

차례

1 들어가며	2
2 첫 변환의 전체 과정	3
2.1 설치와 가장 짧은 사용법	3
2.2 manifest가 알려 주는 것	3
2.3 diagnostic 읽기	3
2.4 종료 code	4
2.5 형식 판별	4
3 하나의 공통 document representation	5
4 Reader, Writer, Bundle	6
5 Office와 container 형식	7
6 하나의 Writer	8
7 입력을 공격적인 data로 다루기	9
8 내장과 Filter	10
8.1 Zig API	10
8.2 Python API	10
8.3 Browser API	10
9 CLI 참고	12
10 벤치마크	13
11 Server	14
12 더 읽을 자료	15

1 들어가며

문서는 글쓴이와 독자 사이의 약속을 전달합니다. 파일 형식은 그 사이에 놓인 역사적인 그릇일 뿐입니다. zenfmt 는 Zig로 작성한 문서 변환기로 19가지 입력 형식을 읽고 GitHub Flavored Markdown을 출력합니다. Pandoc 만큼 넓은 범위를 제공한다고 말하지 않습니다. 대신 작은 engine, 분명한 resource limit, 재현 가능한 출력, 변환 중 잃은 정보를 숨기지 않는 보고에 집중합니다.

이 문서는 zenfmt 사용자를 위한 한국어 안내서입니다. 영어 문서와 같은 장 순서로 CLI, document representation, format reader, Markdown writer, resource limit, Zig와 Python API, browser 버전, server, benchmark를 설명합니다. code 이름, command line option, report code, API field, format 이름은 그대로 복사하고 검색할 수 있도록 영어 표기를 유지합니다.

설계 결정은 docs/zds/ 의 Zen Discussion, 즉 ZDS 기록에 보관됩니다. ZDS는 프로젝트의 설계 역사이므로 영어로 유지합니다. 이 안내서는 구현된 동작을 사용하는 방법을 설명하고, 어떤 선택의 배경이 필요할 때 해당 ZDS를 안내합니다.

zenfmt에는 전체 설계를 이끄는 세 가지 약속이 있습니다.

- 정보 손실을 숨기지 않고 보고하며 terminal과 인접 manifest에 모두 기록합니다.
- core는 개별 형식을 알지 않습니다. 형식마다 독립된 reader 또는 writer plugin이 있습니다.
- 모든 문서를 신뢰할 수 없는 입력으로 보고 archive, stream, nesting, 출력 크기, memory에 명확한 한계를 둡니다.

2 첫 변환의 전체 과정

2.1 설치와 가장 짧은 사용법

GitHub Releases에서 platform에 맞는 native archive를 내려받아 압축을 풀면 바로 실행할 수 있습니다. 이 실행 파일 하나에 CLI와 server가 함께 들어 있으며 Java, Python, npm, OCR, VLM, model file이 필요하지 않습니다. Python에서는 `zenfmt wheel`을 사용할 수도 있습니다.

macOS에서는 repository의 Homebrew cask도 사용할 수 있습니다. Apple Silicon이나 Intel에 맞는 같은 self contained archive를 GitHub Releases에서 직접 받습니다.

```
brew install --cask \
  https://raw.githubusercontent.com/insanai/zenfmt/main/packaging/homebrew/Casks/zenfmt.
rb
```

```
zenfmt report.docx
```

기본 설정에서는 입력 파일 옆에 `report.md`와 `report.md.zenfmt.json`이 생깁니다. 첫 파일은 Markdown artifact이고 두 번째 파일은 canonical JSON manifest입니다. 문서 본문만 stdout으로 보내려면 다음과 같이 실행합니다.

```
zenfmt report.docx --stdout
```

출력 경로를 지정하면 artifact, manifest, media resource를 하나의 transaction으로 게시합니다. 변환이 실패하거나 strict mode가 거부하면 절반만 작성된 target file을 남기지 않습니다.

```
zenfmt report.docx --output build/report.md
```

현재 binary에 포함된 reader와 writer는 다음 command로 볼 수 있습니다.

```
zenfmt --list-formats
```

2.2 manifest가 알려 주는 것

manifest는 변환 과정의 custody record입니다. source와 artifact의 이름, 판별된 format, plugin id, BLAKE3-256 digest와 함께 AST schema, document metadata, diagnostic report, plugin preservation data, facet summary가 들어 있습니다. 같은 input, version, option은 같은 byte를 만듭니다.

이후 변환에서 인접 manifest를 찾으면 zenfmt는 digest를 확인합니다. 일치하면 plugin data를 이어서 사용할 수 있습니다. 일치하지 않으면 stale manifest라고 보고하고 무시하며 오래된 metadata를 신뢰하지 않습니다.

2.3 diagnostic 읽기

diagnostic은 Elm compiler에서 배운 구조로 매번 네 가지 질문에 순서대로 답합니다.

- 무엇이 문제인가요.
- 어디에서 생겼나요.
- zenfmt는 그 때문에 무엇을 했나요.
- 사용자는 다음에 무엇을 할 수 있나요.

각 report에는 `docx.merged-cells-degraded` 같은 안정된 code도 있습니다. automation은 나중에 더 자연스럽게 다듬어질 수 있는 문장 대신 code를 기준으로 판단해야 합니다. loss tier에서 `degraded`는 단순화했지만 내용을 남긴 경우이고, `dropped`는 출력에서 내용이 사라진 경우입니다.

2.4 종료 code

code	뜻
0	artifact와 manifest를 commit했습니다. warning이나 정보 손실이 없다는 뜻은 아닙니다.
1	손상된 container, 잘못된 XML, reader가 찾은 모순 등으로 변환하지 못했습니다.
2	알 수 없는 format, 빠진 option, 덮어쓰기 거부 등 command 사용법에 문제가 있습니다.
3	입력이 명시된 resource limit을 넘었습니다.

2.5 형식 판별

확장자는 첫 hint일 뿐입니다. ZIP central directory의 part name, OpenDocument와 EPUB의 `mimetype`, CFB directory stream, PDF header, RTF signature도 확인합니다. 확장자와 내용이 다르면 내용을 우선하고 차이를 report합니다. 암호화된 문서는 분명하게 거부하고 조용히 건너뛰지 않습니다.

3 하나의 공통 document representation

모든 reader는 같은 document IR을 만들고 모든 writer는 그 IR에서 읽습니다. IR v2는 다음 layer로 나뉩니다.

- block과 inline AST가 paragraph, heading, list, table, link, image, text 같은 보이는 구조를 표현합니다.
- attributes가 identifier, class, key value attribute를 보관합니다.
- resource store가 내장 image 같은 binary resource를 digest로 관리합니다.
- facets가 style, tracked revision, page와 slide geometry, formula, provenance를 sparse stand off annotation으로 보관합니다.
- plugin preservation data가 특정 format에만 의미가 있지만 나중 round trip에 필요할 수 있는 정보를 보관합니다.

AST는 node마다 object를 할당하지 않고 preorder struct of arrays와 `u32` index를 사용합니다. subtree가 연속 범위이므로 탐색, 복사, limit check가 단순합니다. immutable transform은 결과를 다시 만들며 바뀌지 않은 연속 subtree를 array range로 복사할 수 있습니다.

IR에 정보가 있다는 이유로 writer가 표현할 수 있는 척하지 않습니다. 각 writer는 capabilities를 선언하고 lowering planner가 유지, 단순화, 제거 rule을 선택해 report를 만듭니다. strict mode는 파일을 쓰기 전에 같은 기준으로 결과를 판단할 수 있습니다.

4 Reader, Writer, Bundle

core는 conversion pipeline을 조정하지만 DOCX, PDF, Markdown 같은 형식 이름은 모릅니다. format plugin은 descriptor, reader 또는 writer, capabilities, limits, report catalog를 제공합니다. 기본 zenfmt bundle은 배포되는 plugin을 compile time에 조합합니다.

Reader의 책임은 parse 성공에서 끝나지 않습니다.

- container를 펼치기 전과 펼치는 동안 resource limit을 적용합니다.
- 지원하지 않거나 단순화하는 구조에 실행 가능한 안내를 담은 report를 냅니다.
- 구조와 index가 올바른 document IR을 만듭니다.
- media를 임의 path에 쓰지 않고 resource store에 전달합니다.
- preservation data를 자기 namespace에만 저장합니다.

application은 필요한 format만 담은 작은 bundle을 만들 수 있습니다. Zig project는 자체 filter도 선언할 수 있습니다. filter는 visitBlock 또는 visitInline 을 구현하고 project의 build.zig 를 통해 pipeline에 compile됩니다. 내장 scripting runtime이나 process 사이 serialization이 필요하지 않습니다.

5 Office와 container 형식

겉으로 다른 여러 형식은 결국 경계가 있는 container로 읽을 수 있습니다.

- DOCX, XLSX, PPTX, ODT, ODS, ODP, EPUB은 보통 ZIP package입니다.
- 예전 DOC, XLS, PPT는 CFB compound file을 사용합니다.
- RTF는 group이 중첩되는 token stream입니다.
- PDF는 indirect object, xref, content stream, font mapping의 조합입니다.

ZIP reader는 entry 개수, entry 하나의 펼친 크기, 전체 펼친 크기, compression ratio를 제한합니다. 규격에 필요한 part name만 읽으며 archive를 disk에 풀지 않습니다. CFB reader는 sector chain, directory tree, stream boundary를 검사하고 loop를 찾습니다. XML pull parser는 nesting, attribute, text 크기를 제한하며 external entity를 해석하지 않습니다.

spreadsheet의 formula, cell style, merged range는 Markdown으로 완전히 표현하기 어렵습니다. presentation의 position, speaker note, animation도 마찬가지입니다. reader는 보이는 text와 table structure를 가능한 한 남기고 나머지는 facet과 report로 설명합니다. PDF reader는 Zig로 구현한 native text extraction만 사용하며 OCR은 실행하지 않습니다. scan image 자체에 꺼낼 text가 없다면 그 한계를 그대로 보고합니다.

6 하나의 Writer

현재 release는 Markdown만 씁니다. 출력 범위를 좁게 유지하면 많은 reader가 확인 가능한 같은 target을 공유하고 loss policy도 더 분명해집니다.

Markdown writer는 CommonMark와 GFM table을 출력합니다. blank line, newline, escaping을 정규화하고 file 끝에는 newline 하나만 두며 trailing spaces를 만들지 않습니다. 같은 IR은 같은 byte를 만듭니다. embedded resource는 안정된 이름으로 인접 media directory에 게시되고 manifest에 기록됩니다.

직접 표현할 수 없는 구조에는 명시적인 lowering rule을 적용합니다. 예를 들어 merged cell 내용은 첫 cell에 남기고 covered position은 비웁니다. page break는 보통 dropped가 됩니다. 복잡한 style을 시각적으로 같은 Markdown인 것처럼 꾸미지 않습니다. 각 판단은 report로 남습니다.

strict mode로 허용할 손실을 선택할 수 있습니다.

```
zenfmt report.docx --strict=content
zenfmt report.docx --strict=structure
zenfmt report.docx --strict=exact
```

content 는 내용 손실을 거부합니다. structure 는 구조 손실까지 거부하고 exact 는 알려진 모든 손실을 거부합니다. 기본값 off 는 변환을 마치고 손실을 보고합니다. strict refusal은 publication 전에 일어납니다.

7 입력을 공격적인 data로 다루기

zenfmt는 개발용 machine이 우연히 처리할 수 있었던 크기를 안전 경계로 삼지 않습니다. 각 limit에는 이름, 기본값, report code, 해결 방향이 있습니다. 주요 경계에는 input bytes, archive entry 수, 펼친 bytes, compression ratio, XML depth, AST node 수, text bytes, resource bytes, output bytes, conversion time이 있습니다.

CLI에서는 안전한 범위 안에서 일부 limit을 조절할 수 있습니다. server는 request body, concurrency, rate limit, timeout 경계도 둡니다. limit에 도달하는 것은 정상적인 refusal이고 exit code 3이며 crash가 아닙니다.

browser profile은 tab의 memory와 실행 시간이 더 제한적이므로 보수적입니다. WebAssembly module에는 filesystem, network, clock, randomness host import가 없습니다. 변환은 worker에서 실행되어 main page가 계속 반응할 수 있습니다. cancel과 timeout은 현재 작업을 끝내며 page를 끝없이 기다리게 하지 않습니다.

출처를 모르는 파일을 다룰 때는 기본 limits를 유지하고 manifest와 reports를 확인하며 필요한 보장에 맞는 strict level을 선택하세요. 어떤 입력 하나가 성공했다는 이유로 바깥 service의 size나 time limit을 없애지 마세요.

8 내장과 Filter

8.1 Zig API

application에서 CLI와 같은 engine을 호출할 수 있습니다.

```
const zenfmt = @import("zenfmt");

const result = try zenfmt.convert(gpa, io, .{
    .input = .{ .path = "report.docx" },
    .output = .{ .memory = {} },
});
defer result.deinit(gpa);
```

정확한 type과 option은 사용 중인 package API를 확인하세요. caller가 allocator와 I/O capability를 제공하며 결과에는 artifact, manifest, resources, reports가 포함됩니다. binary를 줄이고 싶다면 `zenfmt_core.Bundle`에서 필요한 plugin만 조합할 수 있습니다.

Filter는 application에 compile되는 코드입니다. document node를 받아 새로 구성한 content를 반환합니다. 탐색 중 무효가 될 index를 저장하거나 builder를 건너뛰어 잘못된 relation을 만들지 마세요. filter가 만드는 손실도 공통 report mechanism으로 표현해야 합니다.

8.2 Python API

Python distribution은 typed dependency free wheel이며 해당 platform의 native bridge를 포함합니다. 변환 중에는 GIL을 해제합니다.

```
import zenfmt

conversion = zenfmt.convert("report.docx")
print(conversion.text)
for report in conversion.reports:
    print(report.code, report.problem)
```

str은 언제나 path입니다. memory의 내용은 bytes로 명확히 전달하고 이름도 제공해야 합니다.

```
conversion = zenfmt.convert(
    uploaded_bytes,
    name="upload.docx",
    to="markdown",
)
```

output을 지정하면 transaction publication을 사용하며 target 옆에 manifest와 media를 만듭니다. `zenfmt.Converter`는 immutable reusable policy value로 global config, environment, network를 읽지 않습니다. 실패는 `ConversionError`, `LimitExceededError`, `UnknownFormatError` 같은 작은 exception hierarchy로 나타나고 message는 CLI처럼 문제, 결과, 다음 단계를 설명합니다.

8.3 Browser API

release의 `wasm32-freestanding` bundle에는 module, ES module adapter, worker, TypeScript declaration이 들어 있습니다. adapter는 artifact ensemble을 반환하며 remote server로 fallback하지 않습니다. static origin에서 올바른 `application/wasm` type으로 제공하고 UI에도 timeout과 size policy를 두세요.

Web application은 `npm install @insanai/zenfmt` 으로 같은 dependency free browser distribution을 설치할 수도 있습니다. npm package에는 검사한 module, adapter, worker, declarations, capability contract가 들어 있습니다. native CLI와 server에는 Node나 npm이 필요하지 않습니다.

9 CLI 참고

자주 쓰는 command는 다음과 같습니다.

```
zenfmt INPUT
zenfmt INPUT --output OUTPUT
zenfmt INPUT --stdout
zenfmt INPUT --from docx --to markdown
zenfmt INPUT --strict=structure
zenfmt INPUT --preserve-facets
zenfmt --list-formats
zenfmt --help
zenfmt serve --help
```

stdin에는 file name hint가 없으므로 보통 `--from`이 필요합니다. artifact를 stdout에 써도 diagnostic은 stderr로 나와 shell pipeline에서 분리할 수 있습니다. 명시적으로 허용하지 않으면 기존 출력을 덮어쓰지 않습니다.

지원하는 입력 family는 다음과 같습니다.

분류	형식
word processing	docx, docm, doc, odt, rtf
spreadsheet	xlsx, xlsxm, xlsb, xls, ods, csv, tsv
presentation	pptx, pptm, ppsx, ppsm, ppt, pps, pot, odp
publishing	epub, pdf, html
markup	markdown, asciidoc, rst, text

limit 기본값과 diagnostic code 전체 목록은 영어 reference 장, `--help`, machine readable capabilities에서 확인할 수 있습니다. program은 자연어 문장이 아니라 안정된 code와 schema version에 의존해야 합니다.

10 벤치마크

benchmark는 속도를 재기 전에 correctness를 확인합니다. corpus manifest는 제삼자 파일마다 source, format, size, SHA-256을 기록하지만 license가 서로 다른 원본을 다시 배포하지 않습니다. `benchmarks/fetch_corpus.sh`가 같은 byte를 가져와 digest를 검증합니다.

서로 다른 질문은 나누어 보고합니다.

- Native CLI는 zenfmt, Docling parser only, AnyDoc, Pandoc의 coverage, wall time, CPU time, peak RSS를 비교합니다.
- Browser는 WebAssembly download size, cold ready, warm conversion, native artifact와의 parity를 측정합니다.
- Server는 장기 실행 zenfmt와 Tika Server의 startup, warm latency, peak RSS, concurrency별 throughput을 측정합니다.
- Output preservation은 format별 tool neutral oracle를 사용하고 여러 품질 판단을 하나의 점수로 합치지 않습니다.

Docling에서는 OCR, VLM, ASR, layout model, table model, enrichment, accelerator를 모두 끄고 parser backend만 사용합니다. zenfmt와 비슷한 보통 사양의 machine에서 AI pipeline이 아닌 parser path를 비교하기 위해서입니다. 지원하지 않는 file은 unsupported로 보이며 무한히 느린 값으로 만들거나 AI로 전환하지 않습니다.

ratio는 comparison tool을 zenfmt로 나눈 값입니다. 1.0보다 크면 이 측정에서 comparison tool이 해당 resource를 더 사용했다는 뜻입니다. 두 도구가 모두 성공한 shared files만 사용해 geometric mean을 구합니다. release, machine, benchmark lens가 다른 숫자는 섞지 않습니다. 결과는 맥락을 제공하는 참고 자료이며 모든 문서의 품질 순위가 아닙니다.

11 Server

같은 zenfmt 실행 파일에 server, OpenAPI document, database migration, web assets가 들어 있습니다.

```
zenfmt serve
curl -s -T report.docx \
  "http://127.0.0.1:8998/api/v1/convert?to=markdown"
```

open mode는 기본적으로 127.0.0.1:8998 에서만 listen하고 account 없이 stateless로 동작합니다. local tool이나 통제된 sidecar에 알맞습니다. /docs 는 내장 OpenAPI reference이고 /openapi.json 은 OpenAPI 3.1 contract입니다. 인증 없이 web root를 열면 사용할 수 없는 관리 화면 대신 공개 API 문서로 안내합니다. 공유 deployment에는 secure mode를 사용합니다.

```
zenfmt serve --secure --data-dir ./zenfmt-data
```

secure mode는 user, role, API key, session, audit log, administration UI를 추가합니다. operator가 data directory를 명시하며 다른 위치에 몰래 state를 만들지 않습니다. reverse proxy에서 TLS를 처리하고 접근 가능한 bind address를 제한하세요. API key는 한 번만 보여 주고 log에 credential이나 document body를 남기지 마세요.

conversion endpoint는 일반 response와 streaming을 지원합니다. client는 request size, timeout, retry policy를 두고 HTTP status와 structured error envelope로 실패를 판단해야 합니다. health endpoint는 생존 확인, metrics는 운영 관찰을 위한 것입니다. 민감한 filename 같은 high cardinality value를 metric label로 만들지 마세요.

12 더 읽을 자료

이 한국어판은 zenfmt로 변환하고 배포하는 데 필요한 실용 내용을 담았습니다. 영어판에는 IR layout, 형식별 container, diagnostic catalog, limit table, benchmark raw table을 더 깊게 다룬 내용이 있습니다. 설계 선택과 채택하지 않은 대안은 영어 ZDS에 남아 있습니다.

- project site와 browser converter: <https://insanai.github.io/zenfmt/>
- release와 self contained archive: <https://github.com/insanai/zenfmt/releases>
- source와 issue: <https://github.com/insanai/zenfmt>
- API reference: zenfmt serve 실행 후 `/docs`

결과가 중요한 workflow에서는 source, artifact, manifest를 함께 보관하고 reports를 확인하며 필요한 보장에 맞는 strict level을 선택하세요. zenfmt는 확실히 할 수 있는 일을 명확하게 수행하고 완전히 보존할 수 없는 부분도 똑같이 명확하게 알리는 것을 목표로 합니다.