CAAP

Contextual Accountability Attribution Protocol

Introduction and Relationship to DDRP

Quantum Inquiry | quantuminquiry.org

1. The Problem DDRP Leaves Open

DDRP answers three questions about a document: whether obligation-creating language appeared, whether it was detectable at the time of production, and whether each obligation was structurally resolved or left open. When DDRP completes its work, it produces an immutable artifact: a hash-stable, machine-readable record of what the document contained.

That record answers the structural question. It does not answer the accountability question.

> *Knowing that an obligation existed and was logged is not the same as knowing who acted on it, when, under what authority, and against which version of the record.*

In low-stakes environments, this gap is acceptable. In regulated, legal, or organizationally consequential environments, it is not. A contract review that produces a compliance matrix proves that obligations were found. It does not prove who reviewed them, what decision was made, or whether the decision was made against the current version of the artifact or a superseded one.

This is the problem CAAP is designed to close. Not by modifying DDRP, but by sitting above it.

2. What CAAP Is

CAAP — the Contextual Accountability Attribution Protocol — is a separate protocol layer that records who acted on a DDRP artifact, when, and under what authority. It does not extract obligations. It does not interpret them. It does not re-evaluate whether the extraction was correct. Those functions belong to DDRP and are not repeated.

CAAP operates exclusively on completed DDRP artifacts. Its inputs are the artifact hash, a reference to the source document, and an event: a timestamped, actor-attributed, authority-declared action taken against a specific obligation in the artifact. Its output is an append-only event log that ties every action to a cryptographically verified version of the record it was taken against.

CAAP is	An attribution layer. It records provenance chains, not compliance conclusions.

Three properties define CAAP's design:

- Append-only: Events are never modified or deleted. The log grows in one direction. Past states are permanently accessible.
- Hash-bound: Every event references the artifact_hash of the DDRP artifact it was taken against. If the artifact changes, a new hash is required. An event logged against an outdated artifact is permanently marked as such.
- Authority-declared: Every event requires an actor and an authority_basis field. Logging an action without declaring the basis on which it was taken is structurally prohibited.

3. The Two-Layer Architecture

DDRP and CAAP are not versions of the same system. They are adjacent layers in a two-layer architecture, separated by a published interface contract. Understanding why the separation is enforced requires understanding what happens when it is not.

3.1 What Separation Prevents

When extraction and attribution are handled by the same system, a structural vulnerability emerges: the system can be used to retroactively justify a resolution rather than prove one occurred. If the record of what obligations existed and the record of what was done about them are both mutable and co-located, the distinction between observation and interpretation collapses. The audit trail becomes a narrative document rather than an evidentiary one.

The separation document describes this failure mode directly: if the layers bleed, the system collapses into narrative governance. Narrative governance is the condition in which compliance is demonstrated by writing a convincing story about what happened, rather than by producing a verifiable record of what happened.

Separating the layers structurally forecloses this. DDRP produces the record. CAAP references it by hash only. Neither can modify the other. The boundary between what was found and what was done is cryptographically enforced.

3.2 The Stack

LAYER B	**CAAP**	Contextual Accountability Attribution Protocol — Who acted, when, under what authority, against which artifact
INTERFACE	**Artifact Contract v1.0**	Hash-bound JSON schema. Read-only reference. CAAP consumes. DDRP produces. No shared state.
LAYER A	**DDRP**	Deterministic Document Review Protocol — What obligations exist, whether they are structurally resolved, and whether the record is inspectable

The interface between the layers is the DDRP Artifact Interface Contract v1.0. This contract defines the exact JSON schema that a DDRP artifact must conform to before CAAP will accept it. Artifacts that fail validation are rejected entirely. There is no partial acceptance.

3.3 Directional Flow

Direction	What Moves
DDRP → CAAP	Immutable artifact (hash-stable JSON). Read-only. CAAP consumes it.
CAAP → DDRP	Nothing. CAAP has no write access to DDRP. No bidirectional modification.
CAAP → Log	Append-only event records referencing artifact_hash. Never touching source artifact.
DDRP → Log	Nothing. DDRP is unaware of CAAP. It produces artifacts and stops.

DDRP is explicitly designed to be unaware of CAAP. This is not an oversight. It is the architectural guarantee that DDRP's extraction function remains uncontaminated by attribution concerns. A DDRP implementation does not need to know whether CAAP exists to produce a valid artifact.

4. The Interface Contract

The only formal connection between DDRP and CAAP is the DDRP Artifact Interface Contract v1.0. This document defines the schema a DDRP artifact must satisfy, the hash semantics CAAP uses to verify integrity, and the immutability rules CAAP enforces on artifacts it has accepted.

4.1 Required Artifact Fields

Before CAAP accepts an artifact, it validates the following fields:

Field	Purpose
artifact_schema_version	Structural compatibility check. CAAP rejects unsupported versions without coercion.
artifact_hash	SHA-256 hash of the canonical artifact payload. CAAP recomputes and must match.
document_id	Stable identifier of the source document being reviewed.
extraction_timestamp	UTC timestamp of when the DDRP extraction occurred.
canon_rules_version	Version of the lexical extraction rules used. Informational for attribution context.
obligations	Array of obligation objects produced by DDRP. May be empty.
absent_fields	Explicitly recorded structural elements that were not found. May be empty.

4.2 Hash Semantics

The artifact_hash is a SHA-256 hash computed over the entire artifact payload, excluding the hash field itself. The hash is computed over canonical JSON: UTF-8 encoded, lexicographically sorted keys, no insignificant whitespace. This deterministic serialization requirement ensures that two valid implementations produce identical hashes from identical data.

If the computed hash does not match the declared artifact_hash, CAAP rejects the artifact. A hash mismatch is treated as evidence of mutation, not a recoverable error. This is the enforcement mechanism for the immutability rule: any change to artifact content, however minor, produces a different hash and therefore a different artifact identity.

4.3 The Immutability Rule

Once a DDRP artifact is generated, it is immutable. CAAP enforces this rule by treating the artifact as read-only and referencing it only by hash. Any change to obligation structure requires a new DDRP extraction and a new artifact. CAAP events logged against the previous artifact remain permanently associated with that version, not silently migrated to the new one.

Key Rule	CAAP MUST NOT modify artifact content. CAAP MUST NOT rewrite obligation fields. CAAP MUST treat the artifact as read-only in all operations.

5. Why Adjacency, Not Integration

The natural engineering instinct when two systems exchange data is to integrate them: share a codebase, share a database, share a UI. DDRP and CAAP are explicitly designed to resist this instinct, and the reason is not technical fastidiousness. It is a functional requirement.

5.1 The Contamination Risk

Every obligation DDRP identifies is a factual observation: this language appeared at these coordinates in this document at this time. That observation has no author. It has no intent. It is a structural record.

Every event CAAP logs is an attributed action: this actor, under this authority, took this action against this obligation. That action has an author. It has intent. It is a provenance record.

These two categories of record must remain separate because they answer different questions and are subject to different forms of challenge. A challenge to a DDRP record is a challenge to whether the extraction was correct. A challenge to a CAAP record is a challenge to whether the actor had authority, acted in time, or acted against the right version of the artifact. Merging the layers merges these challenges and makes both harder to resolve.

5.2 The Stability Argument

DDRP is finalized extraction infrastructure. It does not change to support CAAP logic. This is not a limitation of DDRP; it is a design decision that protects the integrity of every artifact DDRP has ever produced. If DDRP were modified to accommodate CAAP, every artifact generated before the modification would have been produced by a different version of the protocol. The evidentiary chain for those artifacts would require qualification.

By treating DDRP as stable and building CAAP as a separate consumer, the historical artifact record remains clean. CAAP can evolve, add event types, strengthen integrity rules, and develop a UI layer without touching the foundation on which it rests.

5.3 The Scope Argument

DDRP answers the structural question. CAAP answers the accountability question. These are not sub-problems of a single larger question. They are questions that arise at different points in the lifecycle of a professional obligation, require different evidence, and implicate different parties.

A contractor using DDRP to review an RFP is asking: what does this document require of me? A contracting officer using CAAP to review the event log is asking: who reviewed this artifact, when, and did they have the authority to resolve this obligation? These are sequential questions, not simultaneous ones. The two-layer architecture reflects that sequence structurally.

6. What CAAP Is Not

Precision about what CAAP does requires equal precision about what it does not do.

CAAP does NOT...	Why this matters
Interpret obligations	Interpretation belongs to the human reviewer. CAAP records that a reviewer acted, not whether their interpretation was correct.
Re-evaluate extraction correctness	Challenging whether DDRP found the right obligations is a challenge to the DDRP artifact, not to CAAP. The systems are not co-responsible for extraction quality.
Modify obligation structure	The artifact is immutable. CAAP cannot add, remove, or change obligations after the artifact is generated.
Add inferred fields	CAAP logs only what is explicitly declared by an actor. Nothing is inferred or generated.
Perform lexical analysis	CAAP has no lexicon. It has no extraction function. It reads artifacts it did not produce.
Provide compliance conclusions	CAAP records that actions were taken. It does not assess whether those actions constitute compliance. That determination belongs to the reviewing authority.

7. The Development Sequence

CAAP is built in phases, with a strict ordering that enforces architecture before implementation. No CAAP code is written until the interface contract is defined and the DDRP repository is tagged and locked.

Phase	Goal
Phase 0 — Interface Contract	Define the DDRP Artifact Interface Contract v1.0. No code. Architecture only. DDRP tagged at v0.2.0-extraction-final.
Phase 1 — Minimal Core (No UI)	Command-line proof of integrity. Upload artifact, verify hash, store read-only, create append-only event log. Actor and authority_basis required on every event.

Phase 2 — Event Integrity Layer	Artifact hash validation, optional re-hash verification, version locking, event-to-obligation reference validation, scope assertion field.
Phase 3 — Minimal UI	Separate UI layer. Shows artifact list, event timeline, actor map, authority chain, artifact version diff. No inline modification.
Phase 4 — Hard Integrity Rules	Reject events without artifact hash, reject past event modification, enforce append-only, artifact replacement only via new upload with new hash.

This sequence is not arbitrary. Phase 0 exists because the interface contract is the only connection between DDRP and CAAP. If that contract is defined after code is written, the code defines the contract by default, and the architectural boundary becomes an implementation detail rather than a design rule.

8. Summary

DDRP and CAAP address adjacent but distinct problems in the management of professional obligations under document.

Protocol	Question Answered
DDRP	What obligations exist in this document, were they detectable, and were they structurally resolved?
CAAP	Who acted on the obligations in this artifact, when, under what authority, and against which version of the record?

DDRP produces the evidentiary foundation. CAAP produces the provenance chain. Neither is sufficient without the other in environments where both the what and the who of obligation management are subject to external scrutiny.

The relationship between them is enforced by three constraints: the interface contract defines the handoff, the hash mechanism verifies integrity, and the separation rule prevents either layer from contaminating the other. Together these constraints produce a two-layer accountability architecture in which every claim is traceable to either a structural observation or an attributed action, and the two are never conflated.

> *DDRP proves the record was made. CAAP proves the record was acted on. The difference is the difference between an observation and a decision.*