Cross-File Associations (CFAs): A Lightweight, Decentralized Model for Expressing File Relationships*

Daniel Hardman

December 2025

Abstract

Folders, archive files, source code repositories, and similar containers are the dominant mechanisms for grouping digital artifacts today [1,2]. Containers offer clarity and durability, but they also impose rigid structure and become fragile when data moves across systems or administrative boundaries. Tagging systems provide a more flexible and decentralized alternative, but at the cost of precision and formal interpretability; prior work documents the semantic ambiguity and weak governance that arise in collaborative tagging and folksonomies [3–5].

This paper introduces *Cross-File Associations* (CFAs), a lightweight mechanism for declaring relationships among files and file-like objects. Like tags, CFAs allow logical groupings that do not depend on physical containment or central authority. Unlike tags, CFAs make relationships explicit and structured in ways that software can interpret reliably. CFAs are simple enough for informal use, yet expressive enough to support automation and reasoning. The paper develops the conceptual foundations of CFAs, illustrates their use through concrete examples, and situates them among existing organizational techniques.

Keywords: file systems, metadata, information organization, decentralized identifiers, cryptographic identifiers, digital signatures

1. Introduction

Imagine Amadeus, a fictional composer who writes musical scores for films and video games. Over the course of his career, he produces a large body of work that spans decades, genres, and collaborators. Some compositions are revised, rearranged, or expanded. Some are coauthored. Rights and licenses vary. File formats change as tools evolve.

Like most people, Amadeus relies on containment to organize his work. Projects live in folders. Deliverables are zipped and emailed. Archives accumulate across laptops, external drives, and cloud services. Over time, the structure frays. Files are misplaced. Copies diverge. Context is lost. Relationships that once seemed obvious become hard to reconstruct.

Containment is intuitive and widely supported, but it is brittle. A file can belong to only one folder at a time unless it is duplicated or indirectly linked. Changing the meaning of a hierarchy requires reorganizing it. When files cross administrative or technical boundaries, containment-based meaning often disappears.

*daniel.hardman@gmail.com

Tagging systems address some of these limitations. A photo can be tagged with multiple labels. A document can be annotated with keywords. Tags are flexible and decentralized, but they are also ambiguous. A tag such as "draft" or "vacation" can mean different things to different people, or even to the same person at different times. Tag vocabularies can be standardized, but doing so reintroduces centralization and governance overhead. Prior empirical and theoretical work on folksonomies has documented these tradeoffs in flexibility, precision, and governance [3–5].

This paper explores an alternative: a way to express relationships among files that is decentralized like tagging, yet more precise and formally interpretable. The mechanism is called a *Cross-File Association* (CFA).

2. Basic concepts

In this paper, a *file* is any digital artifact with identifiable content and a reference that is sufficiently stable to support comparison and association over time. This includes traditional files in a file system, but also database records, email messages, web resources, cryptographic objects, and other digital items. Some files are static; others change over time.

A *container* is any construct that groups files together: a directory, an archive, an email with attachments, a database table, or a cloud storage bucket. Files and containers are not mutually exclusive categories.

A *cross-file association* (CFA) is a declared relationship among two or more files. When a file participates in such a relationship, we say that it *binds* the CFA; binding is a declarative act and does not imply correctness, endorsement, or authority. A file may bind zero, one, or many CFAs.

Each CFA is identified by a *CFA identifier*. Files that bind the same identifier are understood to participate in the same association. CFA identifiers need only be unique within the context in which they are interpreted; they need not be human-readable.

Some CFAs distinguish between files that provide a logical foundation for the relationship and files whose interpretation depends on that foundation. These are called *directed CFAs*. Files that provide the logical foundation for a directed CFA are called *pre files*. Files whose interpretation depends on a *pre file* within a directed CFA are called *co files*.

Other CFAs treat all bound files as peers; these are called *common CFAs*. A file may be a *pre file* in one CFA and a *co file* in another.

CFAs may be declared in ways that modify file content or in ways that rely entirely on external conventions. These approaches are introduced later in the paper.

3. Conceptual foundations of cross-file associations

Cross-File Associations are designed to express relationships among digital artifacts without relying on containment, centralized registries, or domain-specific ontologies. This section clarifies the conceptual commitments that shape CFA design and delimit its scope.

3.1 Files, containers, and abstraction level

CFAs operate at a level of abstraction above file systems and storage hierarchies. They do not replace folders, repositories, or databases. Instead, they provide a way to express relationships that cut across these structures.

In contemporary workflows, digital artifacts are routinely copied, synchronized, renamed, and redistributed across administrative and technical boundaries. In such settings, organizational meaning derived solely from

physical or logical containment is fragile. Because CFAs are not tied to physical location, they remain meaningful when files move or are replicated across systems.

This location-independence is a design objective rather than a claim of superiority. CFAs coexist with containment-based organization and rely on it where appropriate.

3.2 Associations as first-class relationships

A CFA is a relationship, not a container. Conceptually, CFAs form edges in a graph whose nodes are files. Treating relationships as *first-class* in this sense means that they are explicitly representable and inspectable, rather than inferred indirectly from structure.

Binding a CFA is a statement that a relationship is asserted to exist, independent of whether that assertion is accepted, validated, or contested. It is not a claim about ownership, authorship, correctness, or legitimacy. CFAs intentionally avoid embedding such judgments.

CFAs are lightweight by design. They aim to capture common, durable relationships rather than exhaustively model provenance, workflow, or semantic meaning. This boundary is intentional and distinguishes CFAs from provenance systems that seek comprehensive causal and temporal representation [6,7]. More expressive systems may be layered on top of CFAs, but CFAs are not a substitute for them. Unlike tags, CFAs distinguish symmetric grouping from asymmetric dependence, preventing common errors such as treating derivatives, signatures, or reviews as interchangeable members of an undifferentiated set.

3.3 Directed and common associations

Many familiar file relationships are asymmetric. A document may be signed, translated, reviewed, or transformed into another format. In these cases, the derived artifact depends on the original for its interpretation.

Directed CFAs make this dependence explicit by distinguishing *pre files* from *co files*. The direction reflects semantic dependence, not importance, authority, or value.

Other relationships are naturally symmetric. Multiple photographs of the same scene or alternative encodings of the same recording may form a logical set without a single foundational element. These relationships are modeled as common CFAs.

A file may serve as a *pre file* in one CFA and as a *co file* in another.

3.4 Declaring associations without central authority

Any party may assert that a file participates in a CFA. This allows relationships to be declared unilaterally, after the fact, and without coordination among all involved actors.

The ability to assert relationships unilaterally is important in practice. Files are often annotated, commented on, reviewed, or transformed by parties other than their original authors, sometimes long after initial creation. CFAs support these post hoc and third-party relationships without requiring permission or shared infrastructure.

CFAs distinguish between *assertion*—the act of declaring a relationship—and *authority*, which arises from external social or technical mechanisms. This distinction is well established in trust management and identity systems, where assertions must be evaluated independently of the authority that validates or relies on them [8,9]. Some CFA identifiers merely name a relationship. Others are constructed in ways that allow a party to demonstrate control over an identifier's namespace. This distinction supports both informal annotation and stronger governance models without requiring a single global authority.

3.5 Internal and external binding strategies

CFAs may be declared using conventions embedded within file content or through external conventions such as naming patterns. These strategies differ only in how the relationship is expressed; they do not alter the relationship's semantics.

Internal strategies tend to be more robust across containers because the declaration travels with the file. Work in digital preservation and metadata standards has long emphasized the portability advantages of embedded, standardized metadata over container-local conventions [10,11]. External strategies are simpler and often already used informally, but their meaning is typically container-dependent.

CFAs treat these approaches as complementary. A relationship may be declared using different strategies in different contexts, or redundantly, to increase resilience.

3.6 Scope and limits

CFAs are not a provenance system, an access control mechanism, or a semantic ontology. They do not enforce correctness, resolve disputes, or guarantee completeness. They provide a minimal vocabulary for stating that digital artifacts are related in specific, recurring ways.

The remainder of the paper moves from these abstract commitments to concrete mechanisms and examples.

4. Simple examples

The examples in this section are illustrative rather than exhaustive. They demonstrate how CFAs make relationships legible to humans and software, rather than enumerating all possible uses or strategies.

4.1 Common association declared internally

Amadeus wants to mark each file he creates as part of his overall career corpus. He generates a random identifier and embeds it as metadata in each file. All files that bind this identifier belong to the same logical set, regardless of where they are stored.

This association is common: no file is foundational. The mechanism is internal: the association travels with the file.

4.2 Directed association declared externally

Amadeus sends a client several email attachments: two audio files, an invoice, and a digital signature over the invoice. He wants to make it clear that the signature applies only to the invoice.

Using a simple naming convention, he binds the invoice and its signature into a directed CFA. Software that recognizes the convention can infer the relationship and warn when one file is missing. Humans can often infer the same meaning at a glance.

This example illustrates how even minimal external conventions can express relationships more clearly than containment alone.

5. Declaring CFAs in practice

Cross-File Associations are expressed through *declaration strategies*: conventions by which a file signals its participation in a particular association. In this paper, *strategy* refers exclusively to a convention for expressing a CFA, not to a workflow, policy, or governance mechanism.

A strategy determines how a relationship is made legible to humans and software; it does not define the semantics of the relationship itself.

CFA design intentionally supports multiple strategies instead of prescribing a single canonical mechanism. This plurality reflects a core design commitment: relationships among files already exist in practice, expressed through ad hoc conventions, naming patterns, and metadata. CFAs aim to recognize and formalize these practices incrementally, not to replace them wholesale.

5.1 Strategies as design tradeoffs

At a high level, CFA strategies can be understood along two illustrative axes.

The first distinguishes **external** strategies from **internal** ones. External strategies rely on conventions outside file content—typically filenames or container-local context. Internal strategies embed declarations within the content or metadata of a file itself.

The second concerns **deployability versus expressiveness**. Simple strategies are easy to adopt and often require no new tools, but they carry limited semantic precision. More expressive strategies enable stronger interpretation and automation, but require formats that support structured metadata or inline annotations. Tradeoffs of this kind are a recurring theme in software architecture and systems design [12].

These axes are illustrative rather than exhaustive. They serve as a heuristic for understanding design tradeoffs, not as a complete taxonomy.

5.2 External strategies: container-local conventions

External strategies express associations through naming or placement conventions that are meaningful within a shared container. Common examples include sidecar files, shared filename stems, or numeric infixes.

These strategies are highly legible to humans, require no modification of file content, and are already widely used informally. However, they are inherently container-dependent. The meaning of the association can be inferred only when the relevant files are observed together.

CFAs treat this fragility as acceptable in contexts where containment is already meaningful—such as email attachments, temporary work folders, or device-local workflows. The goal is not to eliminate such practices, but to make their semantics explicit enough for software to assist rather than interfere.

5.3 Internal strategies: portable declarations

Internal strategies embed CFA declarations within file content or metadata. Because the declaration travels with the file, internal strategies are portable across containers and systems.

These strategies are particularly well-suited to decentralized settings, where no single container can be assumed to persist, and to relationships among files that are never colocated or are authored by different parties at different times.

The tradeoff is complexity. Not all file formats support arbitrary metadata, and embedding declarations may require format-specific knowledge or tooling. CFAs avoid mandating a particular syntax at this level, instead defining the minimal information that must be conveyed.

5.4 Identifier choice and meaning

Every CFA is distinguished by an identifier that names the association. CFAs place few constraints on identifier form. Identifiers may be human-readable names, randomly generated values, URLs, or cryptographically derived identifiers.

The identifier does not, by itself, confer authority. Authority, where it exists, arises from the properties of the identifier scheme—such as cryptographic control or content addressing—not from CFAs as such. Any party may assert that a file participates in a CFA identified by a given value.

By separating the expression of relationships from their validation, CFAs allow informal annotation and strong governance models to coexist without forcing premature commitments.

5.5 Partial adoption and graceful degradation

CFAs are designed to function as hints rather than mandates. A tool may recognize some strategies and ignore others. A user may declare CFAs in one context and omit them in another. Files within the same association may be unevenly annotated.

This partial adoption is a requirement for real-world uptake, not a failure mode. When CFAs are recognized, they enable software to make better-informed choices. When they are not recognized, files remain ordinary files. No correctness or access guarantees are assumed or enforced.

This property allows CFAs to be layered gradually onto existing ecosystems, improving behavior opportunistically without breaking established workflows.

6. Discussion and future work

Cross-File Associations are intentionally modest in scope. They do not attempt to solve every problem related to digital organization, provenance, or trust. This section situates CFAs relative to adjacent approaches, examines potential failure modes, and outlines directions for future research and standardization.

6.1 Relationship to existing systems

CFAs occupy a conceptual space between several familiar mechanisms for organizing digital artifacts.

Containers such as folders, archives, and repositories express membership implicitly through physical or logical inclusion. They are efficient and widely understood, but brittle when artifacts move across boundaries or must participate in multiple groupings. CFAs do not replace containers; they make explicit some of the relationships that containers can only imply.

Tags and labels provide flexible, many-to-many categorization. However, tags are typically untyped, weakly governed, and semantically ambiguous. CFAs retain the decentralization of tagging while introducing explicit structure—such as directionality and cardinality—that enables more reliable software interpretation.

Version control systems model relationships among files over time, particularly derivation and succession. CFAs do not attempt to encode temporal history or branching structure. Instead, they allow files produced inside or outside version control systems to declare durable relationships that persist even when detached from their original repositories.

Provenance and knowledge graph models aim for expressive, often exhaustive representations of causality, authorship, and process [6,7]. CFAs are deliberately not a substitute for such systems. They provide a lightweight substrate on which richer models may be layered, but they avoid requiring comprehensive

schemas, ontologies, or centralized registries. CFAs are not intended to compete with provenance systems, but with the informal naming conventions and ad hoc metadata that practitioners already rely on in their absence.

Across these comparisons, CFAs can be understood as a *design pattern* rather than a competing system: a minimal way to state that files are related in recurring, interpretable ways.

6.2 Authority, conflict, and adversarial cases

Because CFAs are decentralized, they permit unilateral assertions. Any party may claim that a file participates in a given association. This raises obvious concerns about false, misleading, or conflicting declarations.

CFAs do not attempt to resolve such disputes. They make no claims about correctness, ownership, or legitimacy. Instead, they expose assertions so that tools and users can evaluate them in context. The design deliberately favors visibility of contested assertions over implicit silence, on the assumption that inspectable disagreement is preferable to unexamined inference.

Different identifier choices can mitigate some of these risks. For example, identifiers that support cryptographic control allow a party to demonstrate stewardship over a namespace, while content-addressed identifiers can make tampering evident. However, these mechanisms remain optional. CFAs separate the *expression* of relationships from their *validation*.

Conflicts among CFAs—such as competing claims about derivation or succession—are expected. In many domains, disagreement is inherent. CFAs provide a vocabulary for surfacing these disagreements without imposing a resolution model.

6.3 Failure modes and limits

CFAs are vulnerable to several practical failure modes:

- Associations may be incomplete, stale, or asymmetric.
- External strategies may break when files are renamed or separated.
- Internal strategies may be stripped by tools that rewrite content or metadata.
- Software may recognize some strategies and ignore others.

These limitations are acknowledged design constraints rather than oversights. CFAs are intended to degrade gracefully. When declarations are missing or ignored, files continue to function normally.

Importantly, CFAs do not enforce invariants. They do not guarantee that a signature matches its target, that a derivative is faithful, or that a successor supersedes its predecessor. They merely assert that such relationships are claimed to exist.

6.4 Research and standardization directions

Several avenues for future work follow naturally from this proposal.

Empirical studies could examine how users naturally group files and whether CFA-aware tools improve task outcomes such as retrieval, error prevention, or comprehension.

Interoperability profiles could define subsets of strategies and identifier types suitable for particular domains, such as email, scientific publishing, or digital archiving.

Formalization efforts might specify grammars, comparison rules, or canonical encodings, potentially leading to standardization through community or formal bodies. Such efforts should remain modular to allow partial adoption.

Finally, CFAs invite exploration of how lightweight relational metadata interacts with cryptographic content addressing and decentralized identity systems, particularly in long-lived, multi-party ecosystems.

7. Conclusion

As digital artifacts proliferate across tools, platforms, and organizations, the limitations of containment-based organization become increasingly apparent. CFAs offer a lightweight, decentralized way to make relationships explicit without imposing heavy infrastructure or rigid schemas.

The primary contribution of CFAs is a minimal, composable vocabulary that makes existing file relationships explicit and interoperable without requiring shared infrastructure, schemas, or authority. By treating relationships as first-class and supporting gradual adoption, CFAs enable both humans and software to reason more effectively about collections of related files.

References

[1] Dinneen, J. D., and Julien, C.-A. *The ubiquitous digital file: A review of file management research.* Journal of the Association for Information Science and Technology, 72, 6 (2021), 711–728. https://doi.org/10.1002/asi.24464

[2] Bergman, O., Beyth-Marom, R., and Nachmias, R. *Folder versus tag preference in personal information management.* Journal of the Association for Information Science and Technology, 64, 9 (2013), 1855–1870. https://doi.org/10.1002/asi.22875

[3] Trant, J. *Studying social tagging and folksonomy: A review and framework.* Journal of Digital Information, 10, 1 (2009).

[4] Spiteri, L. F. *The structure and form of folksonomy tags: The road to the public library catalog.* Information Technology and Libraries, 26, 3 (2007), 13–25. https://doi.org/10.6017/ital.v26i3.3279

[5] Golder, S. A., and Huberman, B. A. *The structure of collaborative tagging systems.* Journal of Information Science, 32, 2 (2006), 198–208. https://doi.org/10.1177/0165551506062337

[6] W3C. *PROV-Overview: An overview of the PROV family of documents.* W3C Recommendation, 2013. https://www.w3.org/TR/prov-overview/

[7] W3C. *PROV-DM: The PROV data model.* W3C Recommendation, 2013. https://www.w3.org/TR/prov-dm/

[8] Blaze, M., Feigenbaum, J., and Lacy, J. *Decentralized trust management.* In *Proceedings of the IEEE Symposium on Security and Privacy* (1996), 164–173. https://doi.org/10.1109/SP.1996.10012

[9] Grassi, P. A., Garcia, M. E., and Fenton, J. L. *Digital Identity Guidelines: Federation and Assertions.* NIST Special Publication 800-63C-4, 2025. https://doi.org/10.6028/NIST.SP.800-63c-4

[10] Caplan, P. *Understanding PREMIS.* Library of Congress, 2017. https://www.loc.gov/standards/premis/understanding-premis.pdf

[11] Dappert, A., and Farquhar, A. *Using METS, PREMIS and MODS for archiving e-journals.* D-Lib Magazine, 14, 9/10 (2008). https://doi.org/10.1045/september2008-dappert

[12] Bass, L., Clements, P., and Kazman, R. *Software Architecture in Practice*, 3rd ed. Addison-Wesley, 2013.