Townlight Records 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 Records Manual
  - Contents
  - Using this manual
  - Beta identity
  - First run
  - Daily workflow
  - Connected workflows
  - Architecture
  - Fees messages notifications and local intake
  - Technical specifications
  - Configuration and administration
  - API and integration examples
  - MCP and automation boundary
  - Security and permissions
  - Backup recovery and maintenance
  - Troubleshooting
  - Verified workflows and limits
  - Terms of service and license
  - Glossary
  - Alphabetical subject index
  - Source and document control

Townlight Records Manual

Townlight Beta 1 · Naming candidate documentation edition 2 November 2026

Contents

- Using this manual
- Beta identity
- First run
- Daily workflow
- Connected workflows
- Architecture

- Fees messages notifications and local intake
- Technical specifications
- Configuration and administration
- API and integration examples
- MCP and automation boundary
- Security and permissions
- Backup recovery and maintenance
- Troubleshooting
- Verified workflows and limits
- Terms of service and license
- Glossary
- Alphabetical subject index
- Source and document control

Using this manual

Townlight Records helps records officers, clerks and authorized reviewers. Records manages request intake, responsive evidence, review and the final response. Material received from another Townlight product is evidence for a request. Records separately controls release decisions, response approval, exports and fulfillment.

Staff should begin with First run and Daily workflow. Administrators should also read Architecture, Technical specifications, Configuration, Security and Recovery. Developers should use the exact API/data dictionary and examples. The table of contents and alphabetical subject index link to the relevant sections in the HTML edition; the PDF preserves headings and searchable text. An editable Word edition is included with the same documentation companion.

Beta identity

Identity	Value
Documentation	Townlight Beta 1 naming, Access and Notice candidate
Intended installer version	1.7.0.1
Product component	townlight-records 1.8.0.dev2
Principal import package	townlight_records
Native desktop source version	1.7.0-1
Candidate hashes	SHA-256 checksums and Windows acceptance summary
Current owned software license	AGPL-3.0-only
Acceptance composition	Fresh Windows x64 profile; inspections-preview; Station configured separately

Use an empty database and fresh Windows profile. The installer refuses legacy profile initialization. The approved profile path is `%LOCALAPPDATA%\CivicSuite`. Keep existing profiles and backups.

First run

1. Download the signed MSI, matching source companion and checksums from the public Windows release.
2. Verify the MSI hash and Authenticode publisher before installing on a fresh Windows profile. See the Windows acceptance summary for the tested setup.
3. Launch Townlight under the intended named Windows account. Choose the storage folders. Select **Townlight Inspections candidate** (`inspections-preview`) for the bounded desktop composition, including Access and Notice. A narrower default selection will not expose all workspaces.
4. Enter the city profile, time zone and contacts. Create the first named local administrator, sign in and choose protected backup storage. **Continue without local AI** is valid for the deterministic workflows.
5. Run Health Check and wait for readiness. Open Townlight. In System Health select **Finish Records Migration** to activate the canonical Python/PostgreSQL runtime. Do not fabricate migration receipts or seed accounts.
6. Create named staff through Settings. Import authorized sources using each product's source library. All examples in this package are synthetic; they are not municipal authority.
7. Provision Station separately if it is part of the beta exercise. Selecting the desktop composition does not start or configure Station. Configure vision explicitly only if automatic photo drafting is selected.

Daily workflow

1. Receive and identify the request

Create the request with its requester, description and recorded deadline basis. Save it and retain the request identifier. Record assignment, clarification and work in the request history.

2. Find responsive evidence

Search only the permitted sources. Inspect the source title, date, citation, visibility and version before attaching responsive material. Open the saved attachment to confirm that it is the intended evidence.

3. Preserve the original

Keep original bytes and their hash. Add a separately reviewed release-ready or redacted copy when needed. Renaming a private file does not remove private content.

4. Record each release decision

Explicitly record release, redact or exempt for the relevant document, with the applicable reason and source. Do not treat an AI suggestion or another product's review as the Records legal decision.

5. Draft and approve

Draft the response and review the selected attachments. Record approval of the exact response text. Editing that text changes its hash and requires a new approval.

6. Export both artifacts

Export the approved response and the release package. If release is refused, inspect the missing decision, approval or source-permission message. Resolve that condition before trying again.

7. Fulfill and retain

Use the explicit fulfillment action only after checking the outgoing artifacts. Retain the response, package and history. A preview, attachment or export by itself does not fulfill the request.

Working safely

Use your own named account and save before leaving a screen. A saved draft is durable; unsaved form text may not be. If a write has an unknown result, reopen its record before repeating it.

Read source jurisdiction, dates, citation, version and visibility. A copied source hash identifies exact bytes, not their legal applicability. If a source changes or permission is removed, resolve the source problem rather than substituting stale cached content.

Keep originals and derivatives separate. Review the intended audience before export. A public summary excludes private fields only through the applicable copy contract and your review. A renamed private packet is still private content.

Drafting, review, approval, attachment, export, fulfillment and publication are separate operations. A Townlight workpaper does not itself adopt law, issue a permit, impose a fine or release a records response. The responsible staff member records the applicable decision.

Connected workflows

Receives selected evidence from Meetings, Code, Zoning, Planning, Permitting, Inspections and Station. Station originals are locked private evidence. A product's public copy remains subject to the Records response workflow.

Direction	Material
Input	A request description and requester, authorized source documents, explicit cross-product evidence selections and a reviewer's decisions.
Output	A retained request history, reviewed attachments, approved response text and a release package after the applicable release checks.

Product boundary

The selected staff workflow is accepted. A resident internet deployment, every jurisdiction's requirements, complete federation and exhaustive redaction/accessibility behavior are not certified by that acceptance. SMTP code exists; logging a message or outbox entry does not establish email delivery.

Architecture



Townlight Records architecture and evidence flow

The drawing shows the principal staff/evidence path. Native commands carry trusted identity to Python business services; PostgreSQL and retained original files hold durable state. A downstream copy does not silently transfer decision authority. Station has its own configured authority/media boundary, database and mutable files. The connected-workflow text supplies the drawing's reading order and accessibility alternative.

Fees messages notifications and local intake

Record a fee line or waiver

Open the saved request's existing fee controls. Enter the description, adopted schedule/source basis and exact amount in dollars and cents. For example, a synthetic line can use description "Synthetic reproduction," basis "Fictional test schedule" and amount `12.50`. Save and reopen the request. Inspect `fee_line_items`, `fee_status` and the resulting estimate. The native line validator accepts positive exact cents; additional fractional cents, negative values and a zero line are refused.

To waive the request fee, record the actual waiver reason and save through the waiver control. Reopen the request and confirm waived status and the zero-dollar waiver explanation. Adding a later line preserves the waiver in the current workflow; do not assume it reinstates a payable balance. These are Records workflow amounts and decisions, not an official permit-fee calculator or payment service.

A trusted native client uses `records_domain_request` with `operation: 'apply'`, the real `request_id`, and `action`/`payload` fields from the fragments below at the same level as `operation` and `request_id`. Examples of command payloads follow; IDs and permissions still come from the signed-in synthetic account.

```
{"action":"add-records-fee-line","payload":{"feeLineDescription":"Synthetic reproduction","feeScheduleBasis":"Fictional test schedule","feeLineAmount":"12.50"}}
```

```
{"action":"waive-records-fee","payload":{"feeWaiverReason":"Synthetic public-interest exercise"}}
```

Use the exact outer apply contract in the API reference rather than sending these command fragments to an arbitrary URL. After either operation, read the request again; a saved amount does not approve or fulfill the response.

Save staff and requester messages

In the saved request, enter a message and use its explicit send/save action. The native `add-records-message` command's payload uses `requestMessageBody`, for example "Synthetic clarification received." Inspect the retained message/timeline and intended audience before repeating the action. A persisted staff message or locally queued notification does not prove an email reached a requester. For a clarification requiring a response, retain the clarification note and keep the request's actual status distinct from completion.

Configure templates and inspect the outbox

Administrators use the existing `/notifications/templates` create/list/update API under the application's real authenticated authority. A template supplies `event_type`, `channel` (default `email`), `subject_template` and `body_template`. Use the event name actually produced by the selected workflow and the variables it supplies; template existence alone does not cause that event to be emitted. Staff inspect `/notifications/log` for recipient, rendered content, status and error information.

The existing delivery function is `app.notifications.smtp_delivery.deliver_queued_notifications(session)`. A technical operator invokes it only in the matching configured application environment, with an application-owned async session. It processes **queued email** entries, commits `sent` or `failed` status, and returns counts. Missing SMTP configuration returns `reason: smtp_not_configured`. Missing rendered subject/body or recipient is a failure, not delivery.

SMTP settings are `SMTP_HOST`, `SMTP_PORT`, `SMTP_USERNAME`, `SMTP_PASSWORD`, `SMTP_FROM_EMAIL` and `SMTP_USE_TLS`. Keep secrets in protected process configuration. Inspect the actual setting names/defaults in that reference before selecting a mail server. This beta has no automatic failed-entry retry or native “Retry email” button: failed rows are not selected by the queued-only delivery loop. Preserve errors and resolve the cause through the owning deployment’s supported process; do not invent a retry or reset database rows by hand. The native notification “sent” acknowledgement is not an SMTP transmission receipt. External mail remains a separately selected deployment configuration, not a requirement to send email during synthetic development.

Exercise existing local public intake and tracking

This is the existing local contract, not a new internet portal deployment. Under the application’s configured portal posture, a synthetic client can submit `POST /desktop/v1/public/requests` with:

```
{"requester":"Synthetic
resident","requesterContact":"synthetic@example.invalid","summary":"Synthetic
minutes request","deadline":""}
```

Retain the returned tracking information privately. `POST /desktop/v1/public/lookup` requires **both** `trackingNumber` and matching `requesterContact`; it returns the permitted public status rather than staff notes. `/desktop/v1/public/messages` additionally requires `publicRequestMessage`. Read the status/response detail and treat a portal-mode or identity refusal as a refusal. Do not remove that gate, expose tracking/contact values in logs or assume that changing `PORTAL_MODE` commissions a public website. Named staff still use their own authenticated workspace.

Source contracts: `app/desktop_bridge/schemas.py`, `app/requests/workflow.py`, notification service/router/SMTP delivery, and the public desktop bridge routes. Full outer command shapes and endpoint authentication remain in the Records API reference.

Technical specifications

Runtime and storage

The accepted package targets **Windows x64 fresh installation**. Its native shell uses **Rust 2021, Tauri 2 and Microsoft WebView2**; browser assets are JavaScript built with **Vite 8.1.6**. This describes the native desktop frontend; Station has a separate React operator, specified below or in its product manual. Canonical business workflows use bundled **CPython 3.13.5, FastAPI, Pydantic, SQLAlchemy and Alembic**, with **PostgreSQL 17 / PostGIS** and managed original files. The installed runtime tested here contains FastAPI 0.162.0. Package minimum Python

requirements can differ from the bundled executable; the metadata reference preserves both. No stack rewrite is implied by the beta label.

The native transport is fixed to loopback `127.0.0.1:15480`; PostgreSQL uses the configured local runtime port (default `15432`). Optional local model service defaults to `127.0.0.1:15434`. Those are deployment defaults, not internet endpoints. The runtime has health, module and readiness endpoints; readiness is stronger than process existence. The task queue is the existing local Core task infrastructure, not an assertion that a separate Redis service is required for this desktop composition.

The default native profile is `%LOCALAPPDATA%\CivicSuite`; the historical directory name remains part of compatibility. Do not rename it to Townlight. A configured `TOWNLIGHT_PROFILE_ROOT` must be absolute. Local configuration and secrets live under that profile; protected managed files and database state must be backed up coherently. Source/runtime identity is distinct from a product's display label.

Data and consistency

Requests; assignments and deadlines; search sessions; evidence attachments and original hashes; release copies; exemption decisions; messages; fee line items; response approvals; export artifacts and audit events.

Versioned saves use expected prior revision identifiers where defined. Reopen and reconcile on conflict. Exact exports and handoffs use source IDs, current visibility, revision and SHA256. Review belongs to the saved revision and does not transfer silently to edited text. Database changes use the owning product's migrations; a request or readiness check is not an authorization to mutate schema.

Capacity and platform statements

This beta has synthetic single-profile workflow evidence. It does not have a published benchmark for concurrent staff capacity, minimum RAM/CPU, large-source limits or a field SLA. Do not turn a developer machine's specification into a supported minimum. The signed MSI is 2,082,697,216 bytes; allow additional disk for extracted runtime, database, originals, exports, media and backups. Select equipment/storage after sizing those actual workloads.

The exact runtime payload lock records pinned files, sizes and hashes. The component metadata records shipped dependency declarations. These references are more precise than a manually copied dependency list. Third-party and historical license notices must accompany their matching payloads.

Configuration and administration

Native deployment configuration

Use Setup, Settings and System Health for supported staff operation. Administrators provision folders, backup storage, named accounts and the city profile. Select `inspections-preview` for the bounded desktop composition, including Access and Notice. Station remains a separate

service boundary. Do not use a standalone product's demo server as a substitute for the installed workspace.

Technical deployments use `TOWNLIGHT_PRODUCT_PROFILE`, `TOWNLIGHT_PROFILE_ROOT`, `DATABASE_URL`, `TOWNLIGHT_RUNTIME_PORT`, `PORTAL_MODE` and the runtime-managed secret files. Treat database URLs and native session proofs as secrets. The installed native domain runtime explicitly refuses non-loopback binding. `PORTAL_MODE=private` is the accepted posture; changing it does not by itself commission an internet portal, authentication provider or reverse proxy.

The isolated documentation exporter sets a fictional profile and unusable synthetic database origin only to read schemas; those values are not deployment configuration. Keep deployment values in the intended process/account scope. An already-running process does not automatically inherit later environment edits.

Communications and review configuration

SMTP settings include host, port, username/password, sender address and TLS choice. Configure and verify actual delivery only for a deployment that selects external mail. Connector host allowlists and encrypted data-source configuration are separate controls. Records fee schedules concern the records workflow; they do not authorize Permitting to calculate an official permit fee. Keep resident portal posture and self-registration separate from named local staff setup.

API and integration examples

Native command example

The following JavaScript belongs in the existing trusted Townlight frontend/development environment with `invoke` imported from `@tauri-apps/api/core`. It is not a command for an arbitrary web page or an instruction for nontechnical staff. IDs must come from records created in an isolated synthetic exercise.

```
import { invoke } from '@tauri-apps/api/core';
const result = await invoke('records_domain_request', {request: {operation:
'view', request_id: requestId}});
console.log(result);
```

Trusted local HTTP example

An administrator/developer can inspect an already-configured isolated runtime from a trusted local process. This PowerShell example reads the signed-in synthetic profile's existing proof into memory; it does not create a user, seed a session, reveal the proof in output or bypass a permission check. Do not use real city data in development examples. A read response does not prove approval or publication.

```
$profilePath = Join-Path $env:LOCALAPPDATA 'CivicSuite'
$sessionPath = Join-Path $profilePath 'config\local-session.json'
$session = Get-Content -LiteralPath $sessionPath -Raw | ConvertFrom-Json
$headers = @{ 'X-Townlight-Session' = $session.session_hash }
```

```
Invoke-RestMethod -Uri 'http://127.0.0.1:15480/desktop/v1/records-state'
-Headers $headers
Remove-Variable session, headers
```

A synthetic write with subsequent readback

This example demonstrates the existing Meetings HTTP contract in an isolated exercise. It is intentionally fictional. It does not start a service. Its authentication setup is included so it can be run independently under the signed-in synthetic account. If a write times out, inspect current records before repeating it.

```
$profilePath = Join-Path $env:LOCALAPPDATA 'CivicSuite'
$sessionPath = Join-Path $profilePath 'config/local-session.json'
$session = Get-Content -LiteralPath $sessionPath -Raw | ConvertFrom-Json
$headers = @{ 'X-Townlight-Session' = $session.session_hash }
$body = @{
  title = 'Synthetic documentation exercise'
  scheduled_start = '2026-11-05T18:00:00-07:00'
  location = 'Synthetic room'
  meeting_type = 'regular'
} | ConvertTo-Json
$meeting = Invoke-RestMethod -Method Post `
  -Uri 'http://127.0.0.1:15480/desktop/v1/meetings/meetings' `
  -Headers $headers -ContentType 'application/json' -Body $body
Invoke-RestMethod `
  -Uri ('http://127.0.0.1:15480/desktop/v1/meetings/meetings/' + $meeting.id)
`
  -Headers $headers
Remove-Variable session, headers
```

This write requires Meetings to be enabled and the signed-in staff role to permit it. It is a cross-suite integration example; for this product's request models, use the complete technical reference below.

Complete technical contract

Read the Records API reference for every documented operation, request field, constraint, response declaration and source supplement. Download the OpenAPI JSON for tools. For Station this is the suite-side schema only; its separate service contract is documented in the API supplement. The reference directory includes configuration/source inventories, dependency declarations, native command contracts, MCP signatures and candidate evidence.

MCP and automation boundary

No direct product MCP server is exposed by the accepted Townlight desktop or Station boundary. Do not configure an invented `townlight mcp` command or assume an AI client has staff permissions. The integration surfaces for this beta are native IPC, the documented HTTP APIs, source bundle imports, exact exports and the existing Station CLI.

The bundled transcription code includes a **TextFlowKit MCP adapter** as a separate source capability. It offers job-based `transcribe_media`, `get_transcript`, `list_jobs`, `get_job_status` and `export_transcript`, and additional tools declared in that adapter. It uses stdio first and optional Streamable HTTP, with the MCP SDK as an optional extra. It is not automatically launched, authenticated as Townlight staff or accepted as a Townlight product MCP deployment. This documentation does not add such a service.

For a transcription client already provisioned separately, an illustrative tool call is:

```
{"name":"get_job_status","arguments":{"job_id":"synthetic-existing-job-id"}}
```

Use a job ID returned by that transcription service. This example does not create a Townlight record or bypass its review/Records handoff. The exact optional adapter tool signatures are supplied in the MCP source reference. Station meeting capture does not become live speech recognition because this optional donor adapter exists.

Security and permissions

Use your own named account and save before leaving a screen. A saved draft is durable; unsaved form text may not be. If a write has an unknown result, reopen its record before repeating it.

Read source jurisdiction, dates, citation, version and visibility. A copied source hash identifies exact bytes, not their legal applicability. If a source changes or permission is removed, resolve the source problem rather than substituting stale cached content.

Keep originals and derivatives separate. Review the intended audience before export. A public summary excludes private fields only through the applicable copy contract and your review. A renamed private packet is still private content.

Drafting, review, approval, attachment, export, fulfillment and publication are separate operations. A Townlight workpaper does not itself adopt law, issue a permit, impose a fine or release a records response. The responsible staff member records the applicable decision.

The native session is validated and mapped to a persisted SQL identity. An editable actor/reviewer label is not authentication. Product/module permissions and source visibility apply at the server boundary. Revoked/unavailable authority fails closed for new Station operations; a previous permission result is not a perpetual authorization.

Use protected local folders, account permissions and credential storage. Never include the native session file, database passwords, provider keys or unredacted support bundles in a public issue or generated public documentation. A local public view is not automatically a secured internet deployment. Keep network exposure and real equipment acceptance specific to the chosen beta site.

Backup recovery and maintenance

Save work and use Core’s canonical **Backup Now** before maintenance. That operation stops writers. Preserve the coherent profile/database/document backup and its receipt, then use **Start** to resume. A source export is not a full backup.

Use **Restore Latest Backup** through the supported controls. If the result says **Restore needs service start**, use **Start**, then **Check**, and reopen a known record. Do not repeat restore merely because services need to start. The fresh-profile acceptance verified that a fulfilled request survived restore and a post-backup marker did not.

Keep Station's dedicated database and media evidence together in its own backup. Core's native-profile backup must not be assumed to include an external Station data directory. Stop writers during a coordinated backup. Preserve failed media attempts and logs.

On failure, record the time, selected composition, component/installer versions and exact message. Check System Health for stopped, starting, ready or failed dependencies. Use the existing Retry control after readiness; do not create duplicate business records to refresh a stale view. Share only a redacted diagnostic bundle. Preserve secrets, original evidence, recovery material and unrelated installations.

Troubleshooting

Symptom	Likely meaning	Next action
Workspace absent	Wrong product composition or unavailable canonical runtime.	Check selected composition; activate through Finish Records Migration; wait for readiness.
Cannot sign in / 401	Missing or expired named account session.	Sign in normally; reopen a Station launch ticket if applicable.
Permission denied / 403	Account, module, source or browser-origin permission.	Ask the authorized administrator to inspect the intended permission; do not invent role headers.
Save conflict / 409	The saved revision, expected source hash or lifecycle changed.	Keep edits, reopen current state and reconcile deliberately.
Invalid input / 422	Required field, format, geometry, source envelope or state is invalid.	Read the field detail and correct the input/source; preserve originals.
Service unavailable / 503	Database, schema, connected source or configuration unavailable.	Inspect System Health and the named dependency. Preserve work; retry only after readiness.
Export refused	Missing exact review, approval, audience or source authorization.	Read the refusal and complete the owning product's explicit decision.
Restore needs service start	The snapshot was restored but services have not resumed.	Use Start, then Check and verify a saved record; do not blindly repeat restore.

Verified workflows and limits

Product	Recorded result	Precisely exercised
Records	PASS	Intake, note, original attachment/readback, separate release copy, explicit release decision, draft/approval, approved response and release package exports, fulfillment/readback and native workspace.

These results come from the identified signed candidate's fresh normal-profile Sandbox exercise: real native IPC/HTTP/PostgreSQL and selected headless controls. All accounts, source bundles and media were synthetic. The guest install/uninstall and recovery passed, and the owned Sandbox stopped. No host installed application or standalone CivicCast was changed. This documentation task reuses those receipts; it does not claim a new software acceptance run.

See the Windows acceptance summary for this MSI. The historical evidence index applies to its original build.

The limits remain: actual city field acceptance and equipment/network commissioning; Station automatic provisioning/SCM commissioning; every UI/role combination; field recognition accuracy; and historical in-place upgrades are not certified. They are not five invented feature requirements. Fresh-install controlled beta handoff remains the accepted scope. This package makes no city-deployment, publication or general-availability claim.

Terms of service and license

Scope and status

These terms describe the Townlight Beta 1 software and this documentation package. This public Windows beta is for evaluation. No uptime commitment, support response time, indemnity, data-processing agreement or paid service is created by these documents. A separate signed agreement, if any, governs services it actually provides.

Software license and rights

Current Townlight-owned distributions identified in this candidate use **AGPL-3.0-only**. The full license is supplied with this documentation and with the source companion. The license governs rights to run, copy, modify and convey the covered software; these operational terms do not add restrictions to those rights. Historical grants and dependency licenses retain their own terms. CivicCast remains a separate product and is not relicensed by this documentation work.

When conveying covered software, retain applicable notices and provide the corresponding source and installation/build information required by the license. For a modified version supporting remote network interaction, follow section 13's source-offer requirement. The companion archive identifies the source for this candidate; a link to a different public branch is

not a substitute for the matching source. Read the supplied license for the complete conditions. See the GNU license reference and GNU licensing FAQ.

Data ownership and access

Importing a document does not transfer its ownership or grant permission to publish it. The operating organization controls its accounts, access decisions, lawful source use, retention and review/release policy. Keep resident information, credentials, private notes and restricted evidence in their authorized locations. Documentation examples contain fictional data. No real city account, credential or record is required to exercise these examples in an isolated test environment.

Local operation and external processors

The accepted beta workflow is local and uses named staff accounts. Station has separate storage and credentials. Configure backup destinations and optional model/network providers deliberately. A configured external provider may receive submitted prompts, images or audio according to that configuration and provider's terms; this documentation does not promise that every optional configuration is offline. An email outbox or local public view does not establish external delivery or an internet deployment. This static documentation site has no analytics, account collection or remote font dependency.

Staff decisions and source applicability

Staff verify the jurisdiction, source currency, permissions, citations and proposed audience. Model output and generated guidance require review. No draft is automatically an official legal interpretation, zoning determination, permit approval, fee assessment, enforcement notice or records-release decision. Official permit fee authority remains with the adopted source and responsible organization; imported amounts do not become a new authoritative calculation.

Beta operation and recovery

Use the identified candidate in its supported fresh-install scope. Preserve backups and original evidence. Test physical equipment and chosen network/service configuration before relying on them. The supplied acceptance describes selected synthetic workflows, not every screen, vendor, field condition or upgrade. Operators decide whether a particular deployment meets their own operational obligations.

Warranty and liability

The covered software's warranty and liability conditions are those in sections 15–17 of the supplied AGPL and any applicable separate written undertaking. This documentation creates no additional warranty or limitation beyond those terms. Third-party software is subject to its retained license notices. Nothing here removes rights that applicable law makes nonwaivable.

Support and incident reporting

Use the project's existing SUPPORT and SECURITY guidance for its matching repository. Report reproducible behavior with the version, operation and redacted logs. Do not put credentials, resident records or unredacted evidence in a public issue. Coordinate sensitive

reports through the channel designated in SECURITY. No new support address or service commitment is invented here.

Ending a beta exercise and document changes

When an organization ends testing, it controls retention/export and removes access according to its policy. Uninstall may retain data; back up and inspect the selected removal options. Ending an exercise does not revoke rights already granted by AGPL. Later software/documentation versions must identify their changed candidate and scope; historical receipts retain their original identities. These terms do not claim that a city has accepted a contract merely by reading a manual.

Glossary

Term	Meaning
Approval	An explicit authorized decision on the saved text/artifact. It is separate from saving or attachment.
Canonical record	The owning product’s retained identity and saved version, rather than an editable copied label.
Composition	The selected set of desktop modules; inspections-preview exposes the eight staff products.
Corresponding source	The source/build material matching the delivered covered program, subject to its license.
Derivative / Records copy	A separately prepared document with its own intended audience and hash.
Hash / SHA256	A fingerprint of exact bytes. It detects change; it does not establish legal authority.
Idempotency	Repeating the same identified request has the same intended effect; changing its payload can be refused.
IPC	The native desktop command boundary between its browser interface and Rust host.
MCP	Model Context Protocol; no direct Townlight product server is enabled by this beta.
Original	Retained source bytes. A reviewed derivative does not replace them.
Provenance	Source, version, date, actor and relationship information retained with evidence.
Revision	An immutable saved version. Later edits do not inherit an earlier review automatically.

Term	Meaning
Snapshot	A fixed audience-specific copy tied to a source/reviewed revision.
Station	Townlight’s separate media boundary using a pinned CivicCast-derived engine; standalone CivicCast remains separate.

Alphabetical subject index

- **Accounts and authentication** — Security and permissions.
- **AGPL and source** — Terms of service and license.
- **API contracts** — API and integration examples.
- **Architecture** — Architecture.
- **Backup** — Backup recovery and maintenance.
- **Beta version** — Beta identity.
- **Configuration** — Configuration and administration.
- **Evidence and acceptance** — Verified workflows and limits.
- **Fees and official decisions** — Terms of service and license.
- **First run** — First run.
- **Handoffs and audience** — Connected workflows.
- **MCP** — MCP and automation boundary.
- **Originals and provenance** — Security and permissions.
- **Recovery** — Backup recovery and maintenance.
- **Staff procedures** — Daily workflow.
- **Station configuration** — Configuration and administration.
- **Technical stack** — Technical specifications.
- **Troubleshooting** — Troubleshooting.

Operating subject links

- **Configure templates and inspect the outbox** — Configure templates and inspect the outbox.
- **Exercise existing local public intake and tracking** — Exercise existing local public intake and tracking.
- **Fees messages notifications and local intake** — Fees messages notifications and local intake.
- **Record a fee line or waiver** — Record a fee line or waiver.
- **Save staff and requester messages** — Save staff and requester messages.

Source and document control

This edition is authored from the immutable corresponding-source companion, the generated installed-domain schema and the retained workflow receipts. The current software versions and

signed artifacts are unchanged. Public historical pages are not synchronized or published by this local documentation task.

Workflow subject index

- **Draft and approve** — Daily workflow, step 5.
- **Export both artifacts** — Daily workflow, step 6.
- **Find responsive evidence** — Daily workflow, step 2.
- **Fulfill and retain** — Daily workflow, step 7.
- **Preserve the original** — Daily workflow, step 3.
- **Receive and identify the request** — Daily workflow, step 1.
- **Record each release decision** — Daily workflow, step 4.
- Source inventory and file hashes
- All component metadata
- Installed domain OpenAPI
- Workflow matrix
- Full AGPL text

The resident portal is deployed separately. See the Townlight Portal Guide for its workflow and acceptance limits.