ADAC Specification Style Guide

Authoring Conventions for ADAC Core and Profile Documents

Version: 1.0 | **Status:** Stable | **Date:** 2026

Lead Author: John Vaden

Abstract

This document defines the editorial conventions, structural patterns, and formatting rules that govern every Archival Digital Asset Container (ADAC) specification document — from the core format specification to each domain profile extension. It ensures terminological consistency, predictable document layout, uniform JSON schema conventions, and clear normative language so that implementers encounter a single, coherent authoring voice across the entire ADAC family. All ADAC specification authors, technical editors, and profile extension designers are expected to follow the conventions set forth herein.

---

1 Introduction

1.1 Purpose

The ADAC ecosystem comprises a core format specification and an open-ended family of domain profile extensions (e.g., ADAC-Genealogy, ADAC-Legal). As the family grows, consistency across documents becomes critical for implementers, who must be able to navigate any ADAC specification with the same expectations regarding structure, terminology, and conventions. This style guide codifies the conventions that every ADAC specification author MUST follow, covering

document structure, normative language, terminology, JSON schema formatting, and versioning.

1.2 Scope

This guide applies to the following documents:

- The ADAC Format Specification (core).
- All official ADAC profile extension specifications (ADAC-Genealogy, ADAC-Legal, and future profiles).
- Companion documents such as API documentation, CLI documentation, and implementation guides.

This guide does NOT govern end-user application interfaces, marketing materials, or community-contributed tutorials unless they carry the official ADAC designation.

1.3 Audience

Specification authors, technical editors, profile extension designers, and anyone contributing normative text to the ADAC document family.

---

2 Normative Language

2.1 RFC 2119 Keywords

ADAC specifications use the keywords defined in RFC 2119 and RFC 8174: MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL. These keywords MUST appear in UPPER CASE when conveying normative force. When used in lower case (e.g., "the container should be compact"), the word carries its ordinary English meaning and is non-normative.

2.2 Conformance Clause

Every ADAC specification MUST include a Conformance section (typically §2) that satisfies the following requirements:

- References RFC 2119 and RFC 8174.
- Defines conformance levels if applicable (e.g., Minimal, Standard, Full for the core spec).
- States the normative status of each section.

2.3 Normative vs. Informative Text

Normative sections impose requirements on implementations. Conformance is measured against normative text.

Informative sections (notes, examples, appendices marked "informative") aid understanding but carry no conformance weight. An implementation that ignores informative text entirely remains conformant.

Examples embedded in normative sections are informative unless explicitly stated otherwise.

> **Convention**
>
> Use the prefix "**NOTE —**" for informative asides within normative sections. Use the prefix "**EXAMPLE —**" for illustrative code or data within normative sections.

---

3 Document Structure

3.1 Required Sections (Core Spec)

The core ADAC Format Specification MUST present sections in the following canonical order:

1. Title Block (title, version, status, copyright, date).
2. Abstract.
3. Table of Contents.
4. Introduction (Problem Statement, Solution, Design Principles).
5. Conformance.
6. Terminology.
7. Architecture.
8. Technical sections (Physical Container Format, Manifest, Core Metadata, Masters, Derivatives, Regions, Edits, XMP, Profiles, Provenance, Checksums, Fixity, Encryption, JSON Conventions, Validation, Media Types, Security, Interoperability).
9. Complete Container Example.
10. References (Normative and Informative).
11. Version History.

3.2 Required Sections (Profile Extensions)

Each ADAC profile extension specification MUST present sections in the following canonical order:

12. Title Block.
13. Abstract.
14. Table of Contents.

15. Introduction (purpose, target domain, relationship to core spec).
16. Conformance.
17. Terminology (domain-specific terms).
18. Profile Metadata File schema (the profile JSON file).
19. Linked Entity definitions (namespaced keys, schemas).
20. Region Annotation extensions.
21. Interoperability (interaction with core and other profiles).
22. Complete Profile Example.
23. References.
24. Version History.

3.3 Section Numbering

All sections use decimal numbering: §1, §1.1, §1.1.1 — maximum three levels. Deeper nesting indicates structural problems; refactor into sibling sections instead.

3.4 Cross-References

Internal cross-references use the format "see §N.N" or "as defined in §N.N". Never use page numbers; the specification is rendered as a web page. Profile documents cross-reference core spec sections as "see ADAC Core §N.N".

---

4 Terminology Conventions

4.1 Defined Terms

Every specification MUST include a Terminology section listing all defined terms. When a defined term first appears in normative text, it SHOULD appear in **bold**. Subsequent uses are plain text.

4.2 Consistent Naming

The following canonical terms MUST be used throughout all ADAC specifications. Alternative terms are prohibited in normative text.

Preferred Term	Avoid
container	"package", "archive", "bundle"
master file	"original", "source file", "primary"
derivative	"copy", "rendition", "version"
region	"zone", "area", "bounding box"
linked entity	"annotation payload", "domain data", "extension object"
profile	"plugin", "module", "extension" by itself
provenance log	"audit trail", "history", "changelog"
fixity	"integrity check", "hash verification"
manifest	"index", "catalog", "registry"
edit pipeline	"edit stack", "operations list", "transform chain"

4.3 Namespace Prefixes

Linked entity keys in profile documents use the format `namespace:type` (e.g., `genealogy:person`, `legal:exhibit`). The namespace MUST be a single lowercase word matching the profile's short name. The type MUST be a single lowercase

word. Multiple words use camelCase only when absolutely necessary (e.g., `genealogy:sourceAttachment`).

---

5 JSON Conventions

5.1 Schema Language

All JSON schemas in ADAC specifications are described using JSON Schema (Draft 2020-12). Schema definitions MUST specify `$schema`, `$id`, `type`, and `description` at minimum.

5.2 Property Naming

All JSON property names use camelCase: `adacVersion`, `createdOn`, `sourceMasterId`, `linkedEntities`. Never use snake_case, PascalCase, or kebab-case for property names.

Acronyms of three or more letters follow camelCase: `sha256Hash`, `xmpSidecar`. Two-letter acronyms remain uppercase only at the start: `id`, `ID` (as standalone).

5.3 Date-Time Format

All date-time values MUST use ISO 8601 with UTC offset: `2025-01-15T10:30:00Z`. Date-only values use `YYYY-MM-DD`. Never use Unix timestamps, locale-formatted strings, or ambiguous formats.

5.4 Identifiers

- **Container IDs:** UUID v4, lowercase, hyphenated (e.g., `550e8400-e29b-41d4-a716-446655440000`).
- **Internal IDs** (master, derivative, region): kebab-case descriptive strings (e.g., `master-001`, `region-face-01`).

- **Profile-defined IDs:** Follow the same convention with a namespace prefix when disambiguation is needed.

5.5 Enumerated Values

Enumerated string values use kebab-case: `web-preview`, `chain-of-custody`, `litigation-hold`. Provide a closed list in the schema using `enum`. New values require a specification revision.

5.6 Null and Absent Fields

Absent keys and explicit `null` are semantically equivalent: "no value provided." Implementations MUST NOT distinguish between the two. Required fields MUST NOT be null or absent.

5.7 Extensibility

Profile-specific JSON files MAY include additional properties not defined in the core schema. Core JSON files MUST NOT include undefined properties unless a profile explicitly extends them via the `linkedEntities` mechanism.

5.8 Example Formatting

JSON examples in specification text MUST satisfy all of the following requirements:

- Pretty-printed with 2-space indentation.
- Syntactically valid (parseable by any compliant JSON parser).
- Annotated with a caption or title.
- Marked as informative unless explicitly stated otherwise.

---

6 Profile Wrapper Requirements

6.1 File Location

Profile metadata files reside at `metadata/profiles/<profileName>.json`. The file name MUST match the profile's registered short name in lowercase (e.g., `genealogy.json`, `legal.json`).

6.2 Wrapper Schema

Every profile JSON file MUST include a wrapper object with the following minimum required fields:

Field	Type	Description
`profileName`	string	The registered profile identifier (e.g., `"ADAC-Genealogy"`).
`profileVersion`	string	The profile specification version (e.g., `"1.0"`).
`schemaRef`	string (URI)	A reference to the profile's JSON Schema.

Beyond these three required fields, the profile defines its own domain-specific properties.

6.3 Additive-Only Rule

Profiles MUST be purely additive. A profile MUST NOT:

- Redefine, override, or constrain any core-spec field.
- Alter the semantics of core metadata properties.
- Require modifications to `manifest.json` beyond listing the profile file in the `metadata.profiles` array.

A non-profile-aware reader MUST be able to open and fully process any ADAC container by ignoring profile files entirely.

6.4 Multi-Profile Coexistence

When multiple profiles coexist in a single container, their namespaced linked-entity keys prevent conflicts. Each profile specification MUST document the namespaces it uses and MUST NOT use another profile's registered namespace.

---

7 Writing Style

7.1 Voice and Tense

Use third-person, present-tense, active voice: "The manifest lists all master files." Avoid first person ("we define"), future tense ("will contain"), and passive voice ("is defined by") where possible.

7.2 Sentence Structure

Prefer short, declarative sentences. State one requirement per sentence. If a sentence contains more than one MUST or SHALL keyword, split it into separate sentences.

7.3 Lists

Use numbered lists for ordered sequences (steps, precedence). Use bulleted lists for unordered sets (properties, options). Each list item begins with a capital letter and ends with a period if it is a complete sentence.

7.4 Code and Markup

Inline code references use monospace formatting: `manifest.json`, `adacVersion`, `master/`. File paths use forward slashes. JSON property names in running text are always rendered in monospace.

7.5 Abbreviations and Acronyms

Spell out on first use with the abbreviation in parentheses: "Archival Digital Asset Container (ADAC)". After first use, use the abbreviation alone. Common acronyms (JSON, UUID, SHA, ZIP, URI, URL, ISO) need not be spelled out.

---

8 Interoperability Guidance

8.1 Standards Alignment

Each ADAC specification MUST document which external standards it aligns with and how. The core spec aligns with:

- **Dublin Core** — descriptive metadata.
- **PREMIS** — preservation metadata.
- **XMP** — sidecar metadata.
- **ISO 8601** — date and time representation.

Profile specifications SHOULD document their own domain-specific standards alignments in the same manner.

8.2 Graceful Ignorance

All ADAC specifications MUST uphold the principle of **graceful ignorance**: a reader that does not understand a profile, linked entity namespace, or optional extension MUST be able to skip it without loss of core functionality. No optional component may cause a conformant reader to fail.

8.3 Round-Trip Fidelity

Implementations that read and rewrite ADAC containers MUST preserve all fields they do not explicitly modify — including profile data and linked entities from profiles they do not understand. Lossless round-tripping of unrecognized data is a core interoperability requirement.

---

9 Versioning Rules

9.1 Version Numbering

ADAC specifications use a two-part version number: **MAJOR.MINOR** (e.g., 1.0, 1.1, 2.0). No patch component is used. MAJOR increments signal breaking changes. MINOR increments signal additive, backward-compatible changes.

9.2 Compatibility Guarantees

A container marked `adacVersion: "1.x"` MUST be fully readable by any implementation supporting ADAC 1.0. New optional fields added in minor versions MUST NOT break existing readers.

9.3 Version History Section

Every specification MUST end with a Version History table containing the following columns:

Column	Description
Version	The version number (e.g., 1.0).
Date	The publication date of that version.
Summary of Changes	A concise description of what changed.

9.4 Profile Versioning

Profile extensions maintain their own version numbers independent of the core spec version. A profile MUST declare the minimum core spec version it requires (e.g., "Requires ADAC Core ≥ 1.0").

---

10 References

10.1 Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words," BCP 14, RFC 8174, May 2017.

[JSON-SCHEMA] Wright, A., Andrews, H., Hutton, B., "JSON Schema: A Media Type for Describing JSON Documents," Draft 2020-12.

[ISO8601] International Organization for Standardization, "Date and time — Representations for information interchange," ISO 8601:2019.

[DCMI] Dublin Core Metadata Initiative, "Dublin Core Metadata Element Set, Version 1.1," DCMI Recommendation, 2012.

[PREMIS] Library of Congress, "PREMIS Data Dictionary for Preservation Metadata," Version 3.0, 2015.

10.2 Informative References

[ZIP-APPNOTE] PKWARE, Inc., "ZIP File Format Specification," APPNOTE.TXT, Version 6.3.10.

[XMP] International Organization for Standardization, "Extensible metadata platform (XMP) specification," ISO 16684-1:2019.

[GEDCOM7] The Church of Jesus Christ of Latter-day Saints, "The FamilySearch GEDCOM Specification," Version 7.0, 2021.

[PREMIS-IG] Library of Congress, "PREMIS Implementation Guidelines," 2013.

Version History

Version	Date	Summary of Changes
1.0	2026	Initial publication. Establishes authoring conventions for ADAC Core and all profile extension specifications.
