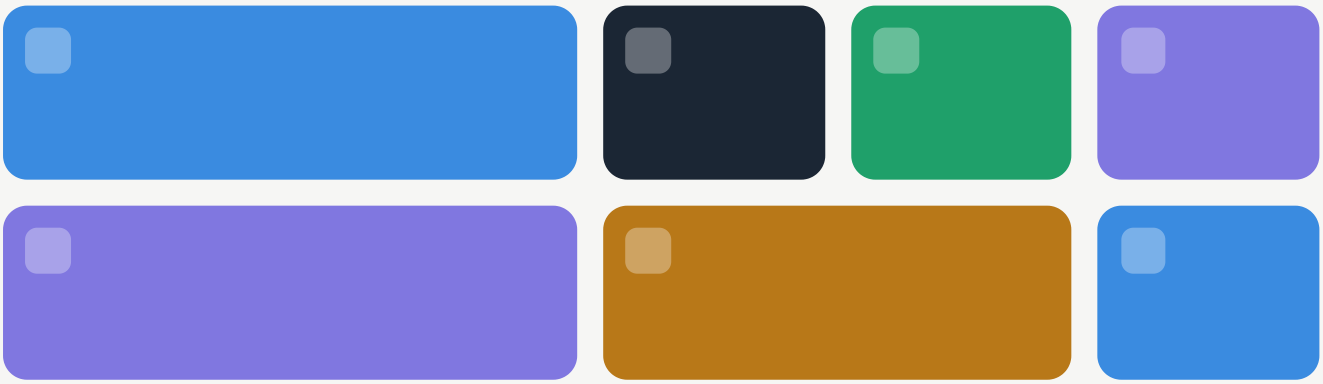
BPMSquare

CUSTOMER EDITION

Overview and security

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

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