Guidedog Discussions GDS 0013

Static Website and Article Publishing in Guidedog

STATE
committed
AUTHORS
Vikrant Rathore
UPDATED
2026-10-31
TYPE
Standards Track
CREATED
2026-10-16
DISCUSSION
Local review; no external discussion URL assigned
LABELS
architecture, build, theme, typst, cli

Record status

The catalog defines the review process and the meaning of each state. This numbered record forms part of the catalog.

Abstract

We publish pages and dated articles through Guidedog's existing document pipeline. We read publication metadata from the first root field list, apply a build-clock gate before resolution, and render eligible documents through project-selected HTML layouts. We express collections with `.. postlist::` and compile optional standalone article PDFs with the shared Typst theme. The `guidedog ssg` commands edit publication state in source files.

Version 1.0.0-beta.6 requires the complete static publishing system described in this record. The implementation includes publication state, mounts, language editions, archives, pagination, collection feeds, pinned themes, starter presets and traced navigation. We publish the release only after the complete acceptance evidence has passed. We do not publish beta.6 with pending SSG work.

Review of the initial proposal

We retain the publishing model and correct several assumptions about the implementation.

- Guidedog is a Rust workspace with `conf.toml` projects. Introducing a competing `guidedog.toml` format would duplicate project discovery, overrides, migration and validation. We extend `conf.toml` instead.
- `gd-jinja` is Guidedog's own template engine. We extend its existing loader and renderer rather than introducing MiniJinja.
- `gd-env` stores sorted `Vec<DocInfo>` values and writes explicit JSON records with `gd-json`. We preserve that representation and bump its cache format from 27 to 28. We do not introduce serde derives or a parallel document map.
- A plain body edit already leaves unrelated pages cached when navigation does not change. `document_settings` hashes titles and navigation, not every source body. Templates can call `hasdoc` and

`toctree` dynamically. We trace their actual arguments and answers before removing the shared HTML navigation digest.

- File size and modification time do not establish content equality. An editor, restored archive or copied file can preserve both while changing the bytes. We retain content hashing.
- A complete build still discovers and verifies inputs across the project. Parsing and HTML rendering can be limited to one changed document, but total build work is not unconditionally $O(1)$. We specify observable rebuild counts instead of an unmeasured 50 ms claim.
- MyST front matter exposes values through the reader's field-list contract. Boolean values appear as `True` and `False` and sequence values appear as JSON text. Publication metadata accepts those representations as well as RST scalar fields. MyST does not currently give every YAML key an exact source span.
- The proposed Typst listing used palette names, font return types and function arguments absent from `theme.typ`. The implemented article template uses the actual shared palette and `configure-document` wrapper.
- A single source update needs one atomic replacement. A multi-file journal is appropriate for a catalog-wide transaction, but unnecessary for replacing one article's metadata.
- Existing builds may fetch configured inventories or Typst packages. This change adds no build-time theme fetching, but does not claim that all existing build modes are offline or that custom templates cannot read permitted project files.

Authoring

We create, schedule, publish and withhold articles with ordinary source files:

```
guidedog ssg new notes.rst --title "Release notes"
guidedog ssg schedule notes.rst --at 2026-10-20T09:00:00Z
guidedog ssg status
guidedog build html --now=2026-10-20T09:00:00Z
guidedog ssg draft notes.rst
guidedog build html --preview
```

We may use `.md` instead of `.rst`. New Markdown articles use YAML front matter. The authoring clock honors `SOURCE_DATE_EPOCH` and may be fixed with `--now`. All generated authoring dates use UTC. Builds retain the existing `SOURCE_DATE_EPOCH` behavior and accept an explicit RFC 3339 `--now` override.
`new` refuses an existing destination. `draft` sets the draft flag without discarding publication history. `schedule` clears the draft flag, sets `publish-at`, and clears an old expiration. `publish` clears draft, schedule and expiration. We preserve an existing past publication date unless `--date` replaces it. We replace a missing or future date with the current UTC date so that publishing takes effect immediately. An explicit future publish date is rejected; we use `schedule` for that case. The commands never commit, tag or move source files.
We validate metadata with the real readers before writing. Edits retain the body bytes, UTF-8 BOM, line endings and file permissions. Inserted metadata follows the first line ending, including leading blank lines, rather than a later mixed-ending example in the body. Duplicate publication fields and unclosed front matter are refused. A directory lock serializes publication commands. A synced temporary file and atomic rename replace an existing source. New sources use an exclusive hard link from the completed temporary file so an existing destination cannot be overwritten. A crash can leave an unused temporary file, but cannot expose a half-written source.



Fig. 14 Source commands edit leading metadata atomically while the build clock governs release and expiration.

Metadata

A leading RST field list or Markdown front matter block carries page metadata:

```
:date: 2026-10-16
:publish-at: 2026-10-20T09:00:00Z
:draft: false
:author: Ada Lovelace
:tags: architecture, rust
:category: engineering
:summary: A short abstract for collections and feeds.
:pdf: article

Publishing a document
=====================

Article body.
```

The recognized fields are:

Field	Meaning
`template`	HTML layout override.
`date`	Publication date or RFC 3339 instant.
`updated`	Significant revision date or RFC 3339 instant.
`publish-at`	Scheduled release instant, overriding `date`.
`expires-at`	Exclusive end of publication, as an RFC 3339 instant.
`draft`	Boolean publication flag, false by default.
`author`	One author; repeated fields preserve source order.
`authors`	Comma-separated authors or a Markdown YAML sequence.
`tags`	Comma-separated tags or a Markdown YAML sequence.
`category`	Primary category.
`series`	Series name.
`summary`	Plain-text abstract.
`pdf`	`article` enables the standalone PDF; absent or `none` disables it.
`nosidebar`	Boolean flag hiding the built-in sidebar.

Unknown metadata fields remain available to other document consumers. Only the first root field list supplies publication metadata. Publication fields and their values are source configuration and are excluded from gettext extraction and translation. Translation catalogs cannot rename a publication key, clear a draft flag or change a release boundary. Later examples and nested field lists cannot change publication state. A dated document is implicitly an orphan for toctree validation.
We validate Gregorian dates, time zones and fractional seconds instead of comparing date strings. Date-only publication values mean midnight UTC. Instants accept numeric offsets and up to nine fractional digits. We reject leap seconds and the unknown offset `-00:00`. Invalid publication metadata prevents an output commit rather than silently publishing a page.

Publication model

Let $r(x)$ be `publish-at`, falling back to `date` when no schedule exists. Let $e(x)$ be the optional expiration instant. For build clock t, release eligibility is

$$\text{eligible}(x, t) = \neg\,\text{draft}(x) \wedge (r(x) = \bot \vee r(x) \leq t) \wedge (e(x) = \bot \vee t < e(x)).$$

Preview builds bypass this gate. A preview output directory can therefore contain unpublished content. We use a separate output directory when retaining both public and preview artifacts.

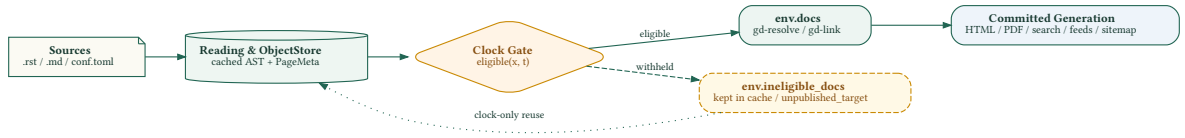

Fig. 15 Publication filtering partitions the cached environment before resolution, indexing and rendering.

We retain withheld `DocInfo` records in `Environment::ineligible_docs` for reuse. We merge them into the previous reading before verifying source digests. After reading, we partition the environment before `gd-resolve` assigns document ids or builds symbol tables. Only `env.docs` reaches navigation, search, inventory, source downloads, collections, feeds and renderers. Cache commit and sweeping retain both eligible and withheld objects. A subsequent clock-only build can publish a scheduled article without parsing it again. A successful rebuild removes artifacts no longer owned by an eligible output unit, including old article PDFs. A failed build retains the previous successful output under the existing transaction contract.

A reference to a withheld document or label reports `link.unpublished_target`. The diagnostic does not repeat the unpublished title or summary. The gate does not inspect arbitrary raw HTML, manually copied extra files or content an author deliberately includes in a public document. Those remain explicit publication inputs.

Collections and discovery

We insert a chronological collection in any RST or MyST document:

```
.. postlist:: 10
   :path: notes/*
   :category: engineering
   :tags: rust
   :author: Ada Lovelace
   :series: compiler
   :sort: -date
   :style: cards
   :excerpts:
```

The optional argument limits the result count; zero or absent means unlimited. Path filters use Guidedog's existing docname glob rules. Category, author and series filters match exactly. Every requested tag must match. We sort by the parsed publication instant, newest first by default, with docname as the stable tie breaker. `:sort: date` selects oldest first. `list` and `cards` use the same semantic list structure with different CSS classes. `:excerpts:` adds rich HTML introductions up to `.. more::`, or escaped plain-text summaries when no marker is present (see *Incremental correctness* below). The link pass produces ordinary list and link nodes, so collection output works through the existing renderers.

We enable deterministic discovery files in `conf.toml`:

```
[site]
base_url = "https://example.org/project"
sitemap = true
feeds = true
```

The HTML and dirhtml builders emit `sitemap.xml`, `atom.xml` and `rss.xml` in the site unit. The base URL includes any hosting prefix. Feeds contain eligible dated documents, use source metadata instead of the wall clock, and escape text and URLs. Sitemaps contain eligible document pages. The feature is opt-in, so projects without `[site]` acquire no discovery files.

Templates and article PDFs

The default HTML template is `page.html`, which extends the existing `layout.html`. Dated documents select `article.html`. A source can select `landing.html` or any project template with `:template:`. Project overrides remain in the existing `templates_path` directories. Built-in names retain the `!` override escape used by Sphinx-compatible layouts. Templates receive publication values in `page.meta`. Values are ordinary escaped text and typed booleans or lists, not executable markup.

An eligible `:pdf: article` document also produces a PDF beside its HTML page. For `notes.rst` with the HTML builder, the files are `notes.html` and `notes.pdf`. For dirhtml, they are `notes/index.html` and `notes/index.pdf`. The generated Typst remains available under `sources/` for inspection. The article template uses continuous prose, a title and author block, an abstract, and running headers and page numbers after page one. It shares typography and directive support with the book and discussion layouts. Project `_templates/article.typ` overrides use the existing `book` function convention with article metadata arguments. Books, discussions and articles share paper-size normalization: `letter` and `us-letter` select Typst's `us-letter`, while the default is `a4`. Untrusted builds use the embedded article template.

Incremental correctness

`LookupKind` remains a small enum; each lookup has a separate string key. We add `Template` and `Postlist` kinds instead of embedding owned strings in the enum. A postlist key contains its complete encoded query, avoiding query-hash collisions. Its answer hashes the ordered selected docnames, titles and page metadata. A body edit therefore does not invalidate a summary-only collection. A membership, ordering, title or metadata change does.

An author-defined `.. more::` boundary selects the rich introduction for `postlist` with `:excerpts:` and for collection templates through `post.excerpt_html`. Without a boundary, both use the escaped plain-text summary. The normal link pass resolves the introduction in its article context, then relocates AST links and namespaces anchors for the destination listing. Targets and footnotes outside the selected AST link to the complete article. Full HTML and PDF articles discard only the marker. Rich listings hash the source object's digest conservatively, including changes after the boundary, and preserve source-scoped reference and asset lookups as `LookupKind::Excerpt`. Feeds retain plain-text summaries and the ordinary publication eligibility gate applies before any excerpt is selected.

Template dependency tracing happens inside `gd-jinja::Environment::load_compiled`. It records cached requests, transitive inheritance, imports, dynamic includes and unsuccessful optional includes. We compare effective sources through the same confined loader before reusing a page. Adding a project override or creating an optional include therefore invalidates the affected pages. Changing an unused template does not invalidate document pages. HTML navigation uses per-page verifying traces. The template context records reads of `toc`, `outline`, `prev`, `next`, `parents`, `rellinks`, `root_url` and article context. Native calls record the actual arguments to `hasdoc`, `pathto` and `toctree`. Their answers hash existence, resolved URLs or the navigation actually rendered at the requested depth. Body toctrees record their selected navigation items, including titles, numbers and child boundaries. A section beyond a traced depth does not invalidate its listing page. The HTML common digest therefore contains configuration, environment settings, inventories and MathJax configuration, without every document's navigation. Books and text builders retain their existing whole-document settings contract. We compute chronological article neighbors once per site build instead of repeatedly sorting the environment during trace checks. We trace the inner `page` dictionary so reading `page.meta.date` does not depend on unused neighbors. Reading `page.older` or `page.newer` records only that neighbor, while rendering the whole dictionary records both.

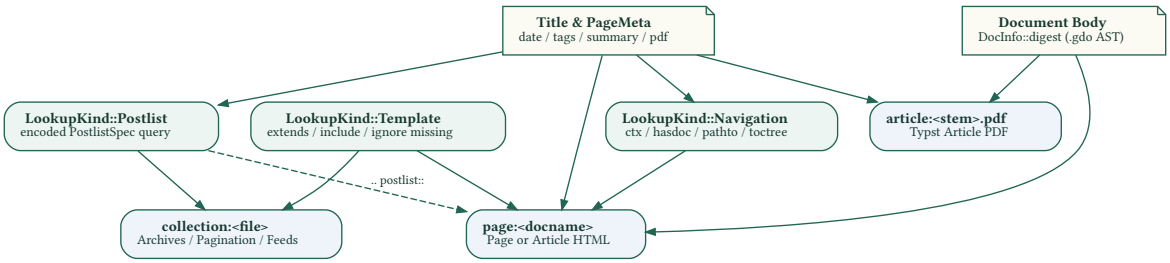

Fig. 16 Per-unit verifying traces isolate body edits from summary-only collections, navigation trees and unused templates.

Archives and pagination

`site.taxonomies` enables tag, category, year, author and series archives. Only eligible dated documents contribute terms. `site.collections` declares named filtered collections with independent destinations, page

sizes, templates and feeds. The filter fields share `PostlistSpec` rather than introducing a second selection language.

A collection starts at `OUTPUT/index.html` and continues at `OUTPUT/page/N/index.html`. Every page has its own output unit and records template and navigation dependencies. Previous and next links are relative to that page. Shrinking or withholding a collection removes its obsolete tail through ordinary output ownership. An empty named collection retains one empty first page.

Taxonomy labels keep their original display text. Portable paths escape uppercase, punctuation and non-ASCII bytes without merging distinct labels. Reserved platform names use a separate escape prefix, and long names use a separately prefixed digest. Before writing a generated output, we reject collisions with author-written pages, additional pages, assets and other generated outputs, including ASCII case aliases and file/directory overlap. Collection feeds contain the complete selection rather than only the current pagination page. Sitemaps include generated collection pages.

Mounted websites

`site.mounts` declares named projects with a source directory, output prefix, builder, setting overrides and optional language editions. An edition can override its source project or use gettext in the common source. Sources remain below the site configuration's directory after canonicalization. Outputs use portable relative directories. Different mounts can share a directory only when their actual files remain disjoint. `site.assets` copies shared files and directories once, under the same collision policy.

`guidedog ssg build` builds every selected mount before committing the complete website. Each project keeps its own incremental cache. One clock applies to every project, and an outer directory lock serializes complete site builds. The existing output journal publishes the aggregate files and manifest together. A failed later mount or missing cross-mount target preserves the preceding complete site.

The `mount://NAME/path` link scheme selects a matching language edition, then the target's default edition. HTML receives a relative URL; PDFs use the configured public site URL. The link pass records each addressed public file in its verifying trace. Assembly validates those files against the current generation, so old artifacts cannot satisfy links to withheld sources.

The supported builders are `html`, `dirhtml`, `pdf` and `discussion-pdf`. The discussion builder validates a catalog and builds a PDF for each active record. The root project's nonempty version and release values apply to its mounts. The `{language}`, `{edition}`, `{site_root}` and `{base_url}` placeholders support declarative setting overrides.

The repository's complete seven-language website is declared in `docs/conf.toml`. `tools/xtask/src/site.rs` selects an executable and delegates to `ssg build`. It contains no project-specific assembly, language loop or publication transaction.

Themes and presets

Theme downloads are explicit `ssg theme install` commands with a required archive SHA-256. Builds never fetch missing themes. The version-one `.gdt` format is bounded UTF-8 JSON containing base64 file bytes beneath `_templates/` and `_static/`. `ssg theme pack` creates this format without depending on an external archive program or executing hooks. We reject symlinks, traversal, special files, case aliases, platform-reserved names and file/directory overlaps. The maximum expanded size is 64 MiB and the maximum file count is 4,096. Paths have at most 32 directory levels, and directory spellings cannot alias under ASCII case folding.

`guidedog-theme.lock` stores the source, archive pin and canonical tree checksum. The checksum hashes a versioned domain tag and sorted length-delimited filenames and file bytes. Installation syncs and verifies an immutable directory named by that checksum before atomically selecting its lock. The encoded lock is validated against the same 16 KiB limit used by offline verification before selection. Offline verification rejects missing, changed or extra files. Project templates take precedence, and project static files can override installed static files. `site.theme = true` selects the verified installation. We apply the installed theme before checking the final configuration and include its tree digest in the settings digest, including for themes that change only static files.

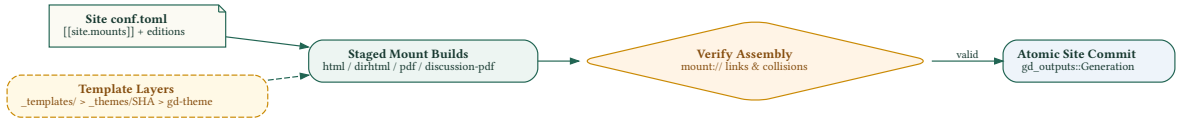

Fig. 17 Mounted website assembly and three-tier offline template resolution.

`ssg init` creates project, journal and research presets. They respectively supply a documentation guide, a chronological journal, and a paper with an abstract, mathematics and a standalone PDF. All three are complete projects using the same build pipeline and editable shared theme modules. Initialization refuses an existing destination.

Acceptance evidence

We require tests for both RST and MyST authoring, exact release and expiration boundaries, offset and fractional timestamps, invalid metadata, cache round trips, and output cleanup. Clock-only changes must reuse parsed objects. A plain body edit must parse one document and emit only its changed HTML page when navigation and metadata are unchanged. Changing an unused template must emit no document pages. A standalone article must compile to a real PDF through the embedded Typst compiler. The existing workspace tests, Clippy, source limits and full website build remain release gates.

Release qualification

Beta.6 requires the complete implementation in this record, including archives, pagination, mounts, editions, themes, presets and navigation tracing. The 1,000-article benchmark is reproducible with `cargo test -p gd-project --test ssg_benchmark -- --ignored --nocapture`. It checks cold and unchanged builds, one body edit and an unused template edit, with actual parsed and written unit counts. The complete workspace checks, multilingual website, internal links and archive smoke tests must pass before publication. Version-control actions remain outside publication commands, and per-article PDFs remain opt-in.

References

- `docs/manual/howto/static-publishing.rst` documents the supported authoring flow.
- `docs/manual/internals/incremental.rst` specifies cache inputs and output ownership.
- `crates/gd-env/src/page.rs` and `page_fields.rs` define publication metadata.
- `crates/gd-jinja/src/environment.rs` records template requests during rendering.
- `crates/gd-project/tests/ssg.rs` exercises clock changes, cache reuse and artifact cleanup.

Review history

- 2026-10-16: Initial discussion draft.
- 2026-10-16: Reviewed against the Rust workspace and revised around existing configuration, cache records, template tracing, source transactions and measured acceptance evidence.
- 2026-10-16: Completed the website orchestration, taxonomy, pagination, themes, presets and navigation contracts; added behavioral, visual, archive-smoke and large-site evidence.

Lifecycle event

```
2026-10-16: prediscussion -> discussion. Assigned a permanent number and opened for discussion.
```

Lifecycle event

```
2026-10-31: discussion -> accepted. Maintainer requested status audit; full
SSG contract shipped in beta.6 and rich excerpts in beta.7
```

Lifecycle event

```
2026-10-31: accepted -> committed. All specified SSG behavior implemented and
qualified in published releases docs/manual/evidence/ssg-review.rst;
docs/manual/evidence/release-1.0.0-beta.6.rst;
docs/manual/evidence/release-1.0.0-beta.7.rst;
crates/gd-project/tests/ssg.rs; crates/gd-project/tests/ssg_benchmark.rs
```