emanuel
celano
04.08.2026
11:36:19
GMT+02:00

EVIDE ANCHOR

Working Paper

*Protocol name: **EVIDE ANCHOR** - "EVIDE ANCHOR preserves declared authority." At the same level as GLM, FEDIS, and DAPI: ANCHOR is the name of the protocol, not of the underlying technical primitive, which remains Declaration (Section 3). The protocol name is normative. The acronym expansion (**A**uthority **N**exus for **C**ontext, **H**olding & **O**rigin **R**ecords) is explanatory only and does not form part of the normative specification.*

Informatica in Azienda - EVIDE Framework
Bologna, 4 August 2026

Dr. Emanuel Celano - Founder & Chief Architect at EVIDE

Table of Contents

1. Origin and Problem

On July 31, 2026, RedHotCyber documented a real incident: an AI agent (Claude Opus 5, "Ultracode" mode), connected directly to the production Supabase database with autonomous access to a GitHub repository, executed prisma migrate diff pointing the --shadow-database-url parameter at the production database itself. Prisma drops the shadow database before reapplying migrations: the tool treated production as a disposable environment, deleting all 22 tables of the project (two of them unrecoverable, since the legacy migration set did not include them at all). The agent noticed the anomaly and reported it to the developer. The article itself attributes the cause not only to the agent's error but to the access architecture: no isolation, no restricted role, no manual confirmation required.

Starting hypothesis: the reconstruction of the incident cannot be reduced to the agent's error alone. The boundary between production and development was not technically enforced nor preserved through an independent evidentiary declaration - it existed only as an implicit expectation. An evidentiary boundary can never be made "impassable": EVIDE preserves the declaration of a boundary, it does not enforce it (Decision 2). In the absence of technically enforced limits and independently preserved declarations, the subsequent reconstruction of the operational perimeter becomes inevitably dependent on the systems and narratives of the parties involved.

Design method: an 8-phase map (Doctrine, Boundary, Temporality, Evidence, Events, Reconstruction, Interoperability, Implementation), preceded by a Phase 0 - writing the forensic questions that an investigator, a court-appointed expert, or a judge would ask after an incident, before even designing the schema. The same method that already produced External Artifacts: first the question, then the answer, then the model - never the other way around.

2. Doctrine (Decisions 1-3)

Decision 1 - the problem it solves:

> *EVIDE independently preserves the operational declarations that define the declared perimeter within which an AI agent was declared authorized to operate at a given time, enabling the subsequent evidentiary reconstruction of the declared context without verifying its correctness or attributing responsibility.*

Decision 2 - what it does NOT do (seven constraints, not six - the seventh is what prevents the system from filling an absence on its own):

1. It does not verify whether declarations correspond to operational reality, nor does it compare what was declared with what was observed - that comparison is for later investigators, never for EVIDE.
2. It does not enforce or apply any declared policy, limit, or privilege.
3. It does not block, authorize, or prevent any agent operation.

4. It does not judge whether the declared authority was legitimate, sufficient, or actually held by whoever exercised it.
5. It does not attribute responsibility or establish fault.
6. It does not assume that the absence of a declaration equals the absence of the corresponding constraint - it is a neutral fact (no one declared it), not a judgment (therefore it did not exist).
7. It never infers, completes, or interprets declarations that do not exist.

Cross-cutting principle (proposed during this design process, applicable to all of EVIDE, not only to this extension):

> *"EVIDE preserves exclusively explicit declarations. It does not infer, complete, or interpret missing declarations." Already true implicitly for Artifacts and Evidentiary Continuity; here made explicit for the first time.*

Decision 3 - evidentiary value (deliberately kept separate from Decision 1: they answer different questions - why the extension exists, versus what it concretely proves years later). **Corrected after external review** that flagged an overclaim: the earlier version stated that a declaration "already existed... before the agent began operating" - but declared_at is a client-supplied value, not verified; the objective fact that EVIDE produces is intake_timestamp_utc (the moment of acquisition, not the declared moment). A developer could compose a Declaration with a backdated declared_at after an incident, and EVIDE would faithfully preserve it without being able to turn that date into independent proof of its original existence:

> *It allows demonstrating, independently and years later, that a specific operational declaration, with that exact content and attributed to that subject, was acquired by EVIDE at a given moment. The originally declared moment remains preserved as declared_at and can be compared against intake_timestamp_utc and any available external time references. Establishing whether the declaration truly existed before the contested action remains an evidentiary assessment - never an assumption made by EVIDE.*

This test - *"ten years from now, will this field allow something specific to be proven?"* - was applied to every field proposed during the design process. Two proposals were discarded precisely because they failed it (a field for EVIDE-computed comparison between declared and observed; a dedicated server-side timestamp, when intake_timestamp_utc already existed at the record level).

3. The Primitive: Declaration and Reference

This is not a structure specific to this extension. Declaration is an architectural primitive of EVIDE. It introduces a primitive distinct from Event/Decision (decision_record, intervention), Artifact/Evidence (evidence_references), and Chain (chain): an explicit, attributable, and time-locatable statement. Accessible everywhere as the declarations: [Declaration, ...] array, activated with extensions: ["declarations"], the same bidirectional pattern already used for evidence_references.

Doctrinal definition:

> *A Declaration represents an explicit, attributable, and time-locatable statement, preserved by EVIDE without verifying its truthfulness, in order to enable its subsequent evidentiary reconstruction.*

Implemented schema - EVIDE Schema 2.1

The structure below is implemented in the EVIDE payload through extensions: ["declarations"] and the declarations array.

Reference structure:

```
Reference {
  reference_type                     // free text - "policy_document", "manifest",
                                      // "prompt", "target_system", ...
  declared_reference_relationship    // free text - "authorizes", "describes",
                                      // "restricts", ... (why it is cited)
  pointer
  hash: { algorithm, value }          // optional - algorithm as free text,
                                       // not restricted to SHA-256 (no EVIDE
                                       // intake_hash to compare against here)
  hash_scope
  hashed_by                           // required if hash is present
}
```

Declaration structure:

```
Declaration {
  declaration_type            // free text - see catalog, Section 4
  declared_value               // the declared content
  declarant                    // who made THIS declaration - not
                               // necessarily authority.dapi_number,
                               // which remains whoever signs the record
  authority_source: {
    declared_attribution_status,  // attributed | fragmented | implicit | unknown -
                                   // reuses the enum already used for
                                   // threshold_authority elsewhere in the schema
    references: [Reference, ...]  // what legitimizes the declarant's authority
  }

  subject_references: [Reference, ...]  // what substantiates the declared content
                                        // (documents, or the target system)
                                        // see Phase 6, "what scope did it cover")

  declared_relations: [                 // array - a Declaration can relate
                                         // to more than one previous Declaration
    {
      target_declaration_digest,        // points to the digest of another
                                         // Declaration, never to its evide_id
      declared_relationship,            // free text - "supersedes", "clarifies",
                                         // "revokes", "restricts", "expands", ...
      declared_reason                   // extended rationale
    }
  ]

  declared_at
```

```
  declared_description
}
// declaration_digest: computed by EVIDE server-side over the canonical form of
// the single Declaration, returned in the response - NEVER in the client payload.
// It is this digest, not an evide_id, that declared_relations points to.
```

Canonicalization: the canonical form is implemented as specified below, based on RFC 8785 (JSON Canonicalization Scheme):

```
declaration_digest = SHA-256("EVIDE_DECLARATION_V1:" || JCS(Declaration_Payload))
```

JCS deterministically orders object keys, normalizes number representation, and removes insignificant whitespace - there is no need to separately handle array ordering, which in JSON is already intrinsically significant and must never be reordered. The "EVIDE_DECLARATION_V1:" prefix provides domain separation: it prevents a Declaration's digest from semantically colliding with that of another EVIDE object type, and explicitly versions the canonicalization scheme itself.

A rule that JCS alone does not decide, and must be stated explicitly alongside the formula: an optional field that was never declared must be **omitted** from the canonical form, never serialized as null. Treating it as null would turn "this field was never declared" into a fictitious declared value - exactly the kind of inference that Decision 2 prohibits.

Recommended convention for `pointer` when `reference_type` indicates a system (not a document): use standard identifiers (e.g. urn:sys:supabase:prod-db, or equivalent ARN-style schemes) instead of free-form strings invented by each integrator. reference_type remains free text by design - this is not a schema restriction but an implementation guideline: without a shared convention, two systems referring to the same database would produce different pointers, making long-term cross-client forensic queries impossible.

Naming principle applied uniformly: every field that carries an interpretation or judgment made by the declarant (not a verifiable structural fact) takes the declared_ prefix - so that it does not sound like a classification confirmed by EVIDE when it is only a declared opinion. Applied not only to the most visibly interpretive fields, but also to ones that appeared neutral at first glance (attribution_status -> declared_attribution_status), a correction found during structured peer review after it had been missed both in the initial design and in an earlier review pass.

Reflection kept for the future, not to be acted on now: if Declaration is "an explicit, attributable, and time-locatable statement," then External Artifact ("the declaration of the existence of an external artifact") and Evidentiary Continuity ("the declaration of a relationship between records") could also be special cases of Declaration, rather than independent primitives - making it not the fourth primitive but the most fundamental of all. A genuine reflection, but a refactor today would compromise the discipline of this very design process. Noted, not acted upon.

4. Initial Catalog of declaration_type

Free text, not a closed enum - consistent with chain_type and every other categorizing field in EVIDE. Six documented types, not exhaustive:

declaration_type	What it declares	Example from the real case
environment_classification	In which environment the declarant states the agent was operating	"production" vs. "shadow/test" - the core of the RedHotCyber incident
privilege_scope	Which operations the declarant states fell within the agent's operational perimeter	"read-only" vs. "schema_migration"
declared_purpose	The purpose for which the declarant states the agent had been enabled to operate - distinct from privileges (what it can do vs. why it was declared permitted)	"resolving a schema issue" does not imply "running migrations against production"
tool_availability	Which tools/systems were granted	direct connection to Supabase, instead of a dedicated shadow environment
prohibited_operations	Declaration of authorized absence - a new pattern for EVIDE, the only negative rather than positive declaration	"DROP TABLE never permitted" - its absence does not imply "everything permitted" (Decision 2)
agent_configuration	Which model, version, or system prompt was in effect	distinct from execution_identity (who the agent is) - this declares under which precise configuration

Identity (owner, agent) does not have its own declaration_type: it reuses declarant, authority_source, and execution_identity, already present in the schema/MCP. No duplication.

5. Questions It Can Answer

Seven forensic questions, each verified against the schema - none required a field that did not already exist at the time it was posed:

Forensic question	Mechanism in the model
Who declared this?	declarant
When?	declared_at (declared by the client) compared against intake_timestamp_utc (an objective, server-side fact already existing at the record level - no new field)
Under what authority?	authority_source.declared_attribution_status + authority_source.references[]

Forensic question	Mechanism in the model
What scope/subject did it cover?	subject_references[], with reference_type extended to categories such as "target_system" beyond documents
Did a later Declaration explicitly linked to this one exist before that moment?	The presence of a Declaration whose declared_relations[].target_declaration_digest points to this one. Corrected after external review: the question was originally phrased as "was it still valid?", resolved by observing the absence of a subsequent link - an inference from silence that directly contradicts Decision 2. The presence of a link is demonstrable; its absence does not demonstrate persistence, effectiveness, or validity (it could mean that a revocation was never acquired, that it happened in another system, or that the client stopped submitting intake). The word "valid" was also removed: it is already a legal or operational judgment, not a fact EVIDE can attest to.
Was it modified or superseded? By whom?	Any Declaration whose declared_relations[].target_declaration_digest points to this one, together with its own declarant
With what continuity over time?	chain.parent_evide_id at the record level - reuses the already existing Evidentiary Continuity

Possible comparisons (Decision 14): *Declared* is the only level EVIDE touches - made independently retrievable and verifiable via declaration_digest. EVIDE does not independently determine the *observed* level, nor does it compute conformity between declared and observed - but logs and telemetry can still be preserved or referenced as independent evidence (Artifacts, evidence_references, Continuity): the statement that "observed is never inside EVIDE" was too absolute, and was corrected after external review. *Reconstructed* remains the work of the investigator who compares the declared level against any observational evidence - always outside EVIDE, never one of its outputs.

6. Cross-Sector Validation

The Claude/RedHotCyber case is a single domain (technical infrastructure). For the primitive to hold, it must answer the same question - *"what was the declared perimeter, and who declared it, before the agent acted?"* - in domains where no database or Prisma migration exists.

Sector	Declarable perimeter (concrete example)	What it would make reconstructible
Insurance & Underwriting	privilege_scope: automatic approval of auto claims under $5,000; subject_references -> target_system "auto-pd-claims"; prohibited_operations: total-loss determinations	If a supervisor raises the limit to $10,000, it is a new Declaration with declared_relations pointing to the first (declared_relationship: "expands") - the change to the perimeter is itself dated and attributed, not reconstructible from an internal changelog alone

Sector	Declarable perimeter (concrete example)	What it would make reconstructible
Banking & Financial Services	declared_purpose: "credit scoring on unsecured consumer loans"; privilege_scope: "read-only, no automatic rejection above regulatory threshold"	If the model is subsequently used for mortgage lending without a corresponding Declaration having been acquired by EVIDE, the record does not document that expansion. The absence remains a neutral evidentiary gap: it demonstrates neither authorization nor prohibition
HR & Recruitment - AI Act High-Risk	declared_purpose: "shortlist candidates for technical roles"; prohibited_operations: "final rejection without human review" - consistent with the AI Act's human-oversight obligation for high-risk HR systems	If the system is later configured to reject candidates without human review, this can be checked against a prohibited_operations Declaration acquired by EVIDE before the event, rather than only against an unanchored company policy
Healthcare & Clinical Decision Support	subject_references -> target_system "ER triage queue"; prohibited_operations: "autonomous therapeutic orders"	A change to agent_configuration (a new system prompt allowing direct medication orders) would be a new Declaration, traceable and comparable against the original perimeter - the same declared/observed distinction that Decision 2 always reserves for the investigator
Autonomous Patrol & Industrial Robotics	environment_classification: "perimeter only, non-public zones"; privilege_scope: "observation, no physical intervention"	A deployment in a public zone without a corresponding updated Declaration is exactly the case covered by Decision 2, point 6: absence does not imply authorization, it is a neutral evidentiary gap to be flagged as such
Biometric Identification	declared_purpose: "1:1 identity verification"; prohibited_operations: "1:N surveillance or tracking"	The distinction between the two uses - often the most delicate legal line in this sector - becomes a dated and attributed declaration, not a post-hoc interpretation of internal policy
Emergency Dispatch & Triage	declared_purpose: "dispatch queue prioritization"; prohibited_operations: "cancelling a response without human confirmation"	Same structure as the Healthcare case, applied to a domain where reconstruction time (who authorized what, when) is probably the single most critical factor
Critical Infrastructure	tool_availability: "telemetry read-only"; prohibited_operations: "sending control commands to switches"	Any escalation toward control commands could be checked against a perimeter Declaration acquired by EVIDE before the event - not against a policy that no one can date with certainty after the fact

Honest observation: no sector among those listed was found where the model stops holding or requires a different mechanism from the one already built for the original case. Every example uses only the six declaration_type values already cataloged - none required a new one. This is itself a signal

(not definitive proof: the stress test was performed on paper, not against real implementations in each sector) that the primitive generalizes correctly beyond the case that originated it.

7. Interoperability (Phase 7) - no new links to build

- **GLM:** no direct link - different layers (GLM is a public manifest, Declaration an intake mechanism).
- **External Artifacts:** references[]/subject_references[] already fully reuse the shape of evidence_references.
- **Evidence Stabilization Buffer:** no interaction - the ESB observes the stabilization of a closure, the Declaration declares a perimeter. Distinct domains.
- **Evidentiary Continuity:** chain links records over time, declared_relations links Declarations within the same record or across different records - two levels of the same principle, not two competing mechanisms.
- **Observer:** a judgment on a declared perimeter would be its own Declaration (declarant = the Observer), never an extension or modification of the original - preserving the neutrality of Decision 2.
- **FCC/DWC/FAC:** Declaration stays outside the server-computed evidentiary_profile - EVIDE does not judge, not even here.

7bis. Position within the EVIDE Ecosystem

Four protocols, four distinct questions - no overlap:

Protocol	Covers	Answers
FEDIS	Integrity, authenticity, chain of custody	Is the acquired data/record the same as originally acquired, by whom and with which tools, and is its custody traceable over time? Developed according to principles and methodologies consistent with ISO/IEC 27037 and with eIDAS signature and timestamping tools; designed to support authentication and integrity needs also relevant under the FRE 901/902 framework
DAPI	Identity	Who is the subject/authority signing or asserting this fact? - accountability
GLM	Governance	What are the general declared rules for this ecosystem/model? - public, high-level
ANCHOR	Perimeters	Within which specific declared boundary (technical, legal, clinical, ethical) was the agent enabled to operate at THAT moment? - point-in-time, not general

The distinction between GLM and ANCHOR is the one most worth keeping in mind: GLM declares the ecosystem's rules once, publicly; ANCHOR declares the perimeter of a single action, at a single moment, for a single agent. These are not two ways of saying the same thing at different levels of detail - they answer two questions that remain distinct even when both answers exist for the same system.

Note on the table: GLM, DAPI, and ANCHOR are native components of the EVIDE ecosystem, but they operate through different mechanisms: GLM through a machine-readable manifest, DAPI through certified identity and a verifiable reference, ANCHOR through Declarations included in the intake payload. FEDIS is a broader, pre-existing forensic certification framework, used across the entire certification service portfolio (web pages, chat, email - not only AI agent decisions), with its own separate professional-use regime. The table remains valid for distinguishing the questions each one answers, but FEDIS operates at a somewhat different level from the other three, not perfectly on par with them.

8. Implementation (Phase 8)

ANCHOR is implemented in Schema 2.1. The declarations field is subject to whitelisting and structural validation, travels through the existing intake endpoint, and is included in EAR records and archive views. Validation enforces the conditional requirement of hashed_by whenever hash is present (the same pattern already frozen in v2.1), and declaration_digest is computed server-side.

9. Implementation and End-to-End Validation

EVIDE ANCHOR has been implemented in Schema 2.1 and validated through five end-to-end scenarios executed via a real MCP client. The tests included standard intake, Evidence References, the Evidence Stabilization Buffer, ANCHOR, and a combination of the three extensions.

Validation confirmed that multiple Declarations can coexist within the same intake record and that ANCHOR does not alter FCC/DWC/FAC inferences. It also surfaced the **Atomic Declaration Rule**: each Declaration should represent a single attributable statement; environment, privileges, purpose, and tools must therefore be preserved as separate Declarations when they constitute distinct facts.

Status

The complete map (Phase 0 plus eight phases) has been finalized. Two substantive doctrinal corrections were found during review and remain visible in this document, not hidden: the Decision 3 overclaim, and the inference of validity from silence, which directly contradicted Decision 2. The canonicalization of declaration_digest (Section 3) is specified: RFC 8785 (JCS) with a domain-separation prefix, plus the explicit rule that absent fields must be omitted, never serialized as null. Protocol name decided: **EVIDE ANCHOR**, with Declaration as the underlying technical primitive - a

structural decision that frees the acronym from having to carry the entire doctrine on its own. No open points remain, whether doctrinal, technical, or related to naming.