Townlight Access 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 Access User Manual
  - Windows staff workflow — 0.5.0
  - Standalone support service
  - Part 1 — For Civic Staff
  - Part 2 — For IT & Technical Staff

Townlight Access User Manual

Windows staff workflow — 0.5.0

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 Access and enable the Access capability. Open Accessibility and enter the actual source title, text, language and image-description information.
2. Save the review. Reopen it to inspect the source, deterministic findings and any saved local-model advice, model identity and outcome. Advice is not a certified accessibility finding or approval.
3. For plain-language or translation work, enter the actual source and request a draft. The Windows path uses the suite's configured local model. Saved drafts retain source/output, language, model/outcome, staff actor/time and the requirement for qualified human review. An unavailable-model sample is labeled as a sample; it is not a translation of the input.
4. Reopen saved drafts from Access, including after restarting the app. Select Export record on the desired review or draft. The desktop writes a Markdown record and verifies its content hash and companion manifest before reporting completion. The fenced JSON is the complete record, including source, output/advice and relevant provenance; ordinary HTML rendering is unnecessary to read it.
5. To hand the record to Records, open the intended Records request, attach the actual exported Markdown file using Records' native file picker, and read it back from that request. A successful Access export alone does not attach evidence to a request or authorize disclosure. Records permissions and review/release rules still apply.

The complete review export includes reviewed source, findings, saved advice/model/outcome where available, actor/time, relevant audit history and advisory limits. Draft exports include the

retained source/output pair and review status. Back up the complete profile using System Health and restore only a matching new-format backup. The Windows acceptance summary records installed export, restart and recovery results. Real-model rewrite and Spanish drafts passed in a 16 GiB Sandbox; the 8 GiB attempts timed out and saved no draft.

No automated result certifies ADA compliance, creates an accessible/tagged PDF, approves publication or replaces a qualified translator or ADA coordinator.

Standalone support service

The separately run FastAPI service provides deterministic checks and planning helpers. Its Spanish and Vietnamese responses are fixed samples, **not translations of the supplied source**. Its deterministic jargon substitutions are separate from the Windows model workflow. A shared write token proves possession of that token; it does not establish an individual staff identity. Standalone audit records use `shared-token-staff` for that reason.

Standalone saved-review exports now return a complete Markdown artifact with source, findings and relevant audit provenance, plus its SHA-256. The staff page offers that actual artifact for download. Other form, tagged-PDF, ADA and publishing helpers remain checklists. The standalone service does not acquire Windows draft persistence or native individual-account identity merely because it shares this package.

Version 0.5.0 · included in Windows MSI 1.7.0.1 Townlight Access is the accessibility, plain-language, multilingual, and ADA Title II review-support module for Townlight.

This manual has two parts. Read the one that fits your role:

- **Part 1 — For Civic Staff** — clerks, communicators, ADA coordinators. Plain English, no setup.
- **Part 2 — For IT & Technical Staff** — architecture, runtime, data model, security, and operations.

Part 1 — For Civic Staff

What Townlight Access is for

Townlight Access helps your office put out public notices, forms, and documents that **everyone can read** — and keep a clean record proving you checked. It is a staff tool that gives advice and does the busywork; **you and your ADA coordinator always make the final call.**

What you can do with it

You want to…	Townlight Access gives you…
Know if a notice is hard to read	A check that flags missing titles, images without descriptions, and dense wording — each with a specific fix

You want to…	Townlight Access gives you…
Cut the jargon	A plain-language rewrite (e.g. “remit payment prior to the deadline” → “pay before the deadline”)
Reach non-English readers	Windows local-model drafts for qualified review; fixed standalone samples do not translate input
Publish an accessible form or PDF	A checklist of what an accessible form or tagged PDF needs
Handle an ADA Title II review	A step-by-step review-support checklist
Answer a records request	A saved, exportable record of every review, with provenance (individual staff identity in Windows; shared-token identity in the standalone service)

How you’ll use it day to day

The following browser instructions describe the separately deployed standalone service. For the Windows app, use the Windows staff workflow above. The standalone service has two surfaces:

- The **public checker** is for trying things out — it forgets everything when you leave.
- The **staff workspace** is for keeping a permanent, official record you can produce later.

Your IT team will give you the link to each.

1. Open the **public checker**. Paste in a draft notice and run the check. You’ll get a list of plain-language fixes — nothing is saved, so it’s a safe place to experiment.
2. When you’re ready to keep a review on the record, open the **staff workspace** and enter the staff password your IT team gave you (a one-time setup). Save the review — it is saved with a time-stamp and the shared-token-staff audit identity. The shared token does not identify an individual staff member.
3. From the staff workspace, **export a records-ready package** for any saved review — ready to hand to a public-records request.

What Townlight Access will *not* do

Townlight Access is a tool for your staff, **not a rubber stamp.** It does **not**:

- give legal advice,
- certify ADA compliance,
- issue official translations, or
- publish anything on its own.

A qualified human — your staff, an ADA coordinator, a translator, legal counsel — always reviews and approves before anything goes public. Translations it drafts are **starting points for a human translator**, never the final word.

Where it stands today

This is **v0.5.0, an early release** we're being upfront about. It's solid enough to evaluate and pilot, but it is **not a finished 1.0 and not a compliance guarantee.** (An earlier `v1.0.0` was published by mistake; we pulled it back to an honest sub-1.0 label rather than overstate where the product is.)

Part 2 — For IT & Technical Staff

Overview

The standalone Townlight Access service uses Python/FastAPI and the pinned **Townlight Core 1.3.0.dev2** source. Its deterministic checks do not call a model. The Windows app has a separate authenticated native bridge and local-model path for review advice, rewrite and translation drafts. Those drafts require human review.

Deployment architecture

The Windows app uses the authenticated native Access bridge, the shared runtime database, individual staff accounts, durable drafts and verified file exports. Back up the complete profile through System Health. The service diagram and configuration below describe the standalone FastAPI package; they do not describe the native Windows screens or their authentication.

```
flowchart TB
  Resident["Resident<br/>(browser)"] -->|"public, read-only"|
Pub["/townlight_access"]
  Staff["City staff<br/>(browser)"] -->|"WebView2 UI"|
StaffUI["/townlight_access/staff"]

  subgraph Sup["Tauri supervisor — local machine only"]
    CA["Townlight Access<br/>(FastAPI)"]
    Peers["Peer modules<br/>Townlight Meetings · Townlight Code · Townlight
Notice · …"]
    PG[("Bundled PostgreSQL<br/>127.0.0.1:15432")]
  end

  Pub --> CA
  StaffUI --> CA
  CA -->|"DATABASE_URL → townlight_access schema"| PG
  Peers --> PG
  CA -->|"depends on"| Core["Townlight Core 1.3.0.dev2"]
  CA -. "records-export hand-off" .-> Records["Townlight Records"]
  Sup -. "wholesale backup of Data/ (incl. Data/postgres)" .->
Backup[("Backup folder")]
```

Request & trust model

There are two surfaces with deliberately different trust levels:

- **Public** (`/townlight_access` → `POST /api/v1/townlight_access/analyze`): stateless. Anyone can analyze content; **nothing is persisted and no token is required.**
- **Staff** (`/townlight_access/staff` → `POST .../review`, `POST .../reviews/{id}/records-export`): persistent. Every write requires the **trusted-write token**.

```
sequenceDiagram
  participant R as Resident (public)
  participant S as City staff
  participant CA as Townlight Access
  participant DB as PostgreSQL (townlight_access schema)

  R->>CA: POST /analyze (no token)
  CA-->>R: findings + fixes (nothing saved)

  S->>CA: POST /review + X-Townlight-Access-Write-Token
  Note over CA: constant-time token check<br/>403 if invalid · 503 if
unconfigured
  CA->>DB: insert review + audit row (one transaction)
  CA-->>S: saved review_id + findings

  S->>CA: POST /reviews/{id}/records-export + token
  CA->>DB: write audit_events row
  CA-->>S: records-ready export
```

Persistence & configuration

Townlight Access resolves its review store in this order:

1. `TOWNLIGHT_ACCESS_REVIEW_DB_URL` — explicit override (a dev SQLite path or a pre-built Postgres URL).
2. `DATABASE_URL` — the supervisor-injected async URL (`postgresql+asyncpg://…:15432/…`), converted to a synchronous psycopg2 URL. **This applies when the standalone service is supplied that environment variable.**
3. **SQLite dev fallback** — `data/townlight_access-reviews.db` under `TOWNLIGHT_ACCESS_DATA_DIR` (default: `./data`), used only when neither variable is set.

Variable	Purpose
`DATABASE_URL`	Shared Townlight Core PostgreSQL (set by the supervisor) — the production default
`TOWNLIGHT_ACCESS_REVIEW_DB_URL`	Explicit SQLAlchemy URL override
`TOWNLIGHT_ACCESS_DATA_DIR`	Directory for the SQLite dev fallback
`TOWNLIGHT_ACCESS_TRUSTED_WRITE_TOKEN`	**Required** server secret for persistent writes

`townlight_access-db-status` (console script) preflights a database with an explicit URL.

Data model

Three tables live in the `townlight_access` schema (translated to the default schema on SQLite). Schema setup is non-destructive (`CREATE SCHEMA IF NOT EXISTS` + `metadata.create_all`); the applied migration id is `townlight_access-windows-local-state-v1`.

```
erDiagram
  accessibility_review_records {
    string   review_id PK
    string   title
    text     body
    bool     has_alt_text
    string   language
    string   status
    json     findings
    text     disclaimer
    datetime created_at
  }
  audit_events {
    string   event_id PK
    string   action
    string   subject_id "soft ref → review_id"
    string   actor
    datetime created_at
  }
  schema_migrations {
    string   schema_version PK
    datetime applied_at
  }
```

HTTP API

Method & path	Auth	Persists?	Purpose
`GET /`	—	no	Module status and boundaries
`GET /health`	—	no	Package + Townlight Core version
`GET /ready`, `GET /api/v1/townlight_access/readiness`	—	no	Persistence readiness gate
`GET /townlight_access`	—	no	Public accessibility checker (UI)
`GET /townlight_access/staff`	—	no	Staff workspace (UI; never embeds the token)
`POST /api/v1/townlight_access/analyze`	—	no	Stateless accessibility analysis

Method & path	Auth	Persists?	Purpose
`POST /api/v1/townlight_access/review`	**token**	**yes**	Save a review record (+ audit)
`GET /api/v1/townlight_access/reviews`	—	no	List saved reviews
`GET /api/v1/townlight_access/reviews/{id}`	—	no	Retrieve a saved review
`POST /api/v1/townlight_access/reviews/{id}/records-export`	**token**	**yes** (audit)	Records-ready export (+ audit)
`GET /api/v1/townlight_access/integration-contracts`	—	no	Published integration contracts
`POST /api/v1/townlight_access/plain-language`	—	no	Deterministic plain-language rewrite
`POST /api/v1/townlight_access/language-variant`	—	no	Multilingual draft variant (human-review flagged)
`POST /api/v1/townlight_access/forms`	—	no	Accessible form publication checks
`POST /api/v1/townlight_access/publishing-workflow`	—	no	Staff publication workflow steps/blockers
`POST /api/v1/townlight_access/ada-title-ii`	—	no	ADA Title II review-support checklist
`POST /api/v1/townlight_access/tagged-pdf`	—	no	Tagged-PDF heading expectations
`POST /api/v1/townlight_access/export`	—	no	Records-ready export checklist (stateless)

Token-guarded writes require the `X-Townlight-Access-Write-Token` header matching `TOWNLIGHT_ACCESS_TRUSTED_WRITE_TOKEN` (compared in constant time). Missing/invalid → `403`; guard not configured → `503` (fails closed). The server token is never embedded in served HTML — the staff page provides a field where an operator pastes it; it is kept only in the browser session.

Security

- **Authentication boundary:** persistence-write routes require the trusted-write token; read and stateless-analysis routes are open. The desktop supervisor binds the service to the local machine.
- **Audit trail:** every persistent write/export writes an `audit_events` row (action, subject, actor, timestamp). `review.create` is committed atomically with the review record.
- **Determinism:** no model/LLM calls and no outbound network calls; checks are pure functions.
- **Backups:** review and audit data live in the shared Postgres cluster, captured by the supervisor's wholesale `Data/` backup. Durability is covered by the test suite (a Postgres reconnect round-trip for the default store; a SQLite file backup/restore round-trip for the dev fallback).

Operations

Readiness: poll `GET /ready` (or `/api/v1/townlight_access/readiness`) — `ready` when the schema can be created and verified.

Local verification (mirrors CI; the release gate requires a real PostgreSQL):

```
python scripts/install-core-candidate.py
python -m pip install -e ".[dev]"
# Set TOWNLIGHT_ACCESS_POSTGRES_TEST_URL to an owned disposable database.
bash scripts/verify-release.sh
```

Integration

Townlight Access **depends on Townlight Core**; Townlight Core does not depend on Townlight Access. It publishes integration contracts at `/api/v1/townlight_access/integration-contracts`, hands records exports to **Townlight Records**, and provides accessibility support to downstream publishers (zone, plan, permit, inspect, grants, procure).

Release status

`v0.5.0` is an early release. Probe gaps #1–#4 (clean install, staff/public authz, audit logging, backup/restore durability) are closed with evidence (see `PROBE-PROGRESS.md`). Access 0.5.0 is included in Windows MSI 1.7.0.1. See the exact installed acceptance and its limits. The earlier `v1.0.0` tag was published in error and is retained only as historical evidence.