`margo pdf`

Purpose

`pdf` renders one Markdown document, then exports it through a selected local PDF engine.

Input and output

Input is one path or `-` for stdin. Unlike `html`, the command requires an explicit `--output PATH` or `--output -`. PDF bytes are binary and go only to that destination; diagnostics stay on stderr.

The document defaults are A4 portrait with margins of 24 mm top, 22 mm right, 26 mm bottom, and 22 mm left. `--image-overflow limit` caps print images to the safe projection; `allow` permits overflow. `--print-chart-data` includes the accessible exact-data tables that PDF otherwise omits after charts.

For configured pre-rendered PDFs, the equivalent page action is `margo.actions.pdf.printChartData: true`; see the corporate branding example below. The object form defaults to `pre-rendered` and the existing boolean and string forms remain compatible.

Examples

Run a compatibility check and engine probe before conversion:

```
margo check docs/guide.md --target pdf
margo doctor
margo pdf docs/guide.md \
  --output build/guide.pdf \
```

```
  --page-size A4 \
  --orientation portrait
```

Document frontmatter can declare page size, orientation, image overflow, and individual margins. Explicit CLI flags override those preferences. Setting all four margin flags to `0` requests a full-bleed page:

```
margo pdf docs/guide.md \
  --output build/guide-full-bleed.pdf \
  --margin-top 0 \
  --margin-right 0 \
  --margin-bottom 0 \
  --margin-left 0
```

Relative PDF links default to `strip`: link text remains, but the relative target is removed. `--base-url` implies `resolve` unless `--relative-links` was explicitly set. Use an absolute public HTTP or HTTPS URL; loopback and unspecified hosts are rejected.

```
margo pdf docs/guide.md \
  --output build/guide-public.pdf \
  --base-url https://docs.example.com/manual/
```

Corporate branding

For a repeatable branded PDF artifact, use a configured site. Set the public name and a local SVG logo in `site.yaml`, then opt the document into a pre-rendered PDF with `margo.actions.pdf: true`:

```
publication/
├── site.yaml
├── brand/
│   ├── company-logo.svg       # local SVG used in the PDF header
│   └── social-preview.jpg     # local 1280x640 JPEG or PNG
└── docs/
    └── report.md
```

```yaml
# publication/site.yaml
version: 1
source: docs
output: dist
assets: local
offline: true
site:
  name: Acme Corporation
  description: Acme's corporate reports.
  base_url: https://docs.acme.example
  home: report.md
  logo: brand/company-logo.svg
  icon: brand/company-logo.svg
  social_image:
    path: brand/social-preview.jpg
    alt: Acme corporate report
locales:
  default: en
  supported: [en]
navigation:
  mode: file-tree
```

```
<!-- publication/docs/report.md -->
---
title: Quarterly report
language: en
margo:
  actions:
    pdf: true
---

# Quarterly report

The report body.
```

Build the configured publication and find the branded artifact at `dist/report.pdf`:

```
margo site ./site.yaml
```

The configured site's `site.name` and local SVG `site.logo` are materialized into the pre-rendered PDF furniture. The logo must be a local SVG at a normalized relative path; remote URLs are not a branding path for the offline site builder. Keep the SVG inert because it is embedded as trusted document branding. `pdf: client` is a different mode: it opens the browser print flow, follows the active site theme, and does not publish a PDF artifact.

To include the exact chart-data tables in a branded pre-rendered PDF, use the object form of the action. The object defaults to `pre-rendered`:

```yaml
margo:
  actions:
    pdf:
      printChartData: true
```

The boolean and string forms remain compatible with existing documents. `printChartData` is only valid for pre-rendered output; browser client printing continues to follow the browser's print behavior.

For a one-off standalone `margo pdf`, keep the logo below the Markdown file's directory and put it in the document instead. For example, with `docs/brand/company-logo.svg` next to `docs/report.md`:

```markdown
![Acme Corporation](brand/company-logo.svg)
```

```
margo pdf docs/report.md --output build/report.pdf
```

That image is part of the document content. Standalone CLI output has no `--brand`, `--logo`, or custom-theme flag, so this does not replace its standard PDF header and footer. Programmatic PDF brand selection is a Go-library concern, not a frontmatter or CLI setting: call `margo.RenderStandalone(rendered, margo.WithPDFBrand(name, logo))`, where `logo` is a materialized `margo.AssetRef` (not a path that Margo fetches at render time).

Failures and diagnostics

Code	Meaning
cli.output_required	No PDF destination was selected
pdf.engine_mode_invalid	Engine is not auto, chromium, or native
pdf.engine_path_invalid	Explicit executable is unusable
pdf.engine_unavailable	No requested candidate is available
pdf.native.compiled_out	This build has no verified native backend
pdf.page_size_unsupported	Size is not A4 or Letter
pdf.orientation_unsupported	Orientation is unsupported
pdf.margin_invalid	A margin is negative or non-finite
cli.relative_link_base_required	resolve has no --base-url
pdf.relative_link_base_invalid	Base URL is unsafe or malformed
pdf.relative_link_forbidden	error policy found a relative link

Existing output requires `--force`. All command failures exit `1` and report to stderr without replacing a protected destination.

Limitations and care

Engine mode is `auto`, `chromium`, or `native`. Auto discovery checks an explicit `--engine-path`, `MARGO_CHROMIUM_PATH`, executables on `PATH`, known platform locations, then the native slot. Margo never downloads a browser. A selected engine that fails does not fall back to another candidate. `doctor` reports candidates but cannot guarantee a later document-specific render will succeed.
