

GUIDE · Differentiable Docs

Reference

Every field, setting, command and convention

The complete list: front-matter fields, manifest sections and keys, the commands, the writing conventions, and everything the gate checks. The user guide explains how to use these; this document states what they are.

Document ID	DOC-USER-GD-012
Version	1.10 APPROVED
Owner	Documentation owner
Date	2026-09-25

Public: may be shared without restriction.

Built from commit 4f80011744 on 2026-10-20.

Document control

Document ID	DOC-USER-GD-012
Type	Guide
Product	Differentiable Docs
Classification	Public
Owner	Documentation owner
Authors	Differentiable Solutions S.r.l.
Reviewers	—
Approved by	Documentation owner

Revision history

Version	Date	Author	Change
1.0	2026-09-16	Differentiable Solutions S.r.l.	First release of the user documentation.
1.1	2026-09-17	Differentiable Solutions S.r.l.	Added the producer to pdf.generator, inline code breaking to look.break-long-code, and the placeholder refusal to the gate; clarified that make pdf and make html ignore build.formats; removed forced chapter page breaks.
1.2	2026-09-17	Differentiable Solutions S.r.l.	Added build.uncontrolled and what the gate does with files that have no front matter.
1.3	2026-09-17	Differentiable Solutions S.r.l.	make toolchain also installs mmdr, which draws diagrams.
1.4	2026-09-18	Differentiable Solutions S.r.l.	Added make pages, which builds the published site.
1.5	2026-09-18	Differentiable Solutions S.r.l.	The paper also sets the printed page of the HTML; a long caveat wraps before the page number; the logo is embedded in the HTML.
1.6	2026-09-18	Differentiable Solutions S.r.l.	Stated that sections are numbered, in the PDF and the HTML alike.
1.7	2026-09-18	Differentiable Solutions S.r.l.	Version and date from the revision history, status optional, PDF/A-2u by default, and fonts only from the project.
1.8	2026-09-18	Differentiable Solutions S.r.l.	The build's output is now under build/differentiable-docs/. make clean removes that folder and nothing else.
1.9	2026-09-18	Differentiable Solutions S.r.l.	The type is read from the identifier's code, and each revision's date and author, and the document's authors, from Git; the type codes are listed, and so is how an entry is found and when a history is refused.
1.10	2026-09-25	Differentiable Solutions S.r.l.	GitLab Flavored Markdown is the input language's reference: callouts are GitLab alerts with an optional title, and the constructs GitLab draws are listed with what the build does with each; added highlighted fenced code to the conventions.

Contents

- 1 The input language 3
- 2 Front matter 3
- 3 The manifest 4
 - 3.1 PDF conformance 8
 - 3.2 Fonts 8
- 4 Commands 8
- 5 Conventions 8
- 6 Document types 9
- 7 What the gate checks 9
- 8 Exit behaviour 10

1 The input language

GitLab Flavored Markdown (decision 0008): what GitLab draws in a merge request is what the PDF and the HTML show. pandoc reads it with its GFM reader and four extensions, for what a controlled document needs and GitLab has no syntax for.

Extension	Gives
attributes	{width=80%} on an image, {#id} on a heading
fenced_divs	::: {.landscape} and ::: {.pagebreak}, layout that has no Markdown spelling
implicit_figures	an image alone in a paragraph becomes a numbered figure
smart	quotes, dashes and ellipses set as print typography

Everything else is GitLab's own: alerts with or without a title, pipe tables, task lists including [~], footnotes, strikethrough, autolinks, emoji, mathematics ($...$, $\dots$, $\dots$, a math block), Mermaid, <sup>, <sub>, `<kbd>`, raw HTML and the YAML metadata block. [[_TOC_]] and [TOC] are dropped, since the document has its own contents. References such as #123, !45 or @name stay as written; they link only inside GitLab.

What GitLab draws and the PDF could not show the same way is refused, with the spelling to use: a >>> multiline quote, an include inside an included file, an inline diff {+ ... +} or {- ... -}, a description list, an alert of a kind GitLab does not have such as [!FINDING], and a plantuml or json:table block. So are a LaTeX \newpage and a grid table, which GitLab would print as text.

2 Front matter

Required unless the project's manifest says otherwise (vocabulary.required). A required field is satisfied by the front matter, by the manifest's defaults, or by what the build works out, in the column on the right.

Field	Required	Value	When not written
title	always	free text	refused
doc-id	yes	matches vocabulary.doc-id-pattern; unique across the project	refused
doc-type	yes	a value from vocabulary.doc-types	the type whose code is the identifier's segment before the number
classification	yes	a value from vocabulary.classifications	defaults.classification, else refused
owner	yes	the accountable person or role	defaults.owner, else refused
version	yes	"MAJOR.MINOR" or "MAJOR.MINOR.PATCH", quoted	the last revision-history entry's
date	yes	YYYY-MM-DD, a real calendar date	the last revision-history entry's
status	no	a value from vocabulary.statuses	the first of them, draft

What Git records, the build reads rather than asks for. Git is asked only for what the front matter leaves out, and not at all outside a Git work tree.

Value	From Git
a revision's date	the author date of the commit that last changed the entry's version: line, in that author's time zone; whitespace-only changes do not count
a revision's author	that commit's author, after .mailmap
an entry not committed yet, or a file Git does not track	today, and <code>git config user.name</code>
author	every author of a commit to the file, or to the folder of a document made of parts, in the order of their first commit; merges and bots (...[bot], GitLab's project_..._bot, group_..._bot) left out

An entry is found by its own `version:` line, so write each one on a line of its own. A shallow clone that does not reach a commit the build needs is refused, naming `GIT_DEPTH`, and so is a history whose entries cannot be lined up with the file, rather than dating them today.

Optional:

Field	Effect
subtitle	a line under the title
abstract	paragraph on the title page; the metadata subject unless <code>subject</code> is set
product	shown above the title
author	list; on the control page and in the metadata; from Git when left out
reviewer	list; on the control page
approved-by	on the control page; expected when <code>status: approved</code>
revision-history	list of {version, change}, and optionally date and author to override Git's; the table on the control page
caveat	warning in the footer of every page, in the caveat colour; a long one wraps before the page number
subject	metadata subject, overriding the abstract
keywords	list; added to the metadata keywords
control-page	false omits the document-control page
toc	false omits the contents
toc-depth	heading depth in the contents (default 2)
h1-pagebreak	true starts every top-level section on a new page
paper	any Typst paper name, e.g. a4, us-letter; also the printed page of the HTML
lang	document language; becomes the PDF's language
logo	path inside the project, relative to its root; embedded in the HTML
organisation	overrides the project's organisation for this document
strings	overrides individual fixed labels

A document may not define `vocabulary`, `manifest`, `template` or `build`. Those are the rules it is judged by, and one that tries is rejected.

3 The manifest

`differentiable-docs.yaml` states only what differs from `config/defaults.yaml`. Settings resolve in four layers, each merged over the one before: the built-in defaults, the project manifest, any area

manifest from the project root down to the document, then the document's own front matter. `make manifest` prints the first two.

Maps merge key by key. The `vocabulary` and `layout` lists merge by each entry's `value` or `area`: an entry with the same key overrides its fields, a new one is appended, and `remove: true` drops one. Other lists and scalars are replaced.

Section	Key	Means	Set in an area manifest
template	name, version	which release the project is built from; defaults only	no
project	name	used in the Pages index and release names	yes
project	organisation	printed in the title band and the footer	yes
project	logo	path inside the project	yes
look	lang	document language	yes
look	paper	any Typst paper name; a4 by default, for the PDF and the printed HTML	yes
look	mainfont, monofont	body and code faces	yes
look	accent	links and rules	yes
look	caveat-color	the colour of a document's footer caveat	yes
look	control-page	whether documents carry a control page	yes
look	break-long-code	let long identifiers break in running text too; table cells always may	yes
build	formats	what a build produces; [pdf] by default, [pdf, html] for both	no
build	exclude	paths under docs/ to leave out; shell globs	no
build	strict-layout	make a document outside the layout an error, not a warning	no
build	uncontrolled	skip (default): a Markdown file with no front matter is named and left alone; refuse: it fails the gate	no
layout	area, purpose, doc-types	the areas under docs/, for scaffolding and checking	no
tables	landscape-threshold	columns at which a table gets its own landscape page; 0 disables	no
tables	width-blend	0 purely content-proportional widths, 1 equal columns, none to leave sizing to the backend	no
meta	author, subject, keywords	auto, none, or a literal for every document	yes
meta	date	auto, build or none; a date is structured, so no literal	yes
meta	keyword-fields	identity fields added to the keywords for indexing	yes
pdf	generator	the application named as the PDF's creator and producer; empty leaves Typst's own	no
pdf	standard	conformance the compiler enforces; see below	no
strings	any label	the fixed wording of the title, control and footer areas, for translation	yes

Section	Key	Means	Set in an area manifest
defaults	any front-matter field	values documents inherit unless they set their own	yes
vocabulary	doc-id-pattern	a Lua pattern every identifier must match	no
vocabulary	classifications	value, color, note per classification	no
vocabulary	statuses	value, label, watermark per status	no
vocabulary	doc-types	value, code, label per type; code is unique	no
vocabulary	required	which front-matter fields are mandatory	no

An area manifest may set only `project`, `look`, `strings`, `defaults` and `meta`. The rules, the build settings and the layout stay project-level, so an area cannot weaken the gate its own documents must pass, and a build-wide setting such as PDF conformance cannot vary document by document.

3.1 PDF conformance

`pdf.standard` is `a-2u` by default: PDF/A-2, an archival file, whose text can always be searched and copied. The build's own output passes veraPDF at that level. It accepts `1.4`, `1.5`, `1.6`, `1.7`, `2.0`, `a-1b`, `a-1a`, `a-2b`, `a-2u`, `a-2a`, `a-3b`, `a-3u`, `a-3a`, `a-4`, `a-4f`, `a-4e` and `ua-1`, several comma-separated, or empty for none. `a-2a` and `ua-1`, for accessibility, require alternative text on every image; `make check names` each image that has none.

3.2 Fonts

A document is set in the build's own fonts, vendored in its `fonts/`, and in any the project keeps in a `fonts/` folder of its own. Fonts installed on the machine are never used, so a laptop sets a document exactly as the pipeline does. A corporate face named in `look.mainfont` goes in the project's `fonts/`; `make check` warns when a face the manifest names is not there.

4 Commands

Command	Does
<code>make</code>	every document in <code>docs/</code> , in the manifest's formats (the PDF, by default)
<code>make pdf / make html</code>	one format only, whatever <code>build.formats</code> says
<code>make examples</code>	the worked examples, to <code>build/differentiable-docs/examples/</code>
<code>make types</code>	smoke-build every skeleton
<code>make check</code>	the gate
<code>make test</code>	the build's own tests
<code>make list</code>	the document keys that would be produced, and the files left alone as not controlled
<code>make scaffold</code>	create missing layout areas under <code>docs/</code>
<code>make manifest</code>	print the effective manifest
<code>make pages</code>	build the site in <code>public/</code> from the documents already built; <code>PAGES_DOCS="user/guide user/reference"</code> presents only those
<code>make toolchain</code>	install the pinned pandoc, Typst and mmdr, which draws diagrams
<code>make clean</code>	remove <code>build/differentiable-docs/</code> , and nothing else
<code>./build.sh <path></code>	one document; <code>--html</code> , <code>-o <file></code> , <code>--typ</code> , <code>--check</code> , <code>--list</code>

From the image, `differentiable-docs` replaces `make` and takes the same arguments.

5 Conventions

Written as	Produces
<code>> [!note]</code>	a callout headed by its kind; also <code>[!tip]</code> , <code>[!warning]</code> , <code>[!caution]</code> , <code>[!important]</code> , in either case
<code>> [!warning] Title</code>	the same callout headed by the title, as GitLab draws it

Written as	Produces
> [!important] Decision required: ...	a decision; > [!warning] Gap: ... a gap; > [!caution] Finding: ... a finding
::: {.note} ... :::	the older spelling of a callout, also .decision, .gap, .finding; builds, with a warning
```powershell ... ```	a code block highlighted in the PDF and the HTML alike; pandoc --list-highlight-languages names the languages, any other is plain
::: {.landscape} ... :::	a landscape page for a wide table
::: {.pagebreak} :::	a page break
::include{file=01-scope.md} at the start of a line	splices in the file, relative to this one, as GitLab does; the lines together in a paragraph of their own, at most 32 in a document, none inside an included file; inside a code block, the file's text
a code block with class include	the older spelling: splices in the listed files, shift-heading-level-by=1 nesting their headings; builds, with a warning
![Caption](images/x.svg){width=80%}	a figure with a caption and alternative text
[C], [I], [A]	evidence badges: confirmed, inferred, assumption
#, ##, ### headings	numbered sections, 1, 1.1, 1.1.1, in the PDF and the HTML alike, so a section can be cited by number

## 6 Document types

The type segment of an identifier names its type, so DOC-PLATFORM-SPEC-001 is a specification and needs no doc-type. A project adds its own types, each with a code of its own, in vocabulary.doc-types.

Type	Code	Type	Code
report	RPT	threat-model	TM
specification	SPEC	risk-register	RR
plan	PLAN	security-requirements	SR
procedure	PR	verification-plan	VP
policy	POL	compliance-matrix	CM
guide	GD	pentest-remediation	PTR
decision-record	ADR	security-release-report	RS
assessment	ASM	vulnerability-advisory	VA
minutes	MIN	other	DOC

## 7 What the gate checks

make check, and the same job in the pipeline:

1. **Names** — no spaces in any path under docs/, examples/ or skeletons/, since they become build targets, and no symlinks anywhere under them: a symlink reaches outside the repository without any path that looks like it does. These two are checked whenever the build reads the tree, so they stop make as well as make check.

2. **Front matter** — every required field, the identifier pattern and its type code, the vocabularies, the version and date formats, a history Git can date, and no `TODO` placeholder left from a skeleton. This also proves every include resolves inside the project, and refuses the syntax GitLab and the PDF would not show alike (the input language, above).
3. **Identifiers** — unique across documents and examples; skeletons are exempt.
4. **Orphan parts** — a file beside an `index.md` that none of its includes names.
5. **Fonts** — the faces the manifest names are available.
6. **Secrets** — private-key headers, GitLab, GitHub, AWS, Google, Slack and OpenAI token shapes, JSON web tokens, and credential-looking assignments, in every Markdown file under the document, example and skeleton roots. A line that legitimately needs such a pattern carries the marker `allow-secret-pattern`.

Items 1 to 3 and 6 fail the build. Items 4 and 5 report and continue.

## 8 Exit behaviour

Anything that would produce a misleading document stops the build: a missing or malformed field, a skeleton placeholder left in the front matter, an unknown vocabulary value, a duplicate identifier, an include or image that is absolute or escapes the project, an inclusion loop, a manifest that does not exist when named explicitly, a manifest format version this release does not read, an identifier whose type code is no type's when no `doc-type` is written, a shallow clone that does not reach the commits dating a revision, a revision history whose entries cannot be lined up with the file, a document at the root of `docs/` rather than in an area, two documents with the same key, a path with a space in it, a symlink under any of the roots, syntax from another Markdown dialect that would print as text, and an output directory outside the project.

Anything that is merely a mistake reports and continues: a missing revision history, a version that disagrees with the last history entry, an approved document with no approver, an approved document built from changes that are not committed, a `doc-type` that contradicts the identifier's code, an orphan part, an unavailable font, a Markdown file with no front matter (named and left alone, unless `build.uncontrolled` is `refuse`), a project with no documents yet, a second spelling of the documents folder beside the one that is read, and a document in an area the layout does not declare — unless `build.strict-layout` is on, which makes the last of these an error.