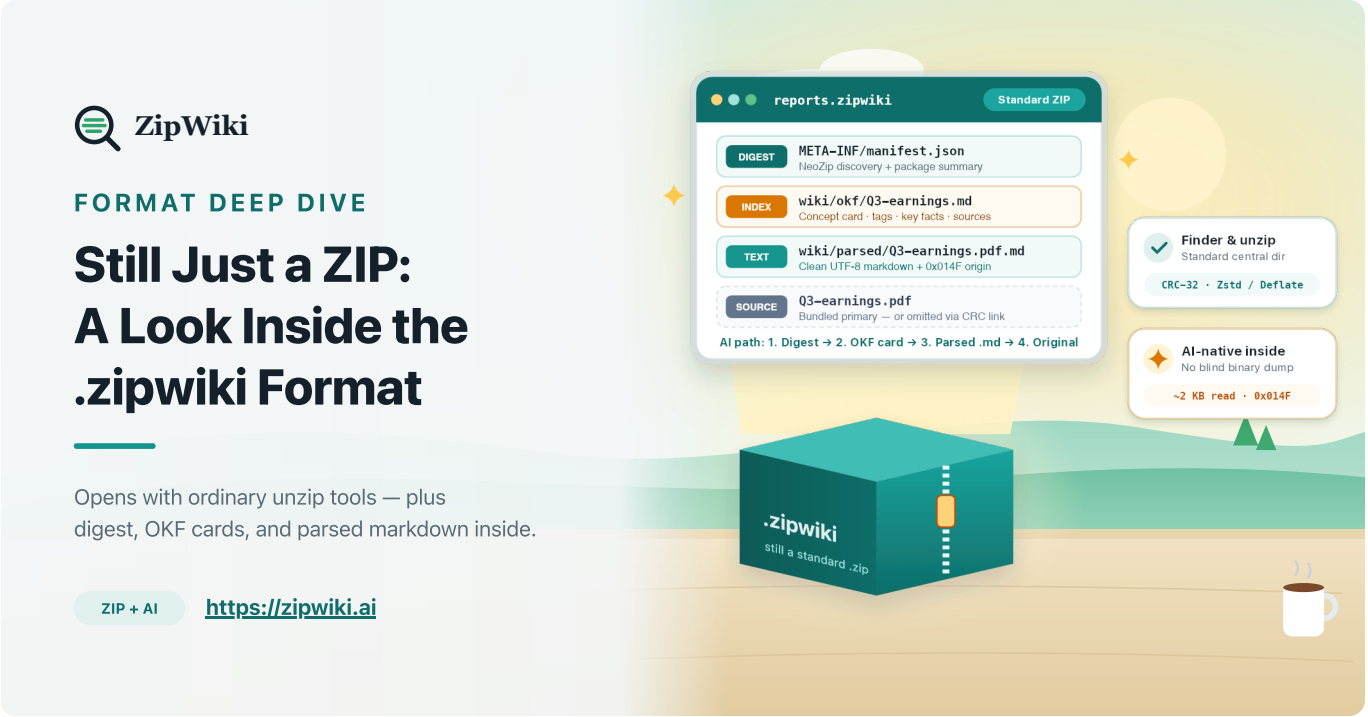


ZipWiki by NeoWare

Open it anywhere

If you rename a .zipwiki to .zip and double-click it, Finder, Explorer, and stock unzip utilities will list and extract its members. That is intentional. ZipWiki does not invent a proprietary container. A package is a standard ZIP that also carries an AI knowledge tree and a small amount of NeoZip metadata.

Agents and humans share the same file. You can email it, sync it, check it into git LFS, or park it on a USB drive. Whatever already moves ZIP files can move ZipWiki packages—no new archive format to adopt first.

What ZipWiki adds (and what it does not)

Stock ZIP tools see ordinary paths and payloads. ZipWiki-aware tools also understand two reserved top-level namespaces and optional Extra Fields on central-directory entries:

- **META-INF/** — package control. The important file is `META-INF/manifest.json`: format `"neozip"`, a `specVersion`, and an `ai` registry that points at the knowledge root.
- **wiki/** — the default AI root. Parsed markdown and OKF concept cards live here so agents can read text without reinventing PDF parsing on every question.
- **Your originals** — primary documents stay at the archive root (or under ordinary folders), unless you omit them and record a verified origin link instead.

Typical layout (simplified):

```
reports.zipwiki
├── META-INF/
│   └── manifest.json          # NeoZip + ZipWiki discovery (and package digest)
├── Q3-earnings.pdf            # optional primary (or omitted)
├── policy-handbook.docx
└── wiki/
    ├── parsed/
    │   ├── Q3-earnings.pdf.md
    │   └── policy-handbook.docx.md
    └── okf/
        ├── index.md           # bundle listing
        ├── Q3-earnings.md     # concept card
        └── policy-handbook.md
```

The AI path: digest → cards → markdown → original

A ZipWiki package is built for selective reading. An agent does not need to inflate every PDF to answer a question. The usual path is:

1. **Read the package digest:** Check the high-level summary inside `manifest.json` to verify relevance.
2. **Scan OKF concept cards:** Review lightweight entries under `wiki/okf/` with descriptions, tags, and pointers.
3. **Open parsed markdown:** Read only the relevant passages in `wiki/parsed/….md`.
4. **Inspect the original binary:** Fall back to the root primary only when signatures, exact styling, or pagination are required (or resolve the origin URI if omitted).

Walking this path keeps model context windows small and deterministic: heavy originals stay parked, and queries resolve against lightweight, portable text.

Parsed markdown is the working copy

Binary office formats and multi-column PDFs are awkward for models. At pack time, ZipWiki (via LiteParse locally, or hosted LlamaParse when you choose) writes clean UTF-8 markdown under `wiki/parsed/`, named after the primary path with `.md` appended.

Tables stay as markdown tables. Figures can land in a sibling `.assets/` folder with relative links from the parse. The parse is what agents should prefer for search and citation; the original remains the source of truth for human review when it is present.

OKF cards: the index, not another database

Open Knowledge Format (OKF) concept files live under `wiki/okf/`. Each primary gets a card (plus an optional `index.md` that lists the bundle). Cards carry structured frontmatter—type, description, key facts, tags, and sources that point back into the package.

For example:

```
---
type: Document
title: "Q3 2026 Earnings Report"
description: "Q3 gross margin expanded 140 bps YoY; recurring software revenue reached $
tags: [finance, revenue, q3]
sources:
  - resource: ../parsed/Q3-earnings.pdf.md
    description: Parsed markdown in this package
---
```

There is no hosted vector store required. The index travels with the ZIP. Share the file and you share the retrieval surface. That is why ZipWiki describes itself as portable document memory rather than “another cloud index.”

Origins: shrink the archive without losing provenance

Large corpora—session laws, SEC filings, clinical PDFs—often already live at stable public or internal URLs. Packing with omit-original can drop those binaries from the ZIP while keeping the parse and OKF card. Extra Field `0x014F` on the parse entry can record the origin URI, size, mtime, and a CRC-32 (or optional SHA-256) of the bytes that were parsed.

Later, `zipwiki origin --fetch` can download the remote file and verify it against that checksum. You get roughly the same AI-readable surface at a fraction of the archive size, without pretending the original never existed.

Still compatible with the ZIP you already know

Standard packages default to Zstandard (method 93). Use Deflate (method 8) via `--legacy` when you need stock OS unzip tools that cannot inflate Zstd. Binaries that do not compress further are stored uncompressed (method 0). Unicode paths, CRC-32 on every member, and ordinary central-directory listing all remain. Central directory entries may carry Extra Fields (`0x014E` for NeoZip integrity and `0x014F` for remote origin resolution). Tools that do not know what those mean simply ignore them—as the ZIP format specifies.

ZipWiki-aware CLIs and MCP servers go further: open, search, read parsed text, and resolve origins. The Application Note is the normative wire guide; this article is the map of what you will see when you peek inside.

ZipWiki is in Beta. Pack a folder, open the .zipwiki like any ZIP, then let an agent walk digest → OKF → parsed markdown. Same file. Far less of a blind box.
