Townlight Notice Manual

Windows x64 Beta 1 · MSI 1.7.0.1

The signed Windows x64 beta 1.7.0.1 passed the documented fresh Windows Sandbox checks with synthetic data.

- Townlight Notice User Manual
  - Windows staff workflow — 0.3.0
  - Standalone package boundary
  - For Non-Technical Users
  - For IT and Technical Staff
  - Start and Smoke-Check Townlight Notice
  - Architecture

Townlight Notice User Manual

Windows staff workflow — 0.3.0

Use a fresh Windows profile. The approved profile path is `%LOCALAPPDATA%\CivicSuite`. Keep existing profiles and backups.

1. Sign in with an account allowed to use Notice and open Public Notices. Create a draft with title, actual notice text, type, event date, staff-selected calendar lead days and staff-verified statutory basis. Save it. Its deadline workpaper uses the shared Core reminder helper; it does not establish legal sufficiency.
2. Select the saved notice, read the deadline workpaper and source, enter review notes, then record staff review. Approve that saved revision only after the responsible staff member has checked the law, timing, content and channels. Mechanical rule success never removes the staff-review requirement.
3. Perform official publication outside Townlight. Retain the actual proof file (up to 5 MB each and 10 MB total for the current approved content), posting location and actual posting date/time including time zone. A filename, target system name or checklist alone is not posting proof.
4. Create the archive after approval and proof. The archive contains the actual notice, retained proof bytes, review/approval history and a per-file SHA-256/size manifest. Write and verify the ZIP export to preserve a usable file. ZIP entry timestamps are fixed for deterministic packaging; the record contains the recorded posting and audit times.
5. For a Records handoff, enter the intended Records request UUID and select Attach archive to Records. This requires Records authority as well as Notice authority. Townlight retains an exact managed copy, verifies its hash, attaches it through the canonical Records writer and reports the request/archive identity and hash. Repeating the

same handoff does not create a duplicate attachment. Open the request in Records and read back the attached archive. Disclosure still requires Records review/release authority.

Saved notices and history can be reopened from Public Notices. A content edit clears the prior review, approval and current proof selection; historical events/proof/archives remain retained. A stale open revision is rejected so staff must refresh and review the saved content. Missing review, approval or actual proof blocks the corresponding step. Back up/restore the matching complete new-format profile through System Health; the Windows acceptance summary records the installed backup/restore and archive readback checks.

This replaces the disconnected standalone Public Notices controls. The separate Meetings notice/template/proof workflow remains available for meeting-specific records; it does not silently approve a different Notice draft. Staff remain responsible for legal sufficiency, official publication and the official record. No subscriber message or official notice is sent automatically.

Standalone package boundary

The standalone FastAPI routes below remain registry/deadline/proof workpapers and deterministic planning helpers. The Windows authenticated lifecycle above is provided by the suite runtime using Notice's durable workflow store; it is not a claim that every standalone helper performs an archive or Records transfer. Fixed rule packs are review aids and are not administered jurisdiction/effective-date legal rules. Subscriber/channel/accessibility/export helpers retain their documented planning boundaries.

For Non-Technical Users

Townlight Notice helps city staff organize public hearing notices, legal notices, bid notices, vacancy notices, publication deadlines, channel planning notes, accessibility and language-readiness checks, proof requirements, subscriber delivery plans, archive/handoff packets, and export manifests. It can create a sample notice registry stub, build publication deadline reminders, check a draft packet against deterministic staff-review rules, create a staff-editable notice template, retrieve saved registry/deadline/publication-proof workpapers when IT enables persistence, assemble publication-readiness checklists, summarize channel planning, build accessibility/plain-language/translation readiness packets, build subscriber delivery plans, assemble archive/handoff packets, and assemble a notice/records export checklist.

Notice 0.3.0 ships in the Windows x64 beta 1.7.0.1 with Townlight Core 1.3.0.dev2. The Windows acceptance summary covers the installed staff workflow. Standalone deployment is documented separately below.

For IT and Technical Staff

Townlight Notice is a FastAPI Python package pinned to the `townlight-core 1.3.0.dev2` release wheel. The current runtime exposes:

Set `TOWNLIGHT_NOTICE_WORKPAPER_DB_URL` to enable SQLAlchemy-backed notice registry, deadline-plan, and publication-proof records. Leave it unset for deterministic stateless operation: registry and deadline POST requests return transient payloads, persisted GET routes return actionable 503 responses, and publication-proof storage is unavailable. When persistence is

enabled, set `TOWNLIGHT_NOTICE_TRUSTED_WRITE_TOKEN` and send the matching `X-Townlight-Notice-Write-Token` header for durable write routes.

- `GET /`
- `GET /health`
- `GET /townlight_notice`
- `GET /docs`
- `GET /openapi.json`
- `POST /api/v1/townlight_notice/registry`
- `GET /api/v1/townlight_notice/registry/{record_id}`
- `POST /api/v1/townlight_notice/deadlines`
- `GET /api/v1/townlight_notice/deadlines/{plan_id}`
- `POST /api/v1/townlight_notice/rule-check`
- `POST /api/v1/townlight_notice/templates`
- `POST /api/v1/townlight_notice/publication-proof`
- `GET /api/v1/townlight_notice/publication-proof/{proof_id}`
- `POST /api/v1/townlight_notice/publication-check`
- `POST /api/v1/townlight_notice/accessibility-review`
- `POST /api/v1/townlight_notice/channels`
- `POST /api/v1/townlight_notice/subscribers/plan`
- `POST /api/v1/townlight_notice/archive-packet`
- `POST /api/v1/townlight_notice/export`

Start and Smoke-Check Townlight Notice

A fresh user can reach the core stateless API and public sample UI without a database, model server, account, or API key. Start the ASGI app target `townlight_notice.main:app`, then confirm `/health` reports Townlight Notice `0.3.0` and Townlight Core `1.3.0.dev2`.

Minimal smoke-check workflow:

1. Create a transient registry stub with `/api/v1/townlight_notice/registry`.
2. Open `/docs` or `/openapi.json` to inspect accepted fields, then create a deterministic rule check with `/api/v1/townlight_notice/rule-check`. Unsupported notice types return supported choices instead of silently falling back to a generic rule.
3. Open `/townlight_notice` and confirm it is a static public sample with boundary copy.
4. If durable workpapers are enabled, confirm the trusted write token is configured before testing persistence-backed writes.

Run:

```
python -m pip install -e ".[dev]"
python -m pytest -q
bash scripts/verify-release.sh
```

Architecture

```
flowchart LR
  Staff["Clerk / communications / purchasing staff"] --> Townlight
Notice["Townlight Notice"]
  Townlight Notice --> Townlight Core["Townlight Core 1.3.0.dev2"]
  Townlight Notice -. future handoff .-> Townlight Meetings["Townlight
Meetings v0.1.0"]
  Townlight Notice -. future handoff .-> CivicProcure["CivicProcure v0.1.1"]
  Townlight Notice --> Export["Notice and records export checklist"]
```

Townlight Notice depends on Townlight Core. Townlight Core does not depend on Townlight Notice. Townlight Notice v0.3.0 uses the shared Townlight Core notice deadline helper plus deterministic sample notice data, staff-review rule packs, notice templates, accessibility and language-readiness packets, subscriber delivery plans, archive/handoff packets, and optional publication-proof workpapers only; live agenda/procurement handoffs, legal sufficiency decisions, legal advice, official notice publication, publication-system write-back, and production notice-system integrations are future work.