Guidedog Discussions GDS 0018

Confinement and content rules

STATE
committed
TYPE
Standards Track
AUTHORS
Vikrant Rathore
CREATED
2026-11-03
UPDATED
2026-11-03
DISCUSSION
No external discussion URL assigned
LABELS
library, security

Record status

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

Contents

1. Abstract 1
2. Motivation and scope 1
3. Specification 2
4. Rationale and alternatives 3
5. Security and operational considerations 3
6. Backwards compatibility 4
7. Acceptance criteria and implementation status 4
8. Open questions 4
9. Review history 4
10. References 4

1. Abstract

We split the trust a build places in a project into two independent parts. The confinement says what the build may reach on the machine: the folders it reads, the network, build plugins, and the files graphs and music name. The content rules say what reaches the output: raw content and unsafe URLs. `conf.toml` states the content rules with `raw_content` and `unsafe_urls`; a library caller may impose its own or supply defaults. `--untrusted` keeps its meaning by setting both parts to their strict values. Typst the project wrote is no longer disabled by confinement: the embedded compiler draws diagrams and typesets books with the project's preamble and templates, reading only the folders the build may read, and a confined build never runs an installed `typst` program.

2. Motivation and scope

One switch, `BuildOptions::untrusted`, decided three unrelated things. It confined the build: reads stayed below the source folder (`gd_conf::confine_untrusted`), nothing was fetched, no build plugin ran, and graphs and music read no file. It set the content policy: raw content and `javascript:` URLs were omitted. And it disabled the Typst the project wrote: `typst-diagram` directives, `pdf_preamble` and the `_templates` layouts.

The third decision protected nothing. The embedded Typst compiler runs no program and reads only below the root it is given (GDS 0004), so once reads are confined, Typst reads no more than the build does. Worse, `confine_untrusted` kept a configured external `typst` program, so an untrusted PDF build still ran a program the project named.
nisaba, a collaborative authoring application (GDS 0017), builds projects it did not write in a snapshot folder. It needs confinement to that folder, its own linked plugins, and a content rule it chooses, without the project's Typst being disabled. The single switch could not express that.
This record covers `guidedog build` and the library API of `gd-env`, `gd-conf` and `gd-project`. Process sandboxing and limits on native memory and time stay out of scope, as in GDS 0004.

3. Specification

The confinement. `gd_env::Confinement` is a plain struct with five fields.

Table 34 Confinement fields

Field	Meaning	`Confinement::TRUSTED` (`Default`)	`Confinement::STRICT`
`reads: ReadScope`	The folders the build reads.	`Project`: the source folder and every folder `conf.toml` names.	`Within(vec![])`: the source folder alone.
`network: bool`	Fetch intersphinx inventories and Typst packages.	`true`	`false`
`plugins: PluginAccess`	The build plugins that may start.	`Any`: linked, then the machine registry.	`None`: a listed plugin fails the build with `plugins.untrusted`.
`graph_files: bool`	Read the files a graph names, within the readable folders.	`true`	`false`
`music_files: bool`	Read `lilypond` file arguments and `\include`, within the readable folders.	`true`	`false`

`ReadScope::Within(roots)` lets the build read the source folder and the folders `roots` lists (absolute, or relative to the source folder). Before the build discovers or reads anything, it applies `gd_conf::confine(config, roots, linked_plugins)` to its own copy of the configuration. That function keeps `include_roots` and every file and folder setting that leads inside those folders once links are followed, and drops the others: `html_static_path`, `html_extra_path`, `templates_path`, `locale_dirs`, `pdf_font_paths`, `autodoc_source_paths`, `odin_autoapi_dirs`, `rust_autoapi_crates`, `html_logo`, `html_favicon`, `pdf_logo`, `mathjax_config_path`, `pdf_preamble` and each file of `plugin_configs`. It resets `typst` to `"typst"`, the embedded compiler, and drops each listed plugin that is not linked into the calling program. It sets `pdf_packages = "offline"`. Each change is a `build.untrusted` warning, `SETTING NOT USED`; the code is kept so existing `suppress_warnings` entries still apply. `gd_conf::confine_untrusted(config)` is `confine(config, &[], &[])`.
`PluginAccess::Linked` starts only the plugins linked into the program; `gd_conf::drop_machine_plugins` drops the others with the same warning. A confined read scope implies it, because a program from the machine registry reads anything. Without the network the build sets `pdf_packages = "offline"` and fetches no inventory; a confined build reads local inventories only in the readable folders (`IntersphinxOptions::confined`).
A confined build typesets PDF output only with the embedded compiler (`gd_typst::embedded_backend`). A program built without it fails with `typst.missing` rather than run an installed program. Typst diagrams compile in every build, with `read_roots` the source folder; the project template and `pdf_preamble` run in the book's restricted root of the readable folders (GDS 0004).
The content rules. `gd_conf::ContentRule` is `Allow` or `Omit`. `gd_conf::ContentRules { raw, unsafe_urls }` holds an `Option<ContentRule>` for each; `None` is a rule not stated. `Config::content` holds the two `conf.toml` settings.

Table 35 Content settings

Setting	Values	Meaning
`raw_content`	`"allow"`, `"omit"`	Raw HTML, Typst and LaTeX (`.. raw::`). Unset: not stated.
`unsafe_urls`	`"allow"`, `"omit"`	`javascript:`, `vbscript:` and non-image `data:` URLs. Unset: not stated.

Any other value is `config.invalid`. `BuildOptions::content_rules(&config)` gives the rules a build applies, each stated, in this order of precedence:

1. `untrusted`: both `Omit`.
2. `BuildOptions::content`: the rules the build imposes.
3. `Config::content`: what `conf.toml` states.
4. `BuildOptions::content_defaults`: the caller's defaults.
5. `Allow`, as Sphinx does.

`gd_project::content_policy(rules)` turns them into the engine's `Policy`; `trusted_policy` and `untrusted_policy` are its two extremes.

The convenience. `BuildOptions::untrusted` stays. `effective_confinement()` returns `Confinement::STRICT` when it is set and `confinement` otherwise; `content_rules` returns both `Omit`. `guidedog build --untrusted` sets it, so the command line needs no new flag. The digest of the effective confinement and rules (`gd_env::trust_digest`) joins the reading and output settings, so a build with other trust reads and writes everything again.

For example, nisaba confines a build to a snapshot, keeps its own linked plugins, and omits raw content unless the project allows it:

```
let options = BuildOptions {
    confinement: Confinement {
        reads: ReadScope::Within(vec![snapshot]),
        network: false,
        plugins: PluginAccess::Linked,
        ..Default::default()
    },
    content_defaults: ContentRules::OMIT,
    ..Default::default()
};
```

A project in the snapshot with `include_roots = ["../shared"]` keeps that root when `../shared` lies in the snapshot, and loses `include_roots = ["../../home"]` with a warning.

4. Rationale and alternatives

Separating the two parts follows the harm each prevents. Confinement protects the machine that builds; the content rules protect the reader of the output. A service that builds its users' projects and serves them on their own origin may confine reads and still allow raw HTML; a desktop editor may trust reads and omit scripts from a preview.

Keeping `untrusted` as a convenience, rather than removing it, keeps existing callers and the flag working. One `conf.toml` setting for both rules would be shorter, but raw HTML and `javascript:` links differ in use: a project may need one and not the other. We chose `Option` rules over a third `"default"` value, so a caller can tell an unset rule from a stated one.

We considered disabling project Typst only when the external program is configured. Ignoring that program in a confined build is simpler and closes the hole entirely.

5. Security and operational considerations

The trust boundary is the confined configuration: a setting that would lead a read outside the readable folders is dropped before discovery. Typst reads only below its root, which holds links to the readable folders alone; a project template or preamble that reads elsewhere fails as in a trusted build. Project Typst may now take memory and time in an untrusted build, as diagrams and Graphviz already could; GDS 0004 names those as bounded by the operating system. Typst packages already on the machine may be imported in a confined build; none is downloaded.

6. Backwards compatibility

`--untrusted` builds now draw Typst diagrams and use the project's `pdf_preamble` and `_templates` layouts that lie inside the source folder; they no longer warn `BOOK TEMPLATE NOT USED` or about `pdf_preamble`. They ignore `typst` and `--typst` with a warning. Trusted builds are unchanged, except that intersphinx local inventories follow the confinement. In the library, `PluginStart::untrusted` becomes `access: PluginAccess`, `IntersphinxOptions` gains `confined`, `build_settings` and `output_settings` take a trust digest instead of a flag, and `BuildOptions` and `Config` gain fields; struct literals must add them, which the 1.0 beta versioning permits.

7. Acceptance criteria and implementation status

Implemented, with tests:

- `gd-conf`: `untrusted_builds_read_only_below_the_source_folder` (the preamble inside is kept, `typst` reset, plugins and outside plugin configurations dropped); `confinement_keeps_what_lies_inside_the_given_roots`; `content_rules_are_read_from_conf_toml`.
- `gd-env`: `untrusted_sets_the_strict_confinement_and_omits_content`; `the_build_imposes_rules_or_supplies_those_the_project_leaves_unset`; `trust_changes_the_digest`; `untrusted_builds_start_no_plugins` (`None` and `Linked`).
- `gd-project` (`tests/confinement.rs`): content rules from the build, `conf.toml` and defaults; a build confined to a snapshot reads inside it only; Typst diagrams render in untrusted builds; linked-only plugins and no network.
- `gd-book`: `an_untrusted_build_uses_the_projects_template`; `a_confined_build_never_runs_a_typst_program` (the program leaves a mark when it runs).
- Integration: `untrusted_builds_draw_diagrams_and_omit_raw_content`; `books_read_only_what_the_build_may_read` (an untrusted template and preamble are used and still confined); the existing plugin, graph and music refusals under `--untrusted`.

Confinement of `[site]` mount selection in `guidedog ssg` is outside this record's scope; each mount builds with its own options.

8. Open questions

Whether `--untrusted` should allow graph and music files within the source folder, as it now allows Typst, is open; this record keeps them refused. Whether the command line should take `--raw-content` and `--unsafe-urls` is open; `-D raw_content=omit` states the rule today.

9. Review history

On 3 November 2026 the maintainer asked to separate confinement from the content policy for nisaba, observing that disabling project Typst protected nothing once reads are confined and that a configured `typst` program still ran in untrusted builds; this record proposes the change and records the implementation. The maintainer requested merging the implemented Nisaba branch for release 1.0.0-beta.12 on 3 November 2026. Local qualification passed the locked workspace build, 2,216 tests, strict Clippy, formatting, source limits, generated tables, 66 music scores, 12,000 fuzz cases, the 1,000-article benchmark and HTML/PDF, discussion and Pages smoke tests. The release review corrected a compatibility paragraph that still described project Typst templates and preambles as omitted, and completed all six manual and website translations.

10. References

- GDS 0004, memory budgets and trust: readable folders and the Typst root.
- GDS 0010, typed build plugins; GDS 0014, Typst diagrams; GDS 0015, LilyPond music.
- GDS 0017, editable source maps and capability reporting: nisaba's use of Guidedog.
- Sphinx `raw` directive and Docutils `raw_enabled`.

Lifecycle event

> `2026-11-03: prediscussion -> discussion. Assigned a permanent number and opened for discussion.`

Lifecycle event

> 2026-11-03: discussion -> accepted. The maintainer requested merging the
> implemented Nisaba branch and releasing it in 1.0.0-beta.12.

Lifecycle event

> 2026-11-03: accepted -> committed. Ship the implemented Nisaba confinement
> and content rules in 1.0.0-beta.12. Locked workspace build and 2216 tests;
> strict Clippy, formatting, source limits and tables; 66 music scores, 12000
> fuzz cases; project confinement and external-program refusal regressions;
> HTML/PDF, discussion and Pages smoke tests.