ADAC Metadata Schema 1.0

Version: 1.0
Status: Draft for implementation
Applies to: ADAC 1.0 containers (document-level and master-level XMP)

---

1. Purpose and Scope

The **ADAC Metadata Schema 1.0** defines the **XMP-level metadata model** for ADAC containers. It specifies:

- The ADAC core XMP namespace
- Document-level ADAC properties
- Master-level ADAC properties for XMP sidecars
- Data types and constraints
- Versioning and profile selection
- Recommended mappings to existing XMP/PDF fields

This schema is format-agnostic but is intended primarily for use in:

- PDF document-level XMP packets
- XMP sidecars in `metadata/xmp/` within ADAC containers

Profile-specific schemas (e.g., ADAC-Legal, ADAC-Genealogy, ADAC-Preservation) are defined in separate documents.

---

2. Namespace and Prefix

2.1 Core ADAC Namespace

- **Namespace URI:** `http://adac.io/schema/1.0/`
- **Preferred prefix:** `adac`

Example declaration:

```
xmlns:adac="http://adac.io/schema/1.0/"
```

The namespace URI is opaque and stable; it is not required to resolve to a document at runtime.

---

3. Data Model and Types

Unless otherwise specified:

- All properties are simple properties (not structs) except where arrays are explicitly defined.
- String values are Unicode text.
- Timestamps use ISO-8601 UTC (`YYYY-MM-DDThh:mm:ssZ`).

XMP types used:

- **Text**
- **Date**
- **URI**
- **Boolean**
- **Bag of Text** (for lists)

Properties are grouped by **scope**:

- **Document-level**: describe the ADAC container as a whole.
- **Master-level**: describe a specific master entry and its XMP sidecar.

---

4. Core Document-Level ADAC Properties

4.1 `adac:adacVersion`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Exactly one
- **Description:** ADAC specification version.
- **Constraint:** MUST be `"1.0"` for ADAC 1.0 containers.
- **JSON mapping:** `manifest.json` → `adacVersion`

4.2 `adac:containerId`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Exactly one

- **Description:** Globally unique identifier for the ADAC container. UUID (RFC 4122) is RECOMMENDED.
- **Constraint:** SHOULD match `manifest.json` → `id` and `metadata/core.json` → `id`.
- **JSON mapping:** `manifest.json` → `id`, `metadata/core.json` → `id`

4.3 `adac:primaryProfile`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Zero or one
- **Description:** The primary ADAC profile that defines the main behavior, icon, or interpretation of the container. The value SHOULD correspond to the `profileType` declared in the profile's root fields (e.g., `"genealogy"`, `"legal"`, `"preservation"`). Well-known profile types are defined in companion profile specifications.
- **JSON mapping:** SHOULD correspond to one of the profiles listed in `metadata/profiles/`.

4.4 `adac:profileVersion`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Zero or one
- **Description:** Version of the selected primary ADAC profile.
- **JSON mapping:** Profile-specific; SHOULD match the `profileVersion` field declared at the root of the corresponding profile file.

4.5 `adac:profiles`

- **Scope:** Document-level
- **Type:** Bag of Text
- **Cardinality:** Zero or one
- **Description:** Authoritative list of all profile files applied to this container.
- Values: Container-relative paths to `metadata/profiles/*.json`.
- **Constraint:** SHOULD match the `metadata.profiles` array in `manifest.json`. The paths in both locations MUST be identical.
- **JSON mapping:** `manifest.json` → `metadata.profiles`

Example:

```
<adac:profiles>
  <rdf:Bag>
    <rdf:li>metadata/profiles/genealogy.json</rdf:li>
    <rdf:li>metadata/profiles/legal.json</rdf:li>
```

```
  </rdf:Bag>
</adac:profiles>
```

4.6 `adac:documentType`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Zero or one
- **Description:** High-level classification of the primary artifact represented by the container (e.g., `"Census Page"`, `"Legal Filing"`, `"Birth Record"`).
- **JSON mapping:** MAY be mirrored from `metadata/core.json` → `description` or a profile-specific field.

4.7 `adac:ingestTimestamp`

- **Scope:** Document-level
- **Type:** Date
- **Cardinality:** Zero or one
- **Description:** Timestamp when the artifact was first ingested into an ADAC container. Represents the initial creation of the container, not subsequent edits.
- **JSON mapping:** MAY correspond to `metadata/core.json` → `administrative.digitizedOn`. Note: a dedicated `administrative.ingestedOn` field may be added to the core JSON schema in a future revision; until then, `digitizedOn` is the closest available mapping.

4.8 `adac:sourceApplication`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Zero or one
- **Description:** Name and version of the application that created or last materially updated the ADAC container.
- **JSON mapping:** SHOULD correspond to `manifest.json` → `createdBy`.

4.9 `adac:hashAlgorithm`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Zero or one
- **Description:** Name of the cryptographic hash algorithm used to compute the Merkle roots stored in `adac:immutableMasterRoot` and `adac:mutableStateRoot`.
- **Constraint:** For ADAC 1.0, MUST be `"SHA-256"` (matching the `"sha256"` value used in `provenance/checksums.json`, normalized to uppercase for XMP consistency).

- **JSON mapping:** `provenance/checksums.json` → `algorithm`

4.10 `adac:immutableMasterRoot`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Zero or one
- **Description:** Merkle root representing the immutable master files in the container. Covers only the `master/` directory.
- **Constraint:** SHOULD match `manifest.json` → `immutableMasterRoot`. If either `adac:immutableMasterRoot` or `adac:mutableStateRoot` is present, both MUST be present.
- **JSON mapping:** `manifest.json` → `immutableMasterRoot`

4.11 `adac:mutableStateRoot`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Zero or one
- **Description:** Merkle root representing all mutable content (all non-master files) in the container.
- **Constraint:** SHOULD match `manifest.json` → `mutableStateRoot`. If either `adac:mutableStateRoot` or `adac:immutableMasterRoot` is present, both MUST be present.
- **JSON mapping:** `manifest.json` → `mutableStateRoot`

4.12 `adac:workflowState`

- **Scope:** Document-level
- **Type:** Text
- **Cardinality:** Zero or one
- **Description:** High-level workflow state of the container (e.g., `"Draft"`, `"InReview"`, `"Approved"`, `"Filed"`, `"Published"`).
- **JSON mapping:** No direct mapping exists in ADAC 1.0 core JSON. This field is reserved for future use when a `workflowState` field is formally added to `metadata/core.json` → `administrative`. Implementations that use this XMP property SHOULD also write the value to an appropriate profile-specific or application-defined field in the container's JSON metadata for consistency.

4.13 `adac:provenanceLogPath`

- **Scope:** Document-level
- **Type:** URI (Text)
- **Cardinality:** Zero or one

- **Description:** Container-relative path to the provenance log file. The canonical path defined by the ADAC 1.0 specification is `provenance/log.json`.
- **Constraint:** SHOULD match `manifest.json` → `metadata.provenanceLog` when present.
- **JSON mapping:** `manifest.json` → `metadata.provenanceLog`

4.14 `adac:checksumsPath`

- **Scope:** Document-level
- **Type:** URI (Text)
- **Cardinality:** Zero or one
- **Description:** Container-relative path to the checksum manifest. The canonical path defined by the ADAC 1.0 specification is `provenance/checksums.json`.
- **Constraint:** SHOULD match `manifest.json` → `metadata.checksums` when present.
- **JSON mapping:** `manifest.json` → `metadata.checksums`

---

5. Master-Level ADAC Properties (XMP Sidecars)

These properties are intended for XMP sidecars stored in `metadata/xmp/` and associated with individual master entries in the manifest. The XMP sidecar file for a given master is named using the master file's base name with a `.xmp` extension (e.g., `master/master_0001.tif` → `metadata/xmp/master_0001.xmp`), and its path is recorded in the master entry's `xmp` field in `manifest.json`.

5.1 `adac:masterId`

- **Scope:** Master-level
- **Type:** Text
- **Cardinality:** Exactly one (for sidecars)
- **Description:** Identifier of the master entry in `manifest.json` to which this XMP sidecar belongs.
- **Constraint:** MUST match a `masters[i].id` value in `manifest.json`.
- **JSON mapping:** `manifest.json` → `masters[i].id`
- **Usage:** REQUIRED for master-level XMP sidecars; SHOULD NOT be used at document level.

5.2 `adac:role`

- **Scope:** Master-level
- **Type:** Text
- **Cardinality:** Zero or one

- **Description:** Role of the master within the container (e.g., `"primary-recto"`, `"primary-verso"`, `"supplemental"`). These values align with the `role` field on master entries in `manifest.json`.
- **JSON mapping:** `manifest.json` → `masters[i].role`
- **Usage:** RECOMMENDED for master-level XMP sidecars when a multi-master container uses role distinctions; SHOULD NOT be used at document level.

5.3 Optional Master-Level Linkage

Implementations MAY also include:

- `adac:containerId` — to link the sidecar back to its container.
- `adac:adacVersion` — to indicate the ADAC version for the master context.

These mirror the document-level properties but are scoped to the sidecar for robustness when the sidecar is extracted from the container.

---

6. Relationship to ADAC JSON Metadata

The ADAC Metadata Schema is descriptive and redundant with respect to the JSON structures inside the container. When conflicts arise between XMP values and the container's internal JSON, the **container's internal JSON is authoritative**; XMP SHOULD be updated to match on the next save.

Implementations SHOULD maintain consistency across the following mappings:

XMP Property	JSON Location
`adac:containerId`	`manifest.json` → `id`, `metadata/core.json` → `id`
`adac:adacVersion`	`manifest.json` → `adacVersion`
`adac:immutableMasterRoot`	`manifest.json` → `immutableMasterRoot`
`adac:mutableStateRoot`	`manifest.json` → `mutableStateRoot`
`adac:profiles`	`manifest.json` → `metadata.profiles`
`adac:provenanceLogPath`	`manifest.json` → `metadata.provenanceLog`
`adac:checksumsPath`	`manifest.json` → `metadata.checksums`
`adac:sourceApplication`	`manifest.json` → `createdBy`
`adac:ingestTimestamp`	`metadata/core.json` → `administrative.digitizedOn`
`adac:workflowState`	*(no ADAC 1.0 core JSON equivalent — reserved for future revision)*

XMP Property	JSON Location
`adac:masterId`	`manifest.json` → `masters[i].id`
`adac:role`	`manifest.json` → `masters[i].role`

---

7. Recommended Standard Namespace Mappings

To maximize interoperability with existing tools, ADAC implementations SHOULD map `metadata/core.json` fields into standard XMP namespaces in the same XMP packet.

7.1 Dublin Core (dc:)

From `metadata/core.json`:

- `title` → `dc:title`
- `creator` → `dc:creator`
- `description` → `dc:description`
- `subject` → `dc:subject` (Bag of Text)
- `source` → `dc:source`
- `format` → `dc:format`
- `language` → `dc:language`
- `coverage` → `dc:coverage`

7.2 Rights and Licensing

From `metadata/core.json` → `rights`:

- `rights.statement` → `dc:rights` and/or `xmpRights:UsageTerms`
- `rights.holder` → `xmpRights:Owner` (Bag)
- `rights.license` → `cc:license` (URI if applicable; SPDX identifiers are RECOMMENDED per the ADAC 1.0 specification)
- `rights.accessRestrictions` → MAY be mapped to `xmpRights:UsageTerms` or a profile-specific field.

These mappings ensure that standard tools (e.g., Adobe Acrobat, Windows Explorer, DAM systems) can read key metadata without being ADAC-aware.

---

8. Use in PDF Document-Level XMP

When embedding ADAC metadata into a PDF:

- ADAC properties are added to the **document-level XMP packet**.
- They coexist with standard schemas:
  - `dc:` (Dublin Core)
  - `xmp:` (XMP Basic)
  - `pdf:` (PDF Schema)
  - `xmpMM:` (Media Management)
  - `xmpRights:` (Rights)
  - `cc:` (Creative Commons)

ADAC does not replace or override existing fields; it augments them.

A minimal recommended ADAC block includes:

- `adac:adacVersion`
- `adac:containerId`
- `adac:primaryProfile`
- `adac:profiles`
- `adac:hashAlgorithm`
- `adac:immutableMasterRoot`
- `adac:mutableStateRoot`

---

9. Versioning

- This document defines **ADAC Metadata Schema 1.0**.
- Future schema versions:
  - MUST use a new namespace URI (e.g., `http://adac.io/schema/2.0/`).
  - MAY reuse the `adac` prefix if context is unambiguous.
- ADAC 1.0 containers SHOULD use only the `http://adac.io/schema/1.0/` namespace for core ADAC properties.

---

10. Conformance

An implementation **conforms** to ADAC Metadata Schema 1.0 if:

1. It declares the ADAC namespace as specified in Section 2.1.

2. It reads and preserves all `adac:*` properties it does not understand (round-trip preservation).
3. It writes properties according to the types and constraints defined above.
4. It maintains consistency between ADAC XMP properties and the container's internal JSON metadata, treating the internal JSON as authoritative in the event of conflict.
5. For master-level sidecars, it sets `adac:masterId` to a valid master `id` from `manifest.json`.
6. When either `adac:immutableMasterRoot` or `adac:mutableStateRoot` is written, both MUST be written together.

This schema operates within the container conformance levels defined in the ADAC 1.0 Format Specification:

Container Level	**XMP Expectation**
Minimal	`adac:adacVersion` and `adac:containerId` MUST be present if XMP is used.
Archival	All applicable document-level properties SHOULD be populated; master-level sidecars SHOULD be present for each master entry.