markpublish Quick Reference

Configuration, Data in the Text and Templates

> *Reference for every key of markpublish.yaml, placeholders and statements in the text, and the building blocks of a theme.*

Version	2.4.1
Date	2026-10-13
Author	Frank Winter
Copyright	© 2026 Frank Winter

markpublish.yaml

Top level: `document`, `theme`, `parts`, optionally `templates_dir`. All paths are relative to the directory of this file. Build with `markpublish build [CONFIG_FILE] [OPTIONS]`.

Structure

There are exactly two structural tiers. Anything deeper is carried by the Markdown file itself.

```
markpublish.yaml
├── document:            metadata and global switches
├── theme:               name of the theme
└── parts:               list of sections
    ├── part:            name of the section
    └── chapters:        list of chapters
        └── file:        one Markdown file
            ├── # H1     chapter heading
            ├── ## H2      structure
            └── ### H3     inside the file
```

document

Metadata and global layout toggles. Only `title` is required.

Key	Default	Meaning
`title`	*required*	Document title (cover page, header, filename)
`subtitle`	–	Subtitle on the cover page
`summary`	–	Summary / abstract on the cover page
`author`	–	Author name
`date`	`auto`	Date (`auto` or `today` for the build date)
`version`	–	Version on the cover page and in the footer
`status`	–	Document status (e.g. `Draft`, `Approved`)
`copyright`	–	Copyright notice on the cover page
`language`	system language	ISO code for static labels (`de`, `en`)

© 2026 Frank Winter

Key	Default	Meaning
`custom`	–	Own values for placeholders in the text (`{{custom.a.b}}`), may be nested
`cover`	`false`	Generate a cover page
`header` / `footer`	`true`	Running headers and footers
`document_toc`	`full`	Main table of contents: `none`, `full` or a depth
`part_toc`	`full`	Default for part divider pages: `none`, `full` or a depth
`chapter_toc`	`full`	Default for chapter divider pages: `none`, `full` or a depth
`autonum_pattern`	*see Numbering*	How numbers are built; slot 1 is the part here
`autonum_reset`	`false`	Levels below start again at one when entered
`part_label`	from i18n	Word before a part number (`Part`); `""` drops it
`chapter_label`	from i18n	Word before a chapter number (`Chapter`); `""` drops it
`pagenum_reset`	`false`	Restart page numbering per part or chapter

If the document consists of **exactly one chapter**, the main table of contents omits that chapter's own line: it would merely repeat the cover title, and an entry without siblings distinguishes nothing. The chapter's subheadings stay listed, and its number still appears above the text or on the divider page. `document_toc` still counts structural levels rather than visible lines: for two visible levels (H2 and H3), write `document_toc: 3`.

Custom fields under `document:` (`department: "R&D"`) reach the theme through the `meta` dictionary (`meta.at("department").value`). Static labels belong in `i18n.yaml`, not here.

Values under `custom` do not reach the theme; in Markdown, `{{custom.a.b}}` inserts them (see "Data in the Text").

`key: !file "path-or-url"` loads a value from another file, anywhere in the configuration. YAML files supply a nested value, any other file its text. Relative paths resolve against the referencing file.

parts

A part brackets chapters and passes settings down to them. Without `break_before` it does not show itself.

© 2026 Frank Winter

Key	Default	Meaning
`part`	*required*	Name of the section
`chapters`	*required*	List of chapters in this section
`toc_title`	`part`	Title in the main table of contents
`divider_title`	`part`	Title on the divider page
`subtitle` / `summary`	–	Subtitle and summary on the divider page
`break_before`	`none`	`divider`, `page` or `none`
`document_toc`	inherited	Contribution to the main TOC; `none` hides the part
`part_toc`	inherited	Local TOC on its own divider page
`chapter_toc`	inherited	Default for the divider pages of its chapters
`autonum_pattern`	inherited	Numbers of its chapters; slot 1 is the chapter here
`autonum_reset`	inherited	Chapters of this part start again at one
`label`	from i18n	Word before its own number
`chapter_label`	inherited	Word before the number of each of its chapters (`Appendix`)
`pagenum_reset`	inherited	Restart page numbering at the start of the part

chapters

A chapter is one Markdown file. Its name comes from that file's `#` heading.

Key	Default	Meaning
`file`	*required*	Path to the Markdown file
`list_of`	–	Instead of `file`: list of `figures` or `tables` at this position
`show_title`	`true`	Show the file's H1 on the content page
`toc_title`	file H1	Title in the table of contents and running header
`divider_title`	file H1	Title on the divider page
`subtitle` / `summary`	–	Subtitle and summary on the divider page
`break_before`	`page`	`divider`, `page` or `none`
`document_toc`	inherited	Contribution to the main TOC (`none`, `full`, depth)
`chapter_toc`	inherited	Local TOC on its own divider page
`autonum_pattern`	inherited	Numbers *inside* the chapter; slot 1 is the H2 here

© 2026 Frank Winter

Key	Default	Meaning
`autonum_reset`	inherited	Headings of this chapter start again at one
`label`	inherited	Word before its own number (`Appendix A: …`)
`pagenum_reset`	inherited	Restart page numbering at the start of the chapter

Inheritance

Everything marked *inherited* flows from the outside in; the innermost setting wins.

```
document ──────────► parts[] ──────────► chapters[]
  autonum_pattern       autonum_pattern       autonum_pattern
  autonum_reset         autonum_reset         autonum_reset
  document_toc          document_toc          document_toc
  part_toc              part_toc                   -
  chapter_toc           chapter_toc           chapter_toc
  pagenum_reset         pagenum_reset         pagenum_reset
  part_label            label                      -
  chapter_label         chapter_label         label
```

Numbering

A pattern describes the levels *below* the place it stands — so slot 1 shifts with that place.

```
  Level            under document      on a part         on a chapter
  ──────────────────────────────────────────────────────────────────────
  Part               slot 1                -                 -
  Chapter (H1)       slot 2              slot 1              -
  H2                 slot 3              slot 2            slot 1
  H3                 slot 4              slot 3            slot 2
   ⋮                   ⋮                   ⋮                 ⋮
  H6                 slot 7              slot 6            slot 5
```

Symbols: `1` → 1, 2, 3 · `01` → 01, 02 · `a` / `A` → a, b / A, B · `i` / `I` → i, ii / I, II · `_` level without a number · `+` repeats the slot before it for every deeper level. Slots are separated by `|`; everything else in a slot is a literal, reserved characters go in single quotes (`'Article '1`).

```
  on document   "_|1|.1|+"              part unnumbered · chapter 1 · 1.1
  on document   "_|_|1|.1|+"            single-chapter document: H2 becomes
1., 2., 3.
```

© 2026 Frank Winter

```
on document   "I|1|.1|+"              Part I · chapters 1, 2, 3 through
on a part     "A|.1|+"                chapter A · A.1 · A.1.1
on a part     "'Article '1|' ('1')'"  Article 1 · Article 1 (2)
```

Without a setting, `"_|1|.1|+"` applies: parts unnumbered, chapters 1, 1.1, 1.1.1.

The part number does **not** enter the chapter numbers; a part with `document_toc: "none"` gets none and consumes none. A notated `label` steps in front of the heading's number (`Appendix A: Schema`) without reaching the subheadings (`A.1`).

© 2026 Frank Winter

Data in the Text

Values from `document:` and `custom:` can be inserted into the Markdown, repeated, looked up and tied to conditions. A value without an entry stops the build with file and line.

Input	Effect
`{{author}}`, `{{custom.a.b}}`	insert a value; email and web addresses become links
`{% for x in custom.group %}` … `{% endfor %}`	repeat lines per entry, in YAML order
`{% for (x, key) in custom.group %}`	also the entry's name as `{{key}}`
`{% set short = custom.a.b %}`	short name for a path, until the end of file or block
`custom.systems[f.system]`	lookup: insert the value of `f.system` as the key
`{% if path %}` … `{% else %}` … `{% endif %}`	lines only for `true`; `else` optional
`{% if path == "value" %}`	compare with a text
`\{{author}}`, `\{% … %}`	show literally; code stays literal anyway

Statements stand alone on their line (with `>` in front inside a quote or callout) and disappear in the build, so a table stays contiguous. Lookups work only in the Markdown, not in the entries under `document:` themselves.

```
| Data flow | System | Operation |
| :--- | :--- | :--- |
{% for f in custom.flows %}
{% set sys = custom.systems[f.system] %}
{% if sys.external %}
| {{f.title}} | {{sys.name}} | Service provider |
{% else %}
| {{f.title}} | {{sys.name}} | In-house |
{% endif %}
{% endfor %}
```

© 2026 Frank Winter

Templates and Themes

Everything visible is controlled by a theme: typography, page layout, and static text. The built-in default theme `default` is provided for PDF.

Where Themes Are Located

The first match wins — a theme is always loaded from exactly one source location and never merged across multiple locations. Which theme is used is determined by the `--theme` command-line option or the `theme:` setting in `markpublish.yaml` (default: `default`).

Rank	Location
1. User	`~/.markpublish/templates/<theme>/` or user-specific AppData directory
2. Project	`--templates-dir` on CLI, `templates_dir:` in YAML, `MARKPUBLISH_TEMPLATES_DIR`, or `./templates/<theme>/`
3. Package	Built-in standard theme `default`

`markpublish templates` lists all templates available on the system along with their resolution rank and origin.

Customizing a Theme

```
markpublish export-template default ./templates
```

This command copies the built-in theme files into your project directory, where resolution rank 2 automatically discovers and uses them on the next build. For PDF output, a theme directory contains:

- `template.typ`: Main Typst layout template defining page geometry, headers, footers, and styles
- `i18n.yaml`: Shared label definitions and localized strings across the theme
- `pdf/i18n.yaml`: Format-specific label definitions and string overrides for PDF
- `fonts/`: Optional custom font files (.ttf, .otf) loaded by Typst
- `assets/`: Image assets, logos, and vector icons

© 2026 Frank Winter

Static Labels (i18n)

Labels resolve through a four-stage cascade; a lower level overrides only the keys it explicitly defines:

Level	File
1. Application	`markpublish/i18n.yaml` (application default strings)
2. Theme	`<templates>/<theme>/i18n.yaml`
3. Target format	`<templates>/<theme>/pdf/i18n.yaml`
4. Project	`./i18n.yaml` next to `markpublish.yaml`

Each file is organized by language code (`de:`, `en:`); the key `"*"` applies to all languages. Level 4 belongs to the project: it allows overriding labels and localizing custom metadata fields (`department: "Department"`). Use `markpublish labels --overridden` or `markpublish labels --theme <theme>` to inspect the cascade before building.

Key CLI Commands

Command	Key Options	Description
`markpublish build`	`--theme`, `--target`, `-o`, `--templates-dir`	Compile document to PDF
`markpublish init`	`[TARGET]`, `--title`, `--lang`	Set up a minimal starter project
`markpublish cheatsheet`	`--lang`, `--theme`, `-o`, `--templates-dir`	Generate this quick reference as PDF
`markpublish manual`	`--lang`, `--theme`, `-o`, `--templates-dir`	Generate the complete user manual
`markpublish templates`	`--target`, `--templates-dir`	List templates and lookup rank
`markpublish export-template`	`[THEME]`, `[TARGET]`, `--target`	Export template files for customization
`markpublish labels`	`--theme`, `--target`, `--templates-dir`, `--overridden`	Inspect static labels and metadata

© 2026 Frank Winter