TribeManager Admin Manual

Table of Contents

This manual is for **administrators**: the people who set up an organization, choose which modules are active, pick each module's complexity mode, configure fields and categories, and manage permissions.

It is ordered by complexity mode (see SPEC.md, *Module complexity modes*): basic, advanced, then expert. Configuration surface grows with each mode.

1. Basic configuration

Basic-mode administration: the minimum needed to stand up an organization and switch modules on. Sensible defaults are assumed, so there is little to configure - activate the modules you need and start working.

1.1. Contacts

Every organization starts with the vCard-aligned default fields. In basic mode a contact carries just its **name** (`FN`) - enough to tell contacts apart and link them to events, tasks, and the rest.

1.1.1. Complexity mode

The **Contact fields** screen has a **Complexity mode** selector - basic, advanced, or expert. It controls how much contact configuration you see and how many fields contacts expose:

- **Basic** - name only.
- **Advanced** - the everyday set (name, structured name, email, phone, address, birthday), which you can turn on and off and extend.
- **Expert** - the full field configuration.

Raising or lowering the mode never deletes data: a field hidden by a lower mode reappears when you raise the mode again.

1.2. Events

Every organization starts with four **participation roles**, named after the calendar standard's own: chair, required, optional, and observer. They are ordinary data - rename them to whatever your organization actually says, and the export keeps working, because each role carries the standard role it maps to.

1.3. Finances

Set up one **account** per real place money sits - a bank account, a cash box - and one per currency: an account holds a single currency, so its balance is a number rather than a table.

Build the **category** tree under the two fixed roots, income and expense. They are fixed because the income statement is income minus expenses, and a third root would be a figure the report has no line for. Below them the tree is entirely yours, as coarse or as detailed as you want.

Membership-fee rules, dunning, and SEPA mandates are not part of the basic surface and are not built yet.

1.4. Documents

File content is kept in the database by default, so a single `pg_dump` backs up your contacts, your books, and your files together. That is a deliberate choice for self-hosting: the most common way to lose documents is to back up the database and forget the file directory.

`tribemanager.documents.max-size-bytes` sets the largest file the instance accepts - 25 MiB by default. It is the one limit in TribeManager that refuses rather than warns, because it protects the instance itself. Raise it if your organization genuinely needs to, and raise `quarkus.http.limits.max-body-size` with it, or uploads will be rejected before the application can explain why.

1.5. Transactional mail

Invitations, updates, cancellations, and reminders are sent through an SMTP relay you point the instance at - your own mail server, your provider's, anything that speaks SMTP. No mail service is

required and none is contacted unless you configure one.

Setting	What it does
`QUARKUS_MAILER_HOST`, `QUARKUS_MAILER_PORT`	Your SMTP relay. Username and password go in `QUARKUS_MAILER_USERNAME` and `QUARKUS_MAILER_PASSWORD` where the relay wants them.
`QUARKUS_MAILER_FROM`	The sender address mail goes out as.
`tribemanager.mail.locale`	The language transactional mail is written in (`en` or `de`). A per-organization language setting follows when organizations become configurable.
`tribemanager.reminders.interval`	How often the reminder queue is checked. One minute by default, which suits reminders measured in hours and days.

If a message cannot be delivered, the failure is logged and the work that caused it stands: an invitation that bounced is a mail problem, and it never undoes the participation it belonged to.

The development stack (`compose.dev.yaml`) includes **Mailpit** at `http://localhost:8025`. It accepts everything the application sends, shows it in a web page, and delivers nothing onwards - so you can see exactly what an invitation looks like without a real mailbox, or a real recipient, being involved.

Further basic-mode administration (installation and first start, choosing modules) is added here as it lands.

2. Advanced configuration

Advanced-mode administration: custom fields, categories, relationship types, and the other curated configuration that advanced mode exposes on top of the basic defaults.

2.1. Contacts

In advanced mode the **Contact fields** screen pre-configures the everyday default fields - structured name, email, phone, address, birthday - on top of the mandatory name. You can:

- **Deactivate** a default field you do not need. It is hidden, not deleted, and can be reactivated at any time.
- **Add your own fields** - give each a label, a type (text, number, or date), and whether it is required. A new field joins every contact's profile.
- **Edit** a field's label, type, or requiredness, or **delete** a field you added.

No field change loses data: deactivating or deleting only hides a field; its history remains.

Every organization ships with a default **relationship-type** vocabulary (member, customer, supplier, partner, donor, venue). Only the member type carries a status; you can extend the vocabulary with your own types via the API (`PUT /api/v1/relationship-types/{name}`).

2.1.1. Importing contacts

The **Import** screen migrates an existing member list from a CSV: upload the file, map each column to a field, and - to avoid duplicates - choose a **match field** so a row updates the matching contact instead of creating a new one. Rows are never rejected; anything incomplete is imported and flagged as a warning.

2.2. Events

The **participation roles** your organization uses are yours to name. The four that ship - chair, required, optional, observer - carry the mapping the calendar standard expects, and so does any role you add, which is why a locally invented vocabulary still exports and imports losslessly.

2.3. Finances

Advanced finance administration is in place: fee rules attached to relationship types and statuses, matching payments against expectations, and bank-statement import in `camt.053`. Dunning is designed but not built. The basic surface (accounts, categories, entries, the income statement) is complete and is what an organization needs to keep books today.

2.3.1. Bank statements without uploading them

A `camt.053` file is normally uploaded by hand, and that stays available. It can also be collected: set `tribemanager.banking.watch.directory` to a directory and `tribemanager.banking.watch.account` to the account the statements belong to, and everything that lands there is imported on the next run (`tribemanager.banking.watch.interval`, 15 minutes by default).

What writes into that directory is outside TribeManager - the bank's own download, `aqbanking` on a timer, a network share the treasurer drops files on. That is deliberate: no banking credential is ever stored here and no bank is ever contacted.

Imported files move to `done/` inside the directory, unreadable ones to `failed/`. Nothing is deleted, and a file that is still being copied is left alone until it stops changing. Importing the same statement twice is harmless - entries already booked are recognized and skipped.

2.3.2. Donations, and what they need before a receipt exists

Mark the category your donations are booked under - the checkbox on an income category - and everything under it counts too. Nothing else can tell donations apart: not the amount, not the payer, not the description, and no shipped vocabulary knows what your organization calls its own giving. Only booked postings that name a person are counted; an anonymous gift stays in the books with nobody to receipt it to, and a planned one is not money that arrived.

The mark is revisioned with the category, so what counted as a donation in 2026 still reads that way in 2036.

2.4. Compliance

The German donation receipt and the access-request document live in a compliance module you can replace. What it needs from you is the handful of facts a statutory form names and the contact model does not carry:

`tribemanager.compliance.issuer.address`	The organization's postal address
`tribemanager.compliance.issuer.tax-office`	The tax office that granted the charitable status
`tribemanager.compliance.issuer.exemption-notice`	Reference and date of the exemption notice
`tribemanager.compliance.issuer.signatory`	Who signs - a name and a role

Leaving one empty is allowed. It appears on the document as a visible gap, which somebody can see and fill in; a receipt that quietly omitted the tax office would be worse.

Advanced configuration for Tasks, Inventory and Documents is added here as it lands.

3. Expert configuration

Expert-mode administration: the complete configuration surface - every field type, category dimension, permission grant, fee rule, and module option the platform can express. Maximum control, maximum complexity.

3.1. Contacts

Expert mode exposes the full contact field configuration:

- **Every field type**, including single-select **enum** fields with a defined set of option keys.
- **Kind scoping** - limit a field to individual, organization, or location contacts.
- All the advanced actions (add, edit, deactivate, delete).

Shipped default fields can be deactivated but not deleted (they would only be recreated); fields you add can be deleted outright.

3.2. Relationship types

Expert mode hands over the **relationship vocabulary** itself. Under **Relationship types** you decide which kinds of connection between your organization and its contacts exist - member, supplier, sponsor, whatever your organization actually has - which contact kinds each applies to, and whether it carries the member status lifecycle (applicant, active, passive, honorary, resigned).

The name is the key: writing a name that already exists redefines that type, a new name adds one. There is no delete - a type contacts were connected with is part of the record, and advanced mode simply uses the shipped vocabulary as it is.

3.3. A note on modes

A complexity mode changes what is **shown**, never what is **stored**. Raising a module to expert exposes more configuration; lowering it again hides that configuration and the fields it created, and deletes nothing. So there is no risk in trying a higher mode, and no migration when you leave it.

Expert configuration for the other modules is added here as it lands. The per-module scope is settled when each module is refined.