Auctio

INSTALLATION AND OPERATIONS MANUAL

The operator of the service

Installing, configuring, opening the accounts, keeping it running: the service, the providers, the checks, the emergencies.

Auctio — encrypted deposit of technical bids beyond Sintel's limits · https://auctio.it · manual version: September 2026 (software 0.1, WP17) · ThinkDifferent S.r.l.

Contents

1. What runs where
2. Requirements and external providers
3. Installation from zero
4. Configuration
5. The inbound proxy
6. Accounts
7. Tenders
8. The manager's area
9. Endpoints
10. Operations: what runs by itself, what to check
11. Backup, drill, restore
12. Upgrade
13. Emergencies
14. Never
15. The command line

1. What runs where

Auctio is one Go binary (`auctio`) that serves the pages and the API, runs the scheduler of the periodic jobs and applies the schema migrations at start. In production it runs as a **pod** of two containers (rootless podman, systemd quadlets): the service (`auctio`, image `localhost/auctio`, unprivileged user, read-only filesystem) and PostgreSQL 16 (`auctio-db`, image pinned by digest). Only the service's port leaves the pod, on loopback (`127.0.0.1:8630`); nginx in front terminates TLS.

```
/var/auctio/
  auctio.env        the service's environment                       0600
  auctio-db.env     PostgreSQL's environment                        0600
  secrets/          pgpass, db.password, smtp.password, tsa-chain.pem 0700, files 0600
  data/             keys/, log/ (the log), blobs/ (encrypted deposits) 0700
  pg/               PostgreSQL's data                               0700
  backup/           the nightly archives, the last fourteen         0700
~/.config/containers/systemd/
  auctio.pod  auctio-db.container  auctio.container
```

What must outlive everything is under `data/`: the keys (`keys/seal.pem`, the seal; `keys/secrets.key`, the key of the secrets at rest), the transparency log (`log/`, tiles and checkpoint, with the origin pinned in `log/origin`) and the encrypted deposits (`blobs/`). The database holds accounts, tenders, deposits, receipts, records, notifications, the hash-chained system log.

2. Requirements and external providers

WHAT	FOR	NOTES
A Linux host with podman ≥ 4.4 (rootless) and systemd; 2 CPUs, 4 GB, disk for the deposits	the pod	SELinux fine (labels are in the units); a dedicated user with `loginctl enable-linger`
nginx (or Plesk) with a TLS certificate	the inbound proxy	chapter 5
RFC 3161 timestamping authority	the timestamp on receipts and records	for full legal value: an eIDAS-qualified provider (Aruba, InfoCert, Namirial…); URL, credentials, certificate chain as PEM. In production the service refuses to start without (`AUCTIO_TIMESTAMPING=rfc3161`)
Anchoring	making the log unchangeable	OpenTimestamps (Bitcoin) with no account: the public calendars; optionally an EVM chain with a wallet (`docs/ANCHORING.md`)
E-mail (SMTP with TLS)	confirmations, codes, notices	a dedicated mailbox (e.g. `noreply@auctio.it`); optionally a **PEC** mailbox for the legal notices (download, closure, destruction)
Access to VIES	the VAT check at registration	outbound to `ec.europa.eu`
A build machine (Git, Go, docker or podman)	building and verifying the image	never the production host
A password manager off the host	the backup identity, the providers' credentials	chapter 11

3. Installation from zero

The full procedure is `docs/RUNBOOK.md` §2; here the order and the points that cannot be got wrong.

1. **Build and verify the image**
   on the build machine, at the tagged version you mean to install:

   ```
   git clone git@git.nexlab.net:nexlab/auctio.git && cd auctio && git checkout wp-17
   deploy/build.sh --verify          # builds, saves dist/auctio-image.tar, compares
   the binary
   scp dist/auctio-image.tar dist/auctio-image.sha256 user@host:
   ```

   `--verify` rebuilds the binary locally and compares it byte for byte with the one in the image: `VERIFIED`, or the build stops. Note the digest.

2. **Generate the backup identity**

, on your laptop, never on the host: `auctio backup keygen > auctio-backup.key`. The file goes into the password manager and a printed copy in the safe; the public part (`age1…`, printed on stderr) goes into `AUCTIO_BACKUP_RECIPIENTS`. Lose the identity and every archive is unreadable.

3. **On the host, as root, once:**
the dedicated user with linger, the directories:

```
useradd -m -s /bin/bash auctio && loginctl enable-linger auctio
mkdir -p /var/auctio/{data,pg,backup,secrets} && chown -R auctio:auctio /var/aucti
o && chmod 700 /var/auctio/*
```

4. **As the service user:**
load the image, write the environment and the secrets, hand the directories to the image's unprivileged user:

```
sha256sum -c auctio-image.sha256 && podman load -i auctio-image.tar
podman tag localhost/auctio:<version> localhost/auctio:latest
cp deploy/auctio.env.example /var/auctio/auctio.env && chmod 600 /var/auctio/aucti
o.env   # every CHANGE-ME
printf 'POSTGRES_USER=auctio\nPOSTGRES_PASSWORD_FILE=/secrets/db.password\nPOSTGRE
S_DB=auctio\n' > /var/auctio/auctio-db.env
cd /var/auctio/secrets
head -c 24 /dev/urandom | base64 | tr -d '\n' > db.password
printf '127.0.0.1:5432:auctio:auctio:%s\n' "$(cat db.password)" > pgpass
# smtp.password, pec.password, tsa-chain.pem (and the TSA credentials) as the prov
iders give them
chmod 600 /var/auctio/secrets/*
podman unshare chown -R 65532:65532 /var/auctio/data /var/auctio/backup /var/aucti
o/secrets
```

5. **The keys, before the first start.**
In production the service never invents a seal (a new one would orphan every receipt):

```
alias auctio-run='podman run --rm -v /var/auctio/data:/data:z -v /var/auctio/secre
ts:/secrets:ro,z --env-file /var/auctio/auctio.env'
auctio-run localhost/auctio:latest keys init
auctio-run localhost/auctio:latest keys show      # the seal's KID: note it
```

6. **The units, and the first start:**

```
mkdir -p ~/.config/containers/systemd
cp deploy/auctio.pod deploy/*.container ~/.config/containers/systemd/
systemctl --user daemon-reload && systemctl --user start auctio-pod
systemctl --user status auctio-db auctio      # two green, "healthy"
podman logs auctio | head -20
```

The service applies the schema, starts the scheduler and states `production=true fake_adapters=null` in its first log line: if it lists *fake* adapters, the environment is wrong and the service refused to start.

7 **The first manager:**

```
podman exec auctio /auctio user add --role manager --login <name> --email <address
> --password-file /secrets/manager.initial
```

(the password in a file under `secrets/`, written with `podman unshare`; delete it after the first sign-in). At first sign-in the manager sets up the authenticator app.

8 **The proxy**
(chapter 5), then the proof: `curl https://<domain>/healthz` → `{"database":"ok","status":"OK"}`; a full round as in the contracting authority's manual, with a test tender due in a few minutes; the receipt verified with `auctio-verify`.

---

4. Configuration

Everything is environment (`/var/auctio/auctio.env`); secrets are files under `/secrets`, never values in the environment. The full list with defaults is `docs/CONFIGURATION.md`; here what a production must decide.

VARIABLE	PRODUCTION VALUE	WHY
`AUCTIO_PRODUCTION`	`1`	turns on the guard: refuses fake adapters, requires public URL, contacts, privacy notice, log origin, a real TSA
`AUCTIO_MIGRATE_AT_START`, `AUCTIO_WORKER_INLINE`	`1`, `1`	one process: schema applied at start, scheduler inside the service
`AUCTIO_DATABASE_URL`	`postgres://auctio@127.0.0.1:5432/auctio?sslmode=disable&passfile=/secrets/pgpass`	the password from a file; the database is in the same pod
`AUCTIO_PUBLIC_URL`	`https://auctio.it`	the address in every link mailed and printed
`AUCTIO_LOG_ORIGIN`	`auctio.it/log/v1`	the log's name in every checkpoint, **forever**: pinned at first start, the service refuses to change it
`AUCTIO_OPERATOR_NAME`, `_CONTACT`, `_EMAIL`, `_LEGAL`, `AUCTIO_PRIVACY_URL`	company name; contacts; the address for contracting authorities; registered office, VAT, PEC; `https://auctio.it/privacy`	every page's footer, the documents' author, the privacy notice
`AUCTIO_TIMESTAMPING=rfc3161`, `AUCTIO_TSA_URL`, `AUCTIO_TSA_USERNAME`, `AUCTIO_TSA_PASSWORD_FILE`, `AUCTIO_TSA_CA_BUNDLE`, `AUCTIO_TSA_POLICY`	as the provider gives them; the chain in `/secrets/tsa-chain.pem`	the timestamp; `auctio timestamp probe` tries the endpoint
`AUCTIO_ANCHORING`	`ots` (or `ots,evm`)	the anchoring of tree heads
`AUCTIO_NOTIFICATIONS=smtp`, `AUCTIO_SMTP_HOST`, `_PORT`, `_TLS`,	the service's mailbox; `tls` on 465 or `starttls` on 587	mail; without PEC the legal notices leave

`_FROM`, `_USERNAME`, `_PASSWORD_FILE`; `AUCTIO_PEC_*`		through the e-mail mailbox
`AUCTIO_VAT_CHECK`	`vies`	the VAT number checked at registration
`AUCTIO_TRUSTED_PROXIES`	the address nginx connects from, as the first log line shows it	only from there is `X-Forwarded-For` believed; behind pasta the host's loopback appears as the host's public address
`AUCTIO_BACKUP_RECIPIENTS`, `AUCTIO_BACKUP_DIR=/backup`, `AUCTIO_BACKUP_KEEP`	the age public keys; 14	the nightly archive; without recipients the job fails every night, on purpose
`AUCTIO_WORKER_NIGHT`, `TZ`	`03:30`, `Europe/Rome`	the hour of the nightly jobs; the dates on pages and PDFs
`AUCTIO_LANGUAGE`	`it`	pages, messages, PDFs

Limits and cadences with a sensible default, to touch only with a reason: `AUCTIO_CHUNK_DEFAULT` (chunk size), `AUCTIO_SERVE_MAX`, `AUCTIO_DISK_RESERVE` (free space under which uploads answer 507), `AUCTIO_UPLOAD_EXPIRY` (72 h), `AUCTIO_SESSION_TTL` (12 h) and `_IDLE` (30 min), `AUCTIO_STH_INTERVAL` (10 min), `AUCTIO_CHECKPOINT_INTERVAL` (10 s), the jobs' intervals `AUCTIO_JOB_*_INTERVAL`.

> **Three things never change after the first start:** `AUCTIO_LOG_ORIGIN`, the `data/log/` directory (neither moved nor renamed), the seal except through `auctio keys rotate`.

5. The inbound proxy

nginx terminates TLS and forwards to `127.0.0.1:8630`. The shape of the block is `deploy/auctio.it/zz020_auctio.conf` (the reference installation, on Plesk): `proxy_http_version 1.1`, the headers `Host`, `X-Real-IP`, `X-Forwarded-For`, `X-Forwarded-Proto`, **no** `X-Forwarded-Prefix` (Auctio accepts no base path), `client_max_body_size 64m` (the browser's PATCH reaches 32 MiB at most),

`proxy_request_buffering off` and `proxy_buffering off` (a 600 MB deposit must not touch nginx's disk), `proxy_read_timeout 600s`. HTTP → HTTPS, HSTS.

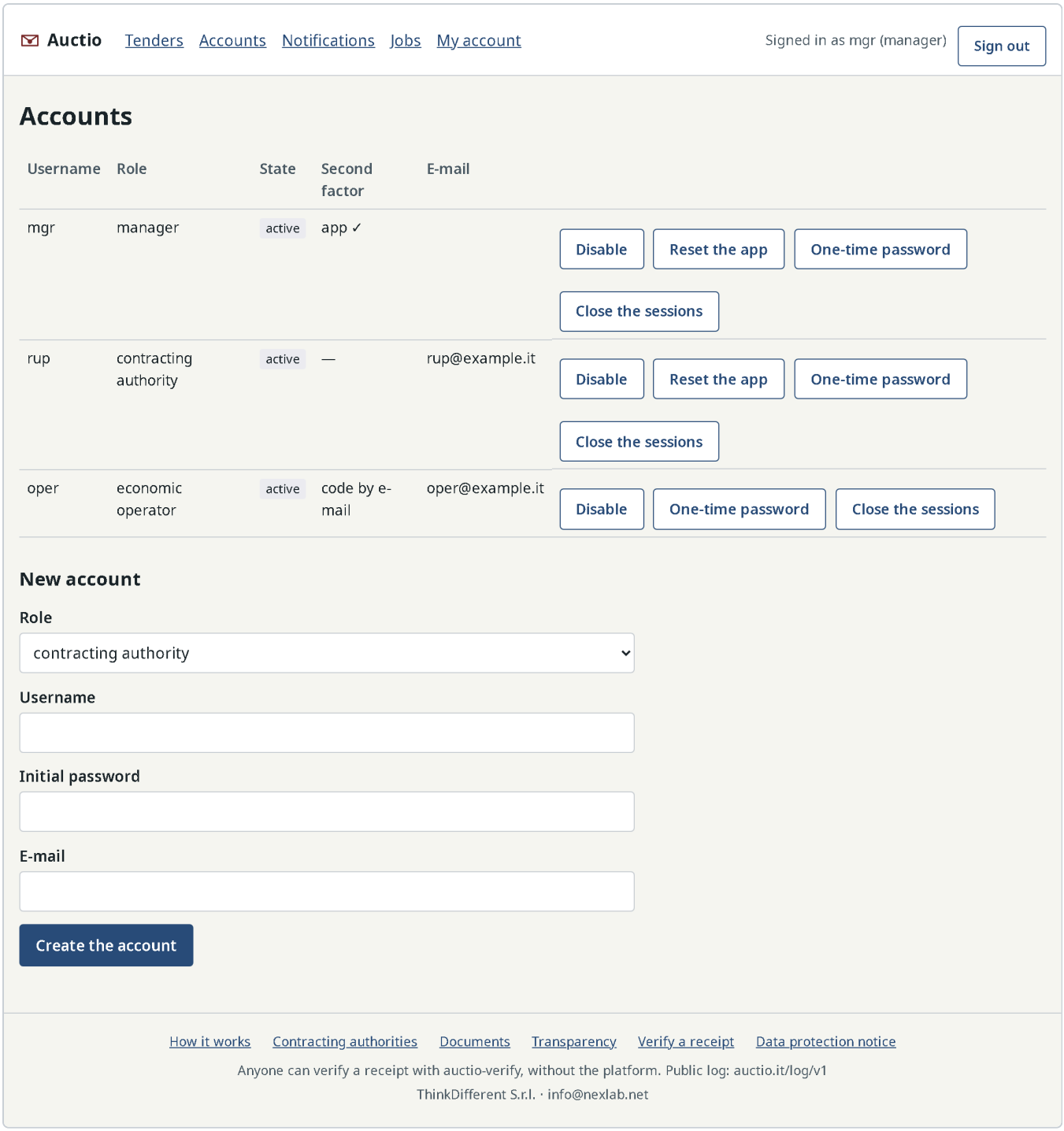
On a Plesk host, a file in `/etc/nginx/conf.d/` with a name sorting *after* `zz010_psa_nginx.conf` (for instance `zz020_auctio.conf`): included before, it would become the address's default server and capture the panel. The Let's Encrypt certificate through `acme.sh` and the HTTP challenge on `/.well-known/acme-challenge/`, renewed by cron with an nginx reload.

After the first start read in the log the line of a request that came through nginx: the `ip` it shows is the one for `AUCTIO_TRUSTED_PROXIES`. Until you set it, the service logs the proxy's address instead of the client's and warns in the log once a minute.

6. Accounts

Three roles. An account has one role.

ROLE	CREATED BY	SECOND FACTOR	CAN
manager	the command line (the first), then the "Accounts" page	authenticator app (TOTP), set up at first sign-in	tenders, accounts, notifications, jobs; never read a deposit
contracting authority	the manager, from the "Accounts" page or the command line, after checking the body	authenticator app	key and opening of the assigned tenders
economic operator	self-registration (`/register`), with e-mail confirmation and VAT check	e-mail code at every sign-in	deposit in open tenders



"Accounts": creation, deactivation, app reset, one-time password, closing sessions.

Creating a contracting authority

From the "Accounts" page: role *contracting authority*, username, e-mail, the initial password you choose. Hand it over through a channel other than the one you gave the username on (phone, PEC). Or from the terminal: `podman exec auctio /auctio user add --role authority --login <name> --email <address> --password-file /secrets/<file>`.

The recurring situations

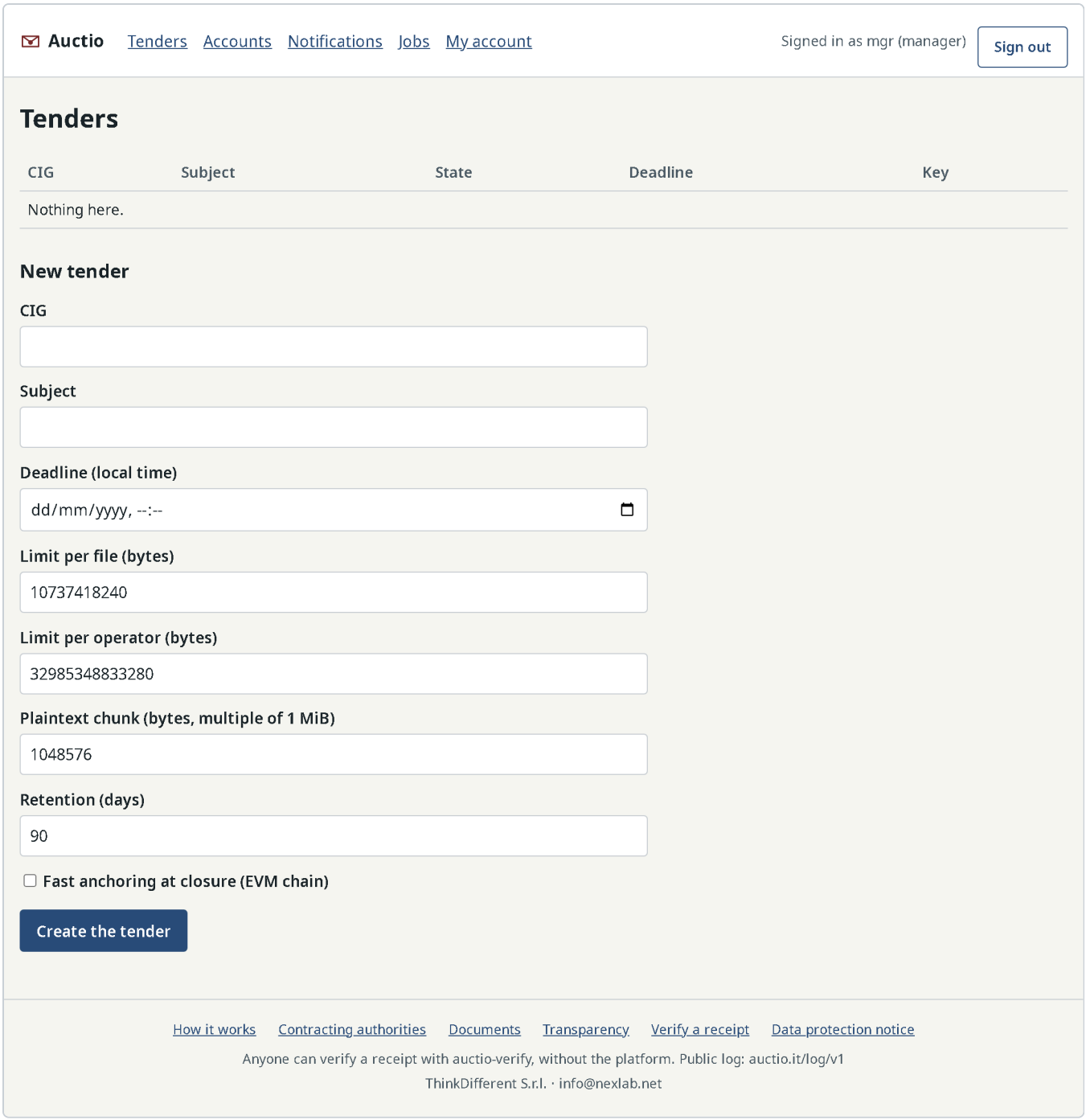
- **lost phone** (manager or contracting authority): "Reset the app" — the account sets the authenticator up again at the next sign-in; first, make sure who is asking;

- **forgotten password**: operators use "Forgot your password?"; for managers and contracting authorities "One-time password", printed once, with the sessions closed;
- **VAT number unverified** (VIES was down): `auctio user verify-vat LOGIN` retries; `auctio user activate LOGIN` confirms by hand a registration whose e-mail does not arrive;
- **a person leaves**: "Disable" closes the account and every session; their deposits and tenders remain.

---

7. Tenders

The manager creates the tender at the contracting authority's request: CIG, subject, deadline (day, hour and minute, Italian time — Sintel's), limits per file and per operator, retention days after the opening, fast anchoring (EVM) of the closure if wanted.



"Tenders": the list and the creation form.

1. **Create**
   the tender: it is born as a *draft*.
2. **Assign**
   the contracting authority (one or more) from the tender's page.
3. **Wait for the key.**
   The contracting authority generates and publishes the key; the tender's page shows the fingerprint when there is one.
4. **Open**

the tender ("Open the tender"): from now on it accepts deposits and the key can no longer change. Give the contracting authority the public page's address (`/t/<identifier>`).

5. **Do nothing**
   until the deadline: the closure is automatic, to the minute, with record, timestamp, log entry and notices. If the contracting authority extends, "Move the deadline"; if it cancels, "Cancel the tender" (the deposits are destroyed).

6. **After the opening**
   by the contracting authority, retention runs and destruction happens by itself at its end. "Declare the opening complete" exists here too, for a contracting authority that did not.

A1B2C3D4E5 — Servizio di progettazione esecutiva — lotto 2

draft · /t/74b1905f2710f0405a3bf154bd335ef5

Deadline	19/09/2026 19:39 CEST (`2026-09-19T17:39:00Z`)
Fingerprint of the tender's public key	The tender key is not published yet.
Maximum file size	10.00 GiB
Maximum volume per operator	30.0 TiB
Chunk size	1.0 MiB
Retention after the opening	90 days

Open the tender | Cancel the tender

Save

CIG

A1B2C3D4E5

Subject

Servizio di progettazione esecutiva — lotto 2

Deadline (local time)

19/09/2026, 19:39

Limit per file (bytes)

10737418240

Limit per operator (bytes)

32985348833280

Plaintext chunk (bytes, multiple of 1 MiB)

1048576

Retention (days)

90

☐ Fast anchoring at closure (EVM chain)

Save

Contracting authority

- rup Remove

Deposits

Identifier	Operator	State	Size	Completed on
Nothing here.				

System log of the tender

When	Event	Actor
19/09/2026 19:34 CEST	`tender.created` `74b1905f2710f0405a3bf154bd335ef5`	`f677e1bf539156614449469dce93561b`



The tender: state, assigned contracting authorities, deposits, the tender's system log, the actions.

A1B2C3D4E5 — Servizio di progettazione esecutiva — lotto 2

opening · /t/74b1905f2710f0405a3bf154bd335ef5

Deadline	19/09/2026 19:39 CEST (`2026-09-19T17:39:00Z`)
Fingerprint of the tender's public key	`60cd6420fd61cccc788c80e75650e50e2ef241f8c05ff9b99eb8967a260f1d0a`
Maximum file size	10.00 GiB
Maximum volume per operator	30.0 TiB
Chunk size	1.0 MiB
Retention after the opening	90 days
Closed on	19/09/2026 19:39 CEST

Declare the opening completed

Contracting authority

- rup Remove

Deposits

Identifier	Operator	State	Size	Completed on
`8035e7d498c4ea51065386bd769dce7c`	`ed8890319567e64776b99250483ab909`	completed	6.0 MiB	19/09/2026 19:34 CEST

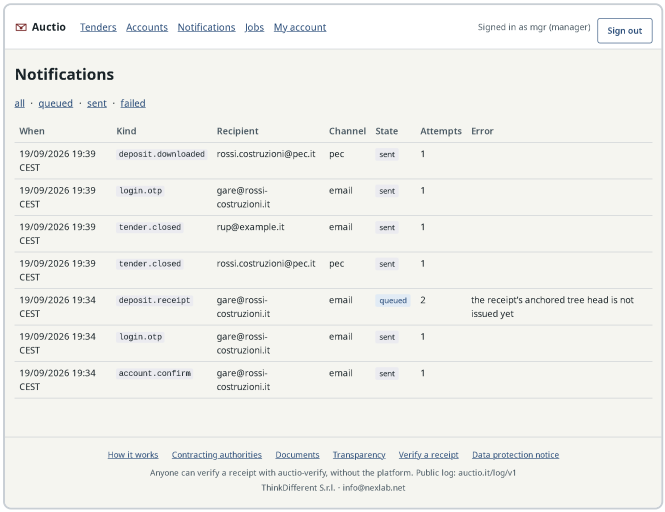
System log of the tender

When	Event	Actor
19/09/2026 19:34 CEST	`tender.created` `74b1905f2710f0405a3bf154bd335ef5`	`f677e1bf539156614449469dce93561b`
19/09/2026 19:34 CEST	`tender.authority_assigned` `a845d7bfea5335beda8fe09452e90a67`	`f677e1bf539156614449469dce93561b`
19/09/2026 19:34 CEST	`tender.keys_published` `74b1905f2710f0405a3bf154bd335ef5`	`a845d7bfea5335beda8fe09452e90a67`
19/09/2026 19:34 CEST	`tender.opened` `74b1905f2710f0405a3bf154bd335ef5`	`f677e1bf539156614449469dce93561b`
19/09/2026 19:34 CEST	`deposit.started` `8035e7d498c4ea51065386bd769dce7c`	`ed8890319567e64776b99250483ab909`
19/09/2026 19:34 CEST	`deposit.chunk_received` `8035e7d498c4ea51065386bd769dce7c`	`ed8890319567e64776b99250483ab909`
19/09/2026 19:34 CEST	`deposit.chunk_received` `8035e7d498c4ea51065386bd769dce7c`	`ed8890319567e64776b99250483ab909`
19/09/2026 19:34 CEST	`deposit.completed` `8035e7d498c4ea51065386bd769dce7c`	`ed8890319567e64776b99250483ab909`
19/09/2026 19:34 CEST	`timestamp.obtained` `8035e7d498c4ea51065386bd769dce7c`	`system`
19/09/2026 19:34 CEST	`receipt.issued` `8035e7d498c4ea51065386bd769dce7c`	`system`
19/09/2026 19:39 CEST	`tender.closed` `74b1905f2710f0405a3bf154bd335ef5`	`cli`
19/09/2026 19:39 CEST	`closure.issued` `74b1905f2710f0405a3bf154bd335ef5`	`system`
19/09/2026 19:39 CEST	`deposit.downloaded` `8035e7d498c4ea51065386bd769dce7c`	`a845d7bfea5335beda8fe09452e90a67`
19/09/2026 19:39 CEST	`chunk.served` `8035e7d498c4ea51065386bd769dce7c`	`a845d7bfea5335beda8fe09452e90a67`
19/09/2026 19:39 CEST	`opening.outcome` `8035e7d498c4ea51065386bd769dce7c`	`a845d7bfea5335beda8fe09452e90a67`

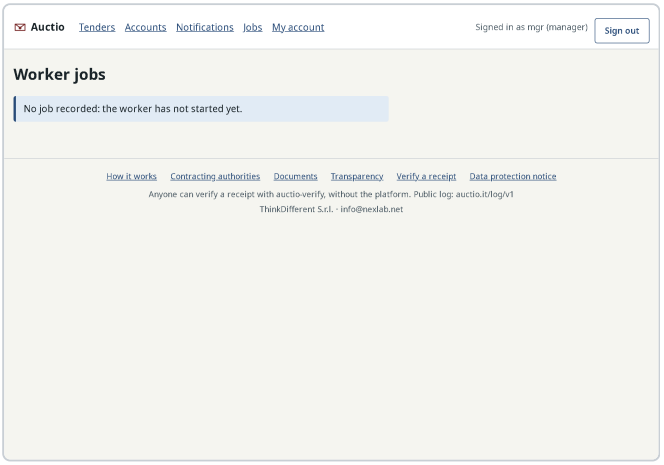
The closed tender: the record, the deposits listed.

8. The manager's area

- **Tenders** and each tender's page: chapter 7; at the bottom, the tender's system log (who did what, when).
- **Accounts**: chapter 6.
- **Notifications**: the mail outbox — queued, sent, failed — with "Retry". A message failing repeatedly is a mail provider problem or a wrong address.
- **Jobs**: the last run of every periodic job with outcome and summary. The first page to look at when something is off.



"Notifications": the outbox.



"Jobs": every job with outcome and summary.

9. Endpoints

Everything goes through the same service, on the same port. Pages and API share session, authorization and the CSRF rule; the full route × role matrix is `docs/AUTHORIZATION.md`, generated by a test.

PATH	WHO	WHAT
`/`, `/come-funziona`, `/stazioni-appaltanti`, `/trasparenza`, `/privacy`, `/documenti`	everyone	the site
`/t/{id}`	everyone	a tender's public page; after the closure, the record
`/login`, `/register`, `/confirm`, `/reset`, `/account`	everyone / session	sign-in, registration, confirmation, reset, account
`/me`, `/t/{id}/deposit`	operator	the tenders, the deposit
`/authority`, `/t/{id}/key`, `/t/{id}/open`	contracting authority	the tenders, the key, the opening
`/manager/tenders`, `/manager/accounts`, `/manager/notifications`, `/manager/jobs`	manager	the manager's area
`/verify/`	everyone	the browser verifier (WebAssembly)
`/api/…`	per role	the JSON API (`docs/API.md`): session, accounts, tenders, deposits (tus for the upload), receipts and records as JSON and PDF
`/api/status`	everyone	version, schema, the client modules' digests
`/healthz`	everyone (nginx, systemd)	`{"database":"ok","status":"OK"}` or 503; the container's health check
`/log/checkpoint`, `/log/tile/…`, `/log/keys`, `/log/sth`, `/log/sth/{n}`, `/log/proof/inclusion/{i}`, `/log/proof/consistency/{a}/{b}`	everyone	the transparency log in the tlog-tiles format, the keys, the anchored tree heads, the proofs
`127.0.0.1:8631/healthz` (inside the pod)	the manager	the scheduler's health, with every job's state; `auctio worker health`

10. Operations: what runs by itself, what to check

The scheduler inside the service runs the jobs below; each exists as a command too (`docs/OPERATIONS.md`).

JOB	WHEN	DOES
`tender.close`	at the earliest deadline, to the second	closes the tenders due: record, timestamp, tree head, notices
`notify.drain`	every minute	sends the queued mail
`receipt.reissue`	every 5 min	issues the receipts a TSA outage deferred
`sth.issue`	every 10 min when the log grew	enters the system log's head into the transparency log, issues a timestamped, anchored tree head
`anchor.upgrade`	every 10 min	confirms pending anchors
`notify.remind`, `deposit.expire`, `tender.destroy`	hourly	deadline reminders, deletion of unfinished uploads (72 h), destruction after retention
`session.purge`, `deposit.scrub`, `integrity.verify`, `backup.nightly`	at night (`AUCTIO_WORKER_NIGHT`)	cleanup; every deposit's digest recomputed; the system log and the transparency log checked against the last anchored head; the encrypted archive

Every day (one minute)

```
podman exec auctio /auctio worker status     # every job ok
podman exec auctio /auctio notify list failed
podman exec auctio /auctio backup list       # a new archive every night
df -h /var/auctio
```

Every week

On the build machine `make audit` (known vulnerabilities, SBOM, newer modules) and `deploy/pin.sh` (a `MOVED` line means a base image has moved: read its notes, update the pin, rebuild with `--verify`, install at the next release). On the host: that the nightly rsync of `/var/auctio/data/blobs` and `/var/auctio/backup` ran.

Once a year, or when a person who could read the file leaves

`auctio keys rotate` retires the current seal and creates the next; retired seals stay for verification and `/log/keys` lists them all. The secrets key is never rotated.

11. Backup, drill, restore

Every night, after the integrity check, the service writes `/var/auctio/backup/auctio-backup-<UTC instant>.tar.age`: an archive encrypted to `AUCTIO_BACKUP_RECIPIENTS` with every table of the database (copied in one transaction), the keys, the transparency log, the origin. **The encrypted deposits are not in the archive**: they are immutable and large; the host's nightly rsync must carry `/var/auctio/data/blobs` and `/var/auctio/backup` out (the files belong to a subordinate uid: the rsync runs as root, or through `podman unshare`).

> The archive is unreadable without the identity (`auctio-backup.key`): a copy of the backup directory reveals nothing; losing the identity loses every archive. The identity is never on the host, nor where the archives are.

The drill, before every release

On a machine that is not the host: an empty PostgreSQL, an empty data directory with the blobs copied in, the identity, the newest archive:

```
podman run -d --name drill-db -e POSTGRES_USER=auctio -e POSTGRES_PASSWORD=drill -e POSTGRES
_DB=auctio -p 127.0.0.1:5433:5432 docker.io/library/postgres:16
mkdir -p /tmp/drill/data && rsync -a host:/var/auctio/data/blobs /tmp/drill/data/
AUCTIO_DATABASE_URL=postgres://auctio:drill@127.0.0.1:5433/auctio AUCTIO_DATA_DIR=/tmp/dril
l/data \
  auctio restore auctio-backup-<instant>.tar.age --identity auctio-backup.key
```

`restore` loads the archive into an empty database and an empty directory, then verifies the system log, the transparency log and, with the blobs, every deposit: a clean outcome is the proof the backup is good. Note date and outcome.

The restore after a loss

The same procedure on the rebuilt host (chapter 3 up to the units, without `keys init`: the keys are in the archive), then the pod. The service comes back with the same seal, the same log, the same origin: receipts and records issued before remain verifiable. `docs/RUNBOOK.md` §5 in detail.

---

12. Upgrade

```
# build machine
git checkout <tag> && deploy/build.sh --verify && scp dist/auctio-image.* user@host:
# host
sha256sum -c auctio-image.sha256 && podman load -i auctio-image.tar
podman tag localhost/auctio:<new> localhost/auctio:latest
systemctl --user restart auctio-pod
systemctl --user status auctio-db auctio
```

The service applies what the schema gained (nothing, if it did not change) and restarts. Read the release notes first: they say whether the schema moves, what the environment gains, whether the units changed (then copy them again and `daemon-reload`). Upgrade at night: closures happen at deadlines. **Rollback**: `podman tag localhost/auctio:<previous> localhost/auctio:latest` and restart; if the schema had moved, `migrate down` once per migration added, then the restart.

13. Emergencies

SYMPTOM	LIKELY CAUSE	WHAT TO DO
Uploads answer 507	free space under `AUCTIO_DISK_RESERVE`	free: journal (`journalctl --user --vacuum-time=30d`), old images (`podman image prune`), archives beyond the fourteen if the rsync has them. **Never** under `data/`
Receipts show as "deferred"	the TSA does not answer	`auctio timestamp probe` ; when it is back, `receipt.reissue` issues them; if the TSA's CA changed, add the new chain to `tsa-chain.pem`
Anchors pending for hours	the OpenTimestamps calendars or the EVM chain unreachable	they confirm by themselves when back; `auctio anchor status`
Mail queued or failed	the SMTP provider, or expired credentials	`auctio notify probe <address>` , then "Retry" from the page or `auctio notify retry ID`
`deposit.scrub` fails	a blob damaged or deleted	an incident: the job names the deposit; restore the blob from the rsync; tell the contracting authority if the tender is open
`integrity.verify` fails	the system log or the transparency log do not match the last anchored head	a security incident: stop the service, preserve everything, `auctio verify-log` and `auctio log verify` say where; `docs/THREAT-MODEL.md`
The service refuses to start: "schema version mismatch", "seal key", "log origin"	binary and data not consistent	read the message: it is exact; never force; chapter 12 for the rollback
The host's panel or another site vanished behind Auctio	the nginx file is included before the host's	chapter 5: the file name must sort after
Dispute over a deposit	—	everything is verifiable from outside: `auctio-verify receipt --file … --closure …` with `/log/keys` saved; the tender's system log on the manager's page; `docs/VERIFICA.md`

14. Never

- Pull `localhost/auctio:latest` or `postgres:16` from a registry into production: the image is the verified tar, the database is the digest in the unit.
- Two pods on one data directory or one database.
- Edit under `data/` by hand; rename or move `log/` ; change `AUCTIO_LOG_ORIGIN` .
- Keep the backup identity on the host, or with the archives.
- Restore onto anything that is not empty: `restore` refuses; do not make it stop.

- Open a tender before the contracting authority has verified the key file and printed the kit.

15. The command line

`podman exec auctio /auctio <command>`, with the container's environment; every command has `--help`.

```
serve                the service (with AUCTIO_WORKER_INLINE the scheduler too)
migrate up|down|status
verify-log [--file export.jsonl]      the system log, recomputed
keys init|show|rotate                 the seal and the secrets key
user add|list|activate|deactivate|reset-totp|reset-password|verify-vat|otp|sessions purge
tender list|show|create|close-due|destroy-due
deposit list TENDER|expire|scrub
log show|sth|verify|reissue           the transparency log
timestamp probe|verify                the RFC 3161 adapter
anchor status|wallet|upgrade|verify|ots info|contract deploy
notify drain|list [STATE]|retry ID|remind|probe ADDRESS
worker [status|run JOB|jobs|health]
backup [list|keygen]                  the encrypted archive, now
restore ARCHIVE --identity FILE
health                                /healthz of the running service
load …                                the load test (never in production)
version
```

The external verification tool, `auctio-verify`, ships with every release for Linux, macOS and Windows with the `SHA256SUMS` file; the guide is `docs/VERIFICA.md`.