Open Knowledge Format (OKF)

Version 0.1 (Draft) · An open specification formalizing the LLM-wiki pattern

Sam McVeety · Amir Hormati

Google Cloud, Data Cloud · June 2026

Mirror rendered from the official specification (github.com/GoogleCloudPlatform/knowledge-catalog, okf/SPEC.md) · Apache License 2.0

OKF is an open, human- and agent-friendly format for representing *knowledge* — the metadata, context, and curated insight that surrounds data and systems. It is designed to be authored by people, generated by agents, exchanged across organizations, and consumed by both.

The format is intentionally minimal: a directory of markdown files with YAML frontmatter. There is no schema registry, no central authority, and no required tooling. If you can `cat` a file, you can read OKF; if you can `git clone` a repo, you can ship it.

---

1. Motivation

The space of knowledge representation for AI agents is evolving quickly, and many incompatible conventions are emerging. OKF takes the position that knowledge is best represented in commonly accessible, established formats that are:

- **Readable** by humans without tooling.
- **Parseable** by agents without bespoke SDKs.
- **Diffable** in version control.
- **Portable** across tools, organizations, and time.

The format is minimally opinionated. It standardizes only the small set of structural conventions needed to make a knowledge corpus *self-describing* — anything beyond that is left to the producer.

Goals

1. Define a universal format that **enrichment agents** can write into.
2. Inform how **consumption agents** should read and traverse it.
3. Facilitate **exchange** of knowledge across systems and organizations.
4. Standardize the small number of **required** fields that must be present for content to be meaningfully consumed.

Non-goals

- Defining a fixed taxonomy of concept types.
- Prescribing storage, serving, or query infrastructure.
- Replacing domain-specific schemas (Avro, Protobuf, OpenAPI, etc.) — OKF *references* them; it does not subsume them.

2. Terminology

- **Knowledge Bundle** — A self-contained, hierarchical collection of knowledge documents. The unit of distribution.
- **Concept** — A single unit of knowledge within a bundle. Represented as one markdown document. May describe a tangible asset (a table, an API), an abstract idea (a metric, a business process), or anything in between.
- **Concept ID** — The path of the concept's file within the bundle, with the `.md` suffix removed. For example, `tables/users.md` has concept ID `tables/users`.
- **Frontmatter** — YAML metadata block delimited by `---` at the top of a markdown file.
- **Body** — Everything in the file after the frontmatter.
- **Link** — A standard markdown link from one concept to another, used to express relationships beyond the implicit parent/child hierarchy.
- **Citation** — A link from a concept to an external source that supports a claim in the body.

---

3. Bundle Structure

A bundle is a directory tree of markdown files. The directory structure is independent of the domain — producers organize concepts however makes sense for the knowledge being captured.

```
path/to/bundle/
├── index.md                    # Optional. Directory listing for progressive
disclosure.
├── log.md                      # Optional. Chronological history of updates.
├── <concept>.md                # A concept at the bundle root.
└── <subdirectory>/             # Subdirectories organize concepts into groups.
    ├── index.md
    ├── <concept>.md
    └── <subdirectory>/
        └── …
```

A bundle MAY be distributed as:

- A git repository (recommended — provides history, attribution, diffs).
- A tarball or zip archive of the directory.
- A subdirectory within a larger repository.

3.1 Reserved filenames

The following filenames have defined meaning at any level of the hierarchy and MUST NOT be used for concept documents:

Filename	Purpose
`index.md`	Directory listing. See §6.

Filename	Purpose
`log.md`	Update history. See §7.

All other `.md` files are concept documents.

Tags themselves remain a first-class concept — see the `tags` frontmatter field in §4.1. OKF does not specify a separate file format for aggregating documents by tag; producers that want a tag-browsing view can synthesize one at consumption time by scanning frontmatter.

---

4. Concept Documents

Every concept is a UTF-8 markdown file. It has two parts:

1. A **YAML frontmatter block**, delimited by `---` on its own line at the start of the file and a closing `---` on its own line.
2. A **markdown body**, containing free-form content.

4.1 Frontmatter

```
---
type: <Type name>                  # REQUIRED
title: <Optional display name>
description: <Optional one-line summary>
resource: <Optional canonical URI for the underlying asset>
tags: [<tag>, <tag>, …]            # Optional
timestamp: <ISO 8601 datetime>     # Optional last-modified time
# … other producer-defined key/value pairs
---
```

Required:

- `type` — A short string identifying the kind of concept. Consumers use this for routing, filtering, and presentation. Example values: `BigQuery Table`, `BigQuery Dataset`, `API Endpoint`, `Metric`, `Playbook`, `Reference`.

  Type values are **not** registered centrally. Producers SHOULD pick values that are descriptive and self-explanatory; consumers MUST tolerate unknown types gracefully (typically by treating them as generic concepts).

Recommended (in priority order):

- `title` — Human-readable display name. If omitted, consumers MAY derive a title from the filename.
- `description` — A single sentence summarizing the concept. Used by `index.md` generators, search snippets, and previews.
- `resource` — A URI that uniquely identifies the underlying asset the concept describes. Absent for concepts that describe abstract ideas rather than physical resources.

- `tags` — A YAML list of short strings for cross-cutting categorization.
- `timestamp` — ISO 8601 datetime of last meaningful change.

Extensions: Producers MAY include any additional keys. Consumers SHOULD preserve unknown keys when round-tripping and SHOULD NOT reject documents with unrecognized fields.

4.2 Body

The body is standard markdown. Producers SHOULD favor structural markdown — headings, lists, tables, fenced code blocks — over freeform prose, since structure aids both human reading and agent retrieval.

There are no required body sections. The following section headings have **conventional** meaning and SHOULD be used when applicable:

Heading	Purpose
`# Schema`	Structured description of an asset's columns/fields.
`# Examples`	Concrete usage examples, often as fenced code blocks.
`# Citations`	External sources backing claims in the body. See §8.

4.3 Example: a concept bound to a resource

```
---
type: BigQuery Table
title: Customer Orders
description: One row per completed customer order across all channels.
resource: https://console.cloud.google.com/bigquery?p=acme&d=sales&t=orders
tags: [sales, orders, revenue]
timestamp: 2026-05-28T14:30:00Z
---

# Schema

| Column        | Type      | Description                                |
|---------------|-----------|--------------------------------------------|
| `order_id`    | STRING    | Globally unique order identifier.          |
| `customer_id` | STRING    | Foreign key into [customers](/tables/customers.md). |
| `total_usd`   | NUMERIC   | Order total in US dollars.                 |
| `placed_at`   | TIMESTAMP | When the customer submitted the order.     |

# Joins

Joined with [customers](/tables/customers.md) on `customer_id`.

# Citations

[1] [BigQuery table schema](https://console.cloud.google.com/bigquery?
p=acme&d=sales&t=orders)
```

4.4 Example: a concept not bound to a resource

```
---
type: Playbook
title: Incident response — data freshness alert
description: Steps to triage a freshness alert on the orders pipeline.
tags: [oncall, incident]
timestamp: 2026-04-12T09:00:00Z
---

# Trigger

A freshness alert fires when `orders` lags more than 30 minutes behind
its expected SLA. See the [orders table](/tables/orders.md).

# Steps

1. Check the [ingestion job dashboard](https://example.com/dash).
2. …
```

---

5. Cross-linking

Concepts MAY link to other concepts using standard markdown links. Two forms are supported:

5.1 Absolute (bundle-relative) links

Begin with `/`, interpreted relative to the bundle root.

```
See the [customers table](/tables/customers.md) for the join key.
```

This is the **recommended** form because it is stable when documents are moved within their subdirectory.

5.2 Relative links

Standard markdown relative paths.

```
See the [neighboring concept](./other.md).
```

5.3 Link semantics

A link from concept A to concept B asserts a *relationship*. The specific kind of relationship (parent/child, references, joins-with, depends-on, etc.) is conveyed by the surrounding prose, not by the link itself. Consumers that build a graph view typically treat all links as directed edges of an untyped relationship.

Consumers MUST tolerate broken links — a link whose target does not exist in the bundle is not malformed; it may simply represent not-yet-written knowledge.

6. Index Files

An `index.md` file MAY appear in any directory, including the bundle root. It enumerates the directory's contents to support **progressive disclosure** — letting a human or agent see what is available before opening individual documents.

Index files contain no frontmatter. The body uses one or more sections, each grouping concepts under a heading:

```
# Section / Group Heading

* [Title 1](relative-url-1) - short description of item 1
* [Title 2](relative-url-2) - short description of item 2

# Another Section

* [Subdirectory](subdir/) - short description of the subdirectory
```

Entries SHOULD include the description from the linked concept's frontmatter. Producers MAY generate `index.md` automatically; consumers MAY synthesize one on the fly when none is present.

---

7. Log Files (optional)

A `log.md` file MAY appear at any level of the hierarchy to record the history of changes to that scope. The format is a flat list of date-grouped entries, newest first:

```
# Directory Update Log

## 2026-05-22
* **Update**: Added new BigQuery table reference for [Customer Metrics]
(/tables/customer-metrics.md).
* **Creation**: Established the [Dataplex Playbook](/playbooks/dataplex.md).

## 2026-05-15
* **Initialization**: Created foundational directory structure.
* **Update**: Added progressive-disclosure guidelines to the root [index](/index.md).
```

Date headings MUST use ISO 8601 `YYYY-MM-DD` form. Log entries are prose; the leading bold word (`**Update**`, `**Creation**`, `**Deprecation**`, etc.) is a convention, not a requirement.

---

8. Citations

When a concept's body makes claims sourced from external material, those sources SHOULD be listed under a `# Citations` heading at the bottom of the document, numbered:

```
# Citations

[1] [BigQuery public dataset announcement]
(https://cloud.google.com/blog/products/data-analytics/...)
[2] [Internal data quality runbook](https://wiki.acme.internal/data/quality)
```

Citation links MAY be absolute URLs, bundle-relative paths, or paths into a `references/` subdirectory that mirrors external material as first-class OKF concepts.

---

9. Conformance

A bundle is **conformant** with OKF v0.1 if:

1. Every non-reserved `.md` file in the tree contains a parseable YAML frontmatter block.
2. Every frontmatter block contains a non-empty `type` field.
3. Every reserved filename (`index.md`, `log.md`) follows the structure described in §6 and §7 respectively when present.

Consumers SHOULD treat all other constraints as soft guidance. In particular, consumers MUST NOT reject a bundle because of:

- Missing optional frontmatter fields.
- Unknown `type` values.
- Unknown additional frontmatter keys.
- Broken cross-links.
- Missing `index.md` files.

This permissive consumption model is intentional: OKF is meant to remain useful as bundles grow, get refactored, and are partially generated by agents.

---

10. Relationship to other formats

OKF is intentionally close to several established patterns:

- **LLM "wiki" repositories** that use markdown + frontmatter as agent-readable knowledge bases.
- **Personal knowledge tools** like Obsidian and Notion, which use hierarchical markdown with cross-links.
- **"Metadata as code"** approaches that store catalog metadata alongside source code rather than in a separate registry.

OKF differs primarily in being **specified** — pinning down the small set of rules needed for interoperability without dictating tooling.

---

11. Versioning

This document specifies OKF version **0.1**. Future revisions will be versioned in the form `<major>.<minor>`:

- A **minor** version bump introduces backward-compatible additions (new optional fields, new conventional section headings).
- A **major** version bump may make breaking changes (renaming required fields, changing reserved filenames).

Bundles MAY declare the OKF version they target by including `okf_version: "0.1"` in a bundle-root `index.md` frontmatter block (the only place frontmatter is permitted in an `index.md`). Consumers that do not understand the declared version SHOULD attempt best-effort consumption rather than refusing the bundle.

Appendix A — Minimal example bundle

```
my_bundle/
├── index.md
├── datasets/
│   ├── index.md
│   └── sales.md
└── tables/
    ├── index.md
    ├── orders.md
    └── customers.md
```

`datasets/sales.md`:

```
---
type: BigQuery Dataset
title: Sales
description: All sales-related tables for the retail business.
resource: https://console.cloud.google.com/bigquery?p=acme&d=sales
tags: [sales]
timestamp: 2026-05-28T00:00:00Z
---

The sales dataset contains transactional tables, including
[orders](/tables/orders.md) and [customers](/tables/customers.md).
```

`tables/orders.md`:

```
---
type: BigQuery Table
title: Orders
description: One row per completed customer order.
```

```
resource: https://console.cloud.google.com/bigquery?p=acme&d=sales&t=orders
tags: [sales, orders]
timestamp: 2026-05-28T00:00:00Z
---

# Schema

| Column        | Type      | Description                   |
|---------------|-----------|-------------------------------|
| `order_id`    | STRING    | Unique order identifier.      |
| `customer_id` | STRING    | FK to [customers](/tables/customers.md). |
| `total_usd`   | NUMERIC   | Order total in USD.           |

Part of the [sales dataset](/datasets/sales.md).
```