

Every message in and out, newest first. It lists connector messages, webhook calls, emails and WhatsApp messages. Calls made to your workspace's API are recorded too, with their method, path and status and never their body. In the connection panel, **Keep bodies of inbound pulls and API calls** keeps the request and answer of those for two days.

Filters.

- Status chips: **All**, **Failed**, **Waiting**, **OK**.
- **Direction**: In and out, Out, In.
- **Document or entity**: Everything, Quote, Customers, Contacts, Products, Projects.
- **From** and **To** dates, in UTC days.

- **Search**: a record reference or an ERP number.
- **Clear filters**.

Columns. Time, Direction, What, Record (a link to the BPMSquare record), ERP number, Status.

Status words.

Word	Meaning
Sent to Dynamics 365 (or SAP S/4HANA)	An outgoing message reached the ERP.
Received from Dynamics 365 (or SAP S/4HANA)	An incoming read worked.
Sent, Received	The same, for webhooks, email and other channels.
Failed	It did not work.
Waiting	It is queued.
Skipped	Nothing was sent, on purpose.
Redirected (demo)	A demo workspace recorded it and sent nothing.
Dry run	A preview.

A line with detail has a small arrow next to its status. Click it to read why. A failed line also has a **What to do** line in plain words. A restart result appears there too.

Actions. Tick lines. The action bar then offers:

- **Restart** sends the ticked failed lines again. With several ticked it reads **Restart (3)**. With no line ticked and the **Failed** chip on, it reads **Restart all failed (12)**. It asks you to confirm and handles up to 200 lines. Nothing lands twice in the ERP. A line that cannot be restarted says so. After a restart each line shows its answer: **Done**, **Still failing** with the reason, **Already fine**, or **Cannot be restarted**.
- **View request** and **View response** work on exactly one ticked line that has a recorded body. Otherwise they are greyed, and the hover says **Select one line** or **No payload recorded**. A panel shows the request or the answer, with **Copy**. Bodies are shown with personal data and tokens removed.
- **Refresh**.

How long things are kept. A message line is kept for 30 days. Its request and answer are kept for 2 days. Each time an administrator opens a body, the look is written to Change history.

Resync

Moves data from the ERP into BPMSquare on demand. Messages and failures are in the Monitor.

Two views: **Full resync** and **Specific records** (kept in the address as `?view=`).

Full resync. One row for each inbound entity.

Column	Meaning
Entity	Customers, Contacts, Vendors (Suppliers for SAP), Products, Projects, Sales orders. A row for an entity that is off in Links says **Switched off in Links**.
Scope	Which records of the ERP a run reads. **Edit** opens the scope panel.
Last run	When it ran and what it did.
Automatic	**Off**, **Hourly** or **Daily**. Daily runs at 02:00 UTC, as the screen says.
Run	**Sync now** reads what changed since the last run. **Full resync** reads everything within the scope.

A **Scope** is a set of conditions. A record is read if it matches every row. The values inside one row are alternatives. Records that fall outside the scope later stay as they are in BPMSquare. Nothing is ever deleted. After you change a scope, the row says a Full resync brings in the rest.

Automatic runs follow BPMSquare's own timetable, so use **Sync now** when you need the data at once. Only one run for an entity can be going at a time.

Specific records.

- **Get records from the ERP**: choose an **Entity**, type up to 100 **Ids or numbers** separated by commas, spaces or new lines, and press **Get N records**. An id is the long code of a record. A number is the one people see. The scope does not apply, and **Sync now** is not affected.
- **Send a record to the ERP** (Dynamics 365 only): choose **Kind of record**, either **Account** or **Standard Quote**, pick the record and press **Send**. An Account goes to the webhook or receiving URL you set up in Chapter 3, the same as the **Push to ERP** action on an account. A Standard Quote follows its awarded or rejected rule.

11. What is two-way and what is read-only

Object	Dynamics 365	SAP S/4HANA	Notes
Customers to Accounts	In	In	
Contacts	In	Not yet	
Suppliers (Vendors)	Not a link on the page	In	The older Dynamics 365 page also reads vendors into Suppliers.
Products	In	In	
Projects	In	Not available	
Sales orders	In (Dataverse), as read-only	In	An order that comes from the ERP is read-only in BPMSquare. Its deliveries and invoices are read-only documents.

Object	Dynamics 365	SAP S/4HANA	Notes
Sales orders out	Send from the order page, to the ERP	Send from the order page, to SAP	The ERP accepts it, then it locks in BPMSquare.
Awarded Standard Quote	Out, as Quote, Project or Sales Order	Not sent	A rejected quote is never sent.
Account	Out, one at a time, to your webhook or receiving URL	Not available	From Resync, Specific records, or Push to ERP on the account (Chapter 3).
Anything else	Not exchanged	Not exchanged	

BPMSquare reads in only what you switch on in **Links**.

A record that came in from the ERP is shown with the ERP's own number. A record BPMSquare sent is linked to its ERP record, so a second send updates or skips and never duplicates.

---

12. The ERP id line on records

Where a record is linked to an ERP record, its page shows one extra line:

```
Dynamics 365 · QUO-01002 · Open in Dynamics
```

It names the system, the number people use in the ERP, and, for Dynamics Dataverse, a link to open the record there. The long technical code is shown only as a tooltip. The line is read-only. It is not BPMSquare's own reference.

It appears on accounts, contacts, products, suppliers, inventory items, Standard Quotes, quotations, deals, estimates, RFQs, purchase orders, invoices, work orders, cases, workforce projects and employees, when they are linked. A record with no link shows nothing.

Search by ERP number in the Monitor's search box to find the matching record.

---

13. Email output and Inbox channels

These two settings are not connectors, but they decide where email goes while you build and test.

Email output

Settings > Integrations > Email output. Admin only. It controls where quotations, invoices, campaigns and workforce notifications are delivered.

Option	What it does
Send to business partners	Email goes to the real address on the account, contact or employee record. For a production workspace.
Send to a specified email address	Every business email, from any transaction, goes to the one inbox you type under **Redirect everything to**. The subject and body name the intended recipient.

While you build an integration, choose **Send to a specified email address** and type your own inbox. Switch to **Send to business partners** when you go live. A demo workspace is always in redirect mode and cannot be changed.

The **Outbound emails** log lists what was sent.

Inbox channels

Settings > Integrations > Inbox channels. Needs the Shared Inbox module. A channel is an address your customers write to.

- **Mailbox**: *None* (a forwarding address only), *Gmail*, or *Mail server (IMAP)*. If it says "Not connected yet", connect that mailbox on the Connectors screen first.
- **Reply name** and **Reply from**: who replies appear to come from.
- A connected mailbox works by itself. Receiving by a forwarding address needs an inbound mail domain on the platform. Ask your BPMSquare contact.

---

14. A checklist for connecting an ERP

1. Set **Email output** to your own inbox.
2. In your ERP, prepare the app registration (Dynamics 365) or the Communication Arrangement (SAP S/4HANA) and write down the values the page asks for.
3. Open **Integration middleware**. Connect, then **Test**. Fix any missing permission.
4. On **Links**, switch on the links you need and save the Standard Quote panel once.
5. On **Resync**, press **Sync now** for **Customers**, then the others. Check **Last run**.
6. Open an account. The ERP id line shows.
7. On **Links**, **More > Send a test quote**. Check the ERP number.
8. Award a real Standard Quote. Watch it in **Monitor** until it reads **Sent to** your ERP.
9. Set **Automatic** to **Daily** or **Hourly** for the entities you rely on.
10. Switch **Email output** to **Send to business partners** when you go live.

CHAPTER 6

Data Workbench

> 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 the person who loads, extracts or corrects records in bulk, and for the consultant who plans a migration. Data Workbench works from spreadsheets. It has three modes:

- **Import** creates new records.
- **Export** downloads records.
- **Update** changes records that already exist.

Data Workbench is a screen, not an API. It signs in as a person, not with an API key. If a system must create or change records on its own, use the REST API (Chapter 2). Only quotations can be written there. For other objects, a person runs Data Workbench.

The examples use a made-up company, Example Trading LLC.

Contents

1. Where it is and who can use it
2. Objects available
3. Import
4. Export
5. Update
6. Custom fields
7. Limits and rules
8. A checklist for a migration

---

1. Where it is and who can use it

Open **Admin > Data Workbench**. The mode is a tab at the top (**Import**, **Export**, **Update**). It stays when you reload the page.

- It needs the **Data Workbench** module, and a business role that includes the Data Workbench entry.
- Each object also needs its own module. If your workspace does not own Cases, Cases do not appear in the list, and the server refuses it too.

- Every row created or changed is recorded in **Change history**, with the name of the person who ran it.

2. Objects available

The list on the left shows only the objects your workspace owns. In **Update**, it shows only those that can be updated.

Object	Import	Export	Update	Notes
Accounts	Yes	Yes	Yes	Import first. Other objects link to it by name.
Contacts	Yes	Yes	Yes	Needs an existing account, by name.
Assets	Yes	Yes	Yes	A blank account makes loaner stock.
Suppliers	Yes	Yes	Yes	
Products	Yes	Yes	Yes	Includes quantity breaks.
BOM components	Yes	Yes	Yes	Adds components to a BOM or kit product that already exists.
Inventory	Yes	Yes	Yes	
Employees	Yes	Yes	Yes	Includes custom fields.
Users	Yes	Yes	No	Sends an email invitation to each person. Never updated in bulk.
Consents	Yes	Yes	No	An append-only ledger. A change is a new answer. Needs Marketing.
Quotes	Yes	Yes	Yes	One row for each line, with the header values repeated. Import creates the quote with its lines in one file.
Quote lines	Yes	Yes	Yes	Adds lines to an existing quote.
Standard quotes	Yes	Yes	Yes	Header only. Add lines with Standard quote lines.
Standard quote lines	Yes	Yes	Yes	Only for existing draft documents, found by their Quote ID.
Sales orders	Yes	Yes	Yes	Header only. Add lines with Sales order lines.
Sales order lines	Yes	Yes	Yes	Only for existing draft orders, found by their Order ID.
Cases, Work orders, Invoices, Purchase orders	Yes	Yes	Yes	
Deals (pipeline)	Yes	Yes	Yes	Amount and probability are computed, not imported.

Object	Import	Export	Update	Notes
RFQs	No	Yes	Yes	An RFQ is raised from a record, with its lines and suppliers. Export, then update header fields.
Estimates	Yes	Yes	Yes	Header only. Import makes a draft from a template name.
ABM programs, ABM target accounts, ABM buying groups	Yes	Yes	Yes	Needs ABM.
Projects (workforce)	Yes	Yes	Yes	Import parents before children.

Contacts, tenders and contract projects are not available through the API (Chapter 1). Contacts are available here.

3. Import

Import creates new records. It never needs an id, because a new record has none yet.

Steps

1. Choose **Import**, then pick the object on the left, for example **Accounts**. The panel says "Import … first" for anything it depends on. For example, contacts need accounts.
2. Under **Or start from the template**, press **Download Accounts template**. The template carries every column and example rows. It always matches your current field setup, including your custom fields.
3. Replace the examples with your data. Save as `.csv`, `.xlsx` or `.xlsm`.
4. Under **Upload your file**, drop the file or click to choose it. It can be up to 15 MB. Your own column names are fine.
5. On **Match columns**, point each column of your file at a BPMSquare field. Columns whose names match a field's label, or a common alias such as "qty" or "description", are matched for you. Check the rest.
6. On **Review**, read "N ready" and "N with problems". Rows with problems are left out. Press **Show only problems** to see them, and **Download problem list** to fix them in your spreadsheet.
7. Press **Import N rows**.
8. The **Result** step shows how many rows were imported, skipped and failed, and the reason for each. Press **Download report** to keep it.

The four steps are named **Upload**, **Match columns**, **Review** and **Result**.

Required columns

A row is rejected if a required column is empty. The required columns are:

Object	Required
Accounts	Name, Type

Object	Required
Contacts	Name, Account (an account that exists, by name)
Assets	Name, Kind
Suppliers	Name
Products	Name
Inventory	Name
Cases	Account, Type, Equipment label, Complaint
Work orders	Account, Authorizing quote (a Quote ID)
Invoices	Account
Purchase orders	Supplier
Quotes	Account. The quote name is optional. If you leave it blank, one is made from the account name.
Deals	Title, Account
Estimates	Name, Template (the name of an active estimate template)
ABM programs	Name
Projects (workforce)	Name
Users	Name, Email, Role

The template marks required columns. Your workspace may also make other fields required in its field setup.

Linking to other records

You link a new record to an existing one by a business key you type: an account name, a Quote ID such as `QT-2026-0042`, a program ID such as `ABM-0001`, a template name. The names are matched case-insensitively. The match is made within your workspace only.

- A name that is not found fails that row, with the reason ("Account … was not found, import it first").
- A name that is new in the same file does not count. Import the parent first, in an earlier file.
- Quotes: put the quote header on the first row of each quote and one line on every row. Repeat the same value in the optional **Quote name** column on each row of a quote if you want to name it yourself. To add lines to a quote that exists, use **Quote lines**.
- Quotes can also be created from a PDF or Word document. The product reads it, you review the rows, then import.

What happens with duplicates

Import does not replace anything.

- **Accounts** skip a row whose name already exists, and a name that appears twice in the file. The result says so.
- **Users** skip an email that already has an account.
- **Every other object creates a new record for every row.** Products, contacts, suppliers, assets and inventory do not check for an existing SKU or name. If you load the same file twice, you get two copies.

To correct records that are already in, use **Export** and **Update**.

---

4. Export

Export downloads records as a CSV file.

1. Choose **Export**, then an object.
2. Under **Choose columns**, tick the columns you want. **Record ID** is always included.
3. Under **Filters**, press **+ Add filter** to export only matching rows. Pick a field, a condition and a value. Rows must match every filter. Leave it empty to export everything.
4. Press **Download … CSV**.

Conditions depend on the field: *contains*, *is*, *is not*, *is empty*, *is not empty* for text; =, >, < for numbers; *on*, *before*, *after* for dates.

The first column of the file is **Record ID**. That is the BPMSquare id (Chapter 1, section 6). Keep it. It is what Update and your own systems match on.

Cells that start with `=`, `+`, `-` or `@` are protected with a leading mark, so a spreadsheet does not run them as formulas.

An export contains real data, including personal fields such as phone and email. The cost price of products is in the Products export, but never in the API.

---

5. Update

Update changes records that already exist. It always needs the **Record ID**.

The rule

A file for Update must come from an earlier **Export**. It must have a **Record ID** column, filled by the export. A file with only names or references is refused. Do not type ids by hand.

The reason is safety. A name can be shared by two records, or be renamed. An id cannot. A row whose id is not in your workspace fails by itself ("No record with id … found in this workspace"). It is never turned into a new record, and it can never touch another workspace.

Steps

1. **Export** the records you want to change. Keep the **Record ID** column.
2. Edit the values in the file. Do not edit the **Record ID**.

3. Choose **Update**, pick the object, and upload the file under **Or use a file you exported**.
4. On **Match columns**, check the mapping. **Record ID** is matched for you.
5. On **Review**, read "N ready" and "N with problems". Rows with problems are left out.
6. Press **Update N rows**. The **Result** step reports updated, skipped and failed rows with reasons.

What Update changes

- **Only the columns you map.** A column you leave unmapped is not touched.
- **A blank cell in a mapped column clears that value.** If you do not want a field cleared, delete its column from the file, or leave it unmapped.
- **Plain fields only.** Fields that link to other records (for example the account of a contact) are not updated in bulk. Change those on the record.
- **Computed columns are not editable.** Totals, statuses set by a workflow and the reference (such as the Quote ID) are export-only.
- **Quotes: header fields only,** one row for each quote. Use **Quote lines** to correct lines.
- Each row that changes something writes a **Change history** entry with the old and new values. Personal fields are recorded as redacted.

Rows are applied one at a time, a few in parallel. Updating thousands of rows takes longer than importing the same number.

---

6. Custom fields

Custom fields you make in **Settings > Custom fields** appear as columns in the template, in the Match columns step and in exports, with the prefix `cf_` on their key. You do not do anything special. If you hide a field in your field setup, it leaves the template.

Custom fields exist for accounts, contacts, assets, suppliers, products, inventory, cases, quotations, deals, employees and more. The template for each object lists the ones it has.

Download a fresh template whenever you add or change a custom field. An old template does not have the new column.

---

7. Limits and rules

- **Formats.** `.csv`, `.xlsx` and `.xlsm`. The old `.xls` is not supported. Open it and save as `.xlsx` or `.csv`.
- **Size.** 15 MB for each file. Split larger files.
- **Sheets.** Only the first sheet of a workbook is read. Empty rows are ignored.
- **Rows.** There is no row limit in the product. The whole file is processed in one request, so a very large file can run for a long time. For more than a few thousand rows, split the file into parts of about 1,000 rows and load them one after another. Check the **Result** of each before the next.

- **Order.** Load parents before children: accounts, then contacts and assets, then quotes, then the things that point at quotes. The panel tells you what to import first.
- **No hidden overwrite.** Import adds. Update changes only what you mapped. Neither deletes anything.
- **Users.** Importing users sends each person an email invitation. If **Email output** is set to a test inbox, the invitations go there. Check that setting first (Chapter 5, section 13).
- **No API.** Data Workbench cannot be called with an API key.
- **Cost price.** It is in the Products file. Treat the file as confidential.

---

8. A checklist for a migration

1. **Plan the order.** Suppliers, accounts, contacts, assets, products, inventory, then quotes, deals, cases, work orders, invoices and purchase orders.
2. **Set Email output to your own inbox** before any user import (Chapter 5).
3. **Download the template** for each object from your own workspace, not from an older project.
4. **Fill a small file first.** Ten rows. Import it. Open the records. Fix your mapping.
5. **Clean the data in the spreadsheet,** not afterwards. Names must match exactly between files. A contact's account must be spelled as the account is.
6. **Load in the right order.** One object at a time. Read **Review** before you press **Import**.
7. **Keep every Result report.** They are your audit of what loaded.
8. **Check the counts.** After each load, compare the number of rows you meant to load with the list. With the API you can count without reading rows: `GET /accounts?count=only` returns `meta.total` (Chapter 2).
9. **Do not load a file twice.** Except for accounts and users, a second load creates copies. To correct, Export and Update.
10. **Export a backup** of the loaded records, with **Record ID**. Your own systems can store those ids (Chapter 1).
11. **Switch Email output back** to **Send to business partners** when you go live.

CHAPTER 7

Recipes

> 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 puts the earlier chapters to work. It has four recipes. Each one starts from nothing and ends with a check that proves it worked.

1. Keep an ERP in step with your accounts, using the change feed.
2. Receive a webhook and process it safely.
3. Create a quotation from your own system and read its status back.
4. Load 5,000 products with Data Workbench.

The code is Node.js 18 or later. Only the receiver in recipe 2 needs a package (`express`). Every example reads its key or secret from a small text file next to the script, such as `bpm.key`. Keep those files out of your source repository. The workspace address `bpm.example.com`, the customer Example Trading LLC and the ids are made up. Use your own.

Before you start any recipe, read Chapter 1, sections 3 and 6.

Contents

1. Keep an ERP in step with accounts
2. Verify and process a webhook safely
3. Create a quotation and read its status back
4. Load 5,000 products with Data Workbench
5. Common questions

---

1. Keep an ERP in step with accounts

Goal. Every account in BPMSquare exists in your ERP, and every change or deletion reaches it within minutes. BPMSquare is the source of truth. Your ERP only receives.

You need.

- A scoped key, **Read** only, with the single object **Accounts**. Name it "ERP account sync".
- A way to create or update an account in your ERP by BPMSquare's account `id`. Store that id in a field of your ERP record.

- A job scheduler that can run a script every five minutes.

The plan.

1. Note the time. Read every account once with the list endpoint and the detail endpoint, and write each to the ERP.
2. Then read the change feed for accounts, from the beginning. Ignore events older than the time you noted. For each other event, read the account again and write it to the ERP, or remove it if it is gone.
3. Save the cursor after each page. From then on, each run starts from the cursor.

Why read the account again, rather than use the `changes` in the event? Because the event may be redacted (phone and email show as `[redacted]`), and because reading the record again makes every step safe to repeat.

The script

Save as `sync-accounts.mjs`. Put the key on one line in `bpm.key`.

```
import { readFile, writeFile } from "node:fs/promises";

const BASE = "https://bpm.example.com/api/v1";
const KEY = (await readFile("bpm.key", "utf8")).trim();
const STATE_FILE = "accounts.state.json";

async function api(path) {
  const res = await fetch(`${BASE}${path}`, { headers: { Authorization: `Bearer ${KEY}` } });
  if (res.status === 404) return null;
  if (!res.ok) throw new Error(`${res.status} ${path} ${await res.text()}`);
  return res.json();
}

// Replace these two with calls to your ERP. Both must be safe to repeat.
async function upsertInErp(account) {
  console.log("upsert", account.ref, account.name, account.id);
}
async function removeFromErp(id) {
  console.log("remove", id);
}

async function readState() {
  try { return JSON.parse(await readFile(STATE_FILE, "utf8")); } catch { return {}; }
}
const saveState = (s) => writeFile(STATE_FILE, JSON.stringify(s));

async function loadAll() {
  for (let page = 1; ; page++) {
    const list = await api(`/accounts?limit=200&page=${page}&sort=created_at&select=id`);
    for (const row of list.data) {
      const one = await api(`/accounts/${row.id}`);
      if (one) await upsertInErp(one.data);
    }
    if (!list.meta.has_more) break;
  }
}

async function readFeed(state) {
  let cursor = state.cursor ?? null;
  for (;;) {
    const q = new URLSearchParams({ object_type: "accounts", limit: "500" });
    if (cursor) q.set("since", cursor);
    const out = await api(`/changes?${q}`);

    // The last action for each account in this page wins.
    const last = new Map();
    for (const e of out.data) {
      if (new Date(e.at) < new Date(state.floor)) continue; // already covered by the first load
      last.set(e.object_id, e.action);
    }
    for (const [id, action] of last) {
      if (action === "delete") { await removeFromErp(id); continue; }
      const one = await api(`/accounts/${id}`);
      if (one) await upsertInErp(one.data);
      else await removeFromErp(id);
    }
```

```
    cursor = out.cursor;
    await saveState({ ...state, cursor }); // only after the page is done
    state.cursor = cursor;
    if (!out.meta.has_more) break;
  }
}

const state = await readState();
if (!state.loaded) {
  state.floor = new Date().toISOString();
  await loadAll();
  state.loaded = true;
  await saveState(state);
}
await readFeed(state);
```

Run it once by hand: `node sync-accounts.mjs`. The first run loads everything. Run it every five minutes after that.

How it behaves

- If it stops half-way through the first load, the next run starts the load again with a later `floor`. Nothing is lost, and `upsertInErp` is safe to repeat.
- If it stops half-way through a page, the cursor is not saved. The next run reads the same page again.
- An account that was created and deleted between two runs shows as a `delete`. The script removes it from the ERP, which is harmless if it was never there.
- Contact people are not in the `accounts` feed. They come inside `GET /accounts/:id`, under `contacts`. Because the script reads the full account each time, a change to the account is enough to refresh them, but a change to only a contact is not logged under `accounts`. If you need contacts to be fresh, also read the full account on a daily schedule.
- Only fields your key can see arrive. Chapter 1, section 7 says which are personal.

Check it worked.

1. In BPMSquare, rename the account "Example Trading LLC" to "Example Trading LLC (Dubai)".
2. Wait for the next run. Your log shows `upsert ACC-0007 Example Trading LLC (Dubai) …`.
3. Compare totals: `curl -s -H "Authorization: Bearer $KEY" "$BASE/accounts?count=only"` returns `meta.total`. It must equal the number of accounts in your ERP that carry a BPMSquare id.

---

2. Verify and process a webhook safely

Goal. BPMSquare tells your system when quotations change. Your receiver accepts only genuine calls, answers fast, does each piece of work once, and survives a repeat.

You need.

- A public `https` address that reaches your receiver, such as `https://hooks.example.com/hooks/bpmsquare`.
- A workspace administrator to make the webhook.
- A scoped key, **Read** only, for **Quotations**, to read the current state of a quotation.

Set up the webhook.

1. **Settings > Integrations > Webhooks > New webhook** (or **Webhooks & push** if that is what your workspace has).
2. Name it "Quotation changes". Kind **Event**. Under **Changes sent**, tick **Quotes** only.
3. Address: `https://hooks.example.com/hooks/bpmsquare`.
4. Create it. On **Security**, copy the signing secret into a file named `whsec.txt` next to the receiver.

The receiver. Save as `receiver.mjs`. It needs the package `express` (`npm install express`).

```
import crypto from "node:crypto";
import { readFileSync, appendFileSync, existsSync } from "node:fs";
import express from "express";

const SECRET = readFileSync("whsec.txt", "utf8").trim();
const BASE = "https://bpm.example.com/api/v1";
const KEY = readFileSync("bpm.key", "utf8").trim();

// Events already handled. Append-only file so a restart does not forget them.
const SEEN_FILE = "seen-events.log";
const seen = new Set(existsSync(SEEN_FILE) ? readFileSync(SEEN_FILE, "utf8").split("\n").filter(Boolean) :
[]);

const queue = [];
let working = false;

function verify(rawBody, given) {
  const expected = crypto.createHmac("sha256", SECRET).update(rawBody).digest("hex");
  return given.length === expected.length && crypto.timingSafeEqual(Buffer.from(given),
Buffer.from(expected));
}

async function handle(event) {
  if (event.object_type !== "quotes" || event.action === "delete") return;
  // Read the current record. The event is a prompt, not the truth.
  const res = await fetch(`${BASE}/quotations/${event.object_id}`, { headers: { Authorization: `Bearer
${KEY}` } });
  if (!res.ok) throw new Error(`read ${event.object_id}: ${res.status}`);
  const q = (await res.json()).data;
  console.log("quotation", q.ref, "status", q.status, "outcome", q.outcome, "total", q.total);
  // Update your own system here.
}

async function drain() {
  if (working) return;
  working = true;
  while (queue.length) {
    const event = queue.shift();
    try {
      await handle(event);
      appendFileSync(SEEN_FILE, event.id + "\n");
      seen.add(event.id);
    } catch (err) {
      console.error("failed", event.id, err.message);
      // Not marked as seen: it will be handled when BPMSquare sends it again.
    }
  }
  working = false;
}

const app = express();
app.post("/hooks/bpmsquare", express.raw({ type: "application/json", limit: "2mb" }), (req, res) => {
  if (!verify(req.body, req.get("X-BPMSquare-Signature") || "")) return res.status(401).end();

  const batch = JSON.parse(req.body.toString("utf8"));
  for (const event of batch.events || []) {
    if (event.object_type === "ping" || seen.has(event.id)) continue; // a test, or a repeat
```

```
    if (!queue.some((q) => q.id === event.id)) queue.push(event);
  }
  res.sendStatus(200); // acknowledge now; the work happens after
  drain();
});

app.listen(3000, () => console.log("listening on 3000"));
```

Why it is built this way.

- `express.raw` keeps the exact bytes. The signature is checked over them.
- It answers `401` for a bad signature and does nothing else.
- It answers `200` before the work is done, so a slow system never makes BPMSquare time out (10 seconds).
- A piece of work that fails is not marked as seen, but BPMSquare will not send it again, because the receiver already answered `200`. If you cannot afford to lose an event, make `handle` write the event to a durable queue (a database table) before you answer. Answer `5xx` if that write fails. BPMSquare then sends the batch again.
- Each event is handled once because the event `id` is remembered.

Test it without BPMSquare. Save as `send-test.mjs`. It signs a batch with the same secret and sends it three ways.

```
import crypto from "node:crypto";
import { readFileSync } from "node:fs";

const secret = readFileSync("whsec.txt", "utf8").trim();
const url = "http://localhost:3000/hooks/bpmsquare";

const body = JSON.stringify({
  delivery_id: "test-1",
  sent_at: new Date().toISOString(),
  events: [{
    id: "test-event-1",
    object_type: "quotes",
    object_id: "b7d1e0c2-5a3f-4e18-8c92-6f4a1d9e2b03",
    object_label: "QT-2026-0042",
    action: "update",
    changes: [{ field: "status", from: "draft", to: "sent" }],
    actor: "sara.admin@example.com",
    at: new Date().toISOString(),
  }],
});
const sign = (b) => crypto.createHmac("sha256", secret).update(b).digest("hex");

async function send(label, b, signature) {
  const res = await fetch(url, { method: "POST", headers: { "Content-Type": "application/json", "X-
BPMSquare-Signature": signature }, body: b });
  console.log(label, res.status);
}

await send("valid     ", body, sign(body));
await send("repeat    ", body, sign(body));
await send("tampered  ", body.replace("sent", "send"), sign(body));
```

Put the id of a real quotation of your workspace in `object_id` first. Otherwise the read of the quotation fails and the receiver logs `failed` instead.

Expected output: `valid 200`, `repeat 200`, `tampered 401`. The receiver logs the quotation line once, because the repeat is skipped, and logs nothing for the tampered request.

Check it worked on the real thing.

1. On the webhook's **Test** tab, press the test. The receiver logs nothing for a test message and answers `200`. The webhook's row shows the last call as successful.
2. Change a quotation's status in BPMSquare. At the next delivery your receiver logs the quotation. Ask your BPMSquare contact how often deliveries run.
3. Send the `tampered` request again. It must be `401`.
4. Restart the receiver and send the `valid` request again. It must be skipped, because `seen-events.log` remembers it.

---

3. Create a quotation and read its status back

Goal. Your order system creates a quotation in BPMSquare for a customer, follows it until the customer decides, and shows its invoiced and paid amounts.

You need.

- A scoped key with **Read** and **Write**, for **Quotations** and **Accounts**. Name it "Order system".
- The customer already exists in BPMSquare. If not, create it on screen, or load it with Data Workbench (Chapter 6).

The plan.

1. Find the account's `id` by name.
2. `POST /quotations` with your own order number in `ref_no`. Store the `id` that comes back.
3. Read the quotation by `id` for its status.
4. Read the quote-to-cash figures from the list.

The script

Save as `quote.mjs`.

```
import { readFile } from "node:fs/promises";

const BASE = "https://bpm.example.com/api/v1";
const KEY = (await readFile("bpm.key", "utf8")).trim();

async function call(method, path, body) {
  const res = await fetch(`${BASE}${path}`, {
    method,
    headers: { Authorization: `Bearer ${KEY}`, "Content-Type": "application/json" },
    body: body ? JSON.stringify(body) : undefined,
  });
  const json = await res.json().catch(() => ({}));
  if (!res.ok) {
    const detail = json.details ? " " + JSON.stringify(json.details) : "";
    throw new Error(`${method} ${path} -> ${res.status} ${json.error ?? ""}${detail}`);
  }
  return json;
}

// 1. Find the account by name.
const q = new URLSearchParams({ filter: "name:eq:Example Trading LLC", select: "id,name", limit: "2" });
const found = (await call("GET", `/accounts?${q}`)).data;
if (found.length !== 1) throw new Error(`expected one account, found ${found.length}`);
const accountId = found[0].id;

// 2. Create the quotation. Your own order number goes in ref_no.
const created = (await call("POST", "/quotations", {
  account_id: accountId,
  name: "Supply of 22 kW motors and installation",
  ref_no: "ORD-88123",
  quote_date: "2026-10-26",
  valid_until: "2026-11-25",
  gst_rate: 5,
  terms: "Payment 30 days from invoice.",
  lines: [
    { sl_no: "1", description: "Supply of 3-phase induction motor 22 kW", uom: "Nos", qty: 2, rate: 18500,
category: "material" },
    { sl_no: "2", description: "Installation and commissioning", uom: "Lot", qty: 1, rate: 4500,
discount_pct: 10, category: "labour" },
  ],
})).data;
console.log("created", created.ref, created.id, "total", created.total, "status", created.status);

// 3. Read it back by id.
const one = (await call("GET", `/quotations/${created.id}`)).data;
console.log("status", one.status, "outcome", one.outcome, "valid until", one.valid_until);

// 4. Quote to cash, for this account.
const cash = (await call("GET", `/quotations?filter=${encodeURIComponent("account.id:eq:" +
accountId)}&select=ref,status,outcome,total,invoice_count,invoiced_total,paid_total,balance_due`)).data;
console.table(cash);
```

Run `node quote.mjs`. The first line shows the new reference (for example `QT-2026-0043`) and its id, `total 41050`, and status `draft`.

Following it afterwards

- **By id.** Keep the `id` with your order number. Read `GET /quotations/:id`. It carries `status`, `outcome` (`open`, `won`, `lost`, `dropped`), `total`, `revision` and `superseded_by`.
- **By the change feed.** Read `GET /changes?object_type=quotes` with a stored cursor (Chapter 3). A quotation moved to won or lost shows as an update to `outcome`. Match on `object_id`.
- **Revisions.** If someone revises the quotation, the older version gets `superseded_by` set and becomes read-only. A `PATCH` on it answers `409`. Follow `superseded_by` to the newer version.
- **Changing it.** `PATCH /quotations/:id` with only the fields to change. If you send `lines`, send the complete set. They replace the old lines. If the call fails, read the quotation back before you repeat it.
- **Money.** `total` is before tax. `invoiced_total`, `paid_total` and `balance_due` come from the quotation's own invoices that are not cancelled.

A `POST` is not repeatable

If the call times out, you do not know if the quotation was created. Do not simply send it again, or you may create two.

1. List the account's quotations created since you started: `GET /quotations?filter=account.id:eq:<id>;created_at:gte:<time you started>&select=id,ref`.
2. Open each with `GET /quotations/:id` and look for your `ref_no`.
3. Create it again only if none carries it.

Check it worked.

1. The script prints a reference and an id.
2. In BPMSquare, open the quotation. The title, the two lines and the total of 41,050 are there.
3. In **Change history**, the quotation is recorded as created by `api:v1`.
4. Mark it sent on screen. `GET /quotations/:id` now shows `status: "sent"`.

---

4. Load 5,000 products with Data Workbench

Goal. Put a 5,000-item catalogue into **Products**, once, correctly, with prices and quantity breaks.

You need.

- The Products and Data Workbench modules, and a business role that includes Data Workbench.
- Your product category tree set up under **Settings** (categories and sub-categories). The Category and Sub-category columns take the codes from that tree.
- Suppliers loaded first, if you want to name a preferred supplier in the file. The supplier must exist already, spelled the same.
- A read-only API key with the **Products** object, to check the result.

Steps.

1. **Get the template.** Open **Admin > Data Workbench > Import > Products**. Press **Download Products template**. It carries every column of your workspace, your custom fields included, with example rows.
2. **Prepare the data in a spreadsheet.** The main columns are:

Column	Notes
Name	Required.
SKU	Your item code.
Category, Sub-category	Codes from your category tree.
Status	`active` or `inactive`. Blank means active.
Item type	The label or key of one of your item types, such as `finished_product`.
List price, Cost price, Tax %	Numbers.
Cost price as of	A date.
Quantity breaks	One cell: `from:rate; from:rate`, for example `100:390; 500:360`. A blank rate means that quantity is priced normally.
Unit of measure, Lead time (days), Description	
Supplier	The name of an existing supplier. Optional.

   A sample, saved as `.csv`:

```
Name,SKU,Category,Sub-category,Status,Item type,List price,Tax %,Unit of measure,Quantity breaks,Description
22 kW IE3 induction motor,MOT-22-IE3,Motors,Induction,active,finished_product,18500,5,Nos,5:17800; 10:17200,"415 V, 4 pole, B3 foot mount"
Deep groove ball bearing 6310-2RS,BRG-6310-2RS,Bearings,Ball,active,finished_product,64,5,Nos,,"50 x 110 x 27 mm, sealed"
```

3. **Avoid duplicates before you load.** Products Import does not check for an existing SKU or name. Load the catalogue once only. If some products already exist, press **Export** for Products, compare the SKUs with your sheet, and take the existing ones out of the import file. To change prices of products that exist, use Export and Update instead (Chapter 6, section 5).
4. **Split the file** into five files of 1,000 rows: `products-1.csv` to `products-5.csv`. There is no row limit, but the whole file is processed in one request. Smaller files are easier to check and to repeat.
5. **Load a small test first.** Take the first 10 rows into `products-test.csv`. Import it. On **Match columns**, check that every column points at the field you mean. On **Review**, read "10 ready". Press **Import 10 rows**. Open **Products** and check them: price, category, quantity breaks. If something is wrong, correct the products with Export and Update, and fix the sheet. Then take the 10 rows out of file 1, because they are already in.

6. **Load the five files, one by one.** For each: **Upload**, **Match columns** (keep the header names the same in every file, so the columns match the same way), **Review**, **Import N rows**. Read the **Result**. Download the report. A row with a problem, such as a category code that does not exist, is left out and listed with the reason. Fix those rows in a new file, `products-fix.csv`, and import that.
7. **Never import the same file twice.** If a load was interrupted, check **Products** first, to see how far it got.

Check it worked.

1. The five **Result** reports add up to 5,000 imported rows, plus any you fixed later. Add the 10 from the test.
2. Count through the API:

```
curl -s -H "Authorization: Bearer $KEY" "$BASE/products?count=only&aggregate=count,sum:list_price"
```

   `meta.total` is your count of products (plus any that existed before). `meta.aggregates.sum_list_price` should equal the sum of the List price column in your sheets.
3. Spot check one by SKU: `curl -s -H "Authorization: Bearer $KEY" "$BASE/products?search=MOT-22-IE3"`. The price, `qty_breaks` and `item_type` match your sheet. The cost price is not in the API. Check it with an Export of Products.
4. **Change history** lists the creates, with your name.

Common questions

1. Why do I get `404` on an endpoint that is in this guide? Either your workspace does not own the module behind it, or the id is not one in your workspace. A module that is off answers exactly like an address that does not exist. Ask your BPMSquare contact which modules your workspace has (Chapter 1, section 5).

2. Can I create or change accounts, products or invoices through the API? No. Quotations are the only object you can create, change or delete through the API. The other objects are read-only there. To load or correct them in bulk, use Data Workbench (Chapter 6). The one other write is raising a draft project invoice from worked hours.

3. A key that worked yesterday now answers `401`. What happened? The key was revoked, it passed its expiry date, or it is the workspace key and someone regenerated it. A scoped key whose date ran out behaves like a key that was never made. Look at the key list under **Settings > Integrations > API keys & developer**. Make a new key, and give your integration the new value.

4. Should I use the change feed or a webhook? Use the change feed when you want to set the pace, when you cannot offer a public address, or when you need the news within seconds. Use an Event webhook when you want BPMSquare to prompt you, and you are happy with batches on a schedule. Many integrations use both: the webhook says "something changed", and the feed is read for exactly what (Chapter 3).

5. Why did a webhook arrive late, or twice? Webhooks are delivered in batches on a schedule, not at the moment of the change. A batch is sent again if your receiver did not answer with a success, and each repeat has a new `delivery_id`. Keep the event `id` of what you handled, and skip repeats (Chapter 3, section 5).

6. Why does a change show `[redacted]`? Phone, email and tax number of accounts and contacts, and staff phone and email, are protected. The change feed and webhooks show that a change happened but not the value. Read the record with its endpoint if your key has the scope to see it.

7. An awarded quote did not reach Dynamics 365. Where do I look? Open **Settings > Integrations > Integration middleware** and the **Monitor** tab. Set the status chip to **Failed**. Click the arrow next to the status to read the reason and the **What to do** line. Fix the cause, tick the line, and press **Restart**. A quote never lands twice. If there is no line at all, check that the Standard Quote link is on under **Links**, that the quote's status is Accepted (awarded), and that the customer was read in from Dynamics (Chapter 5).

8. Can I run a Data Workbench import from a script? No. Data Workbench is a screen that signs in as a person. A script cannot call it with an API key. For scheduled loads of quotations, use the REST API. For other objects, a person runs Data Workbench, or your BPMSquare contact can advise on another way.