

EVIDE Minimum Intake JSON

Evidentiary Intake Structure and Interoperability Model

Schema v2.1 · Public Specification v2.1 · July 2026

This public edition describes the EVIDE evidentiary intake structure and interoperability model. Operational access details are available to certified API partners only.

What EVIDE Is

EVIDE (External Evidentiary Deposit) is a structured evidentiary intake layer designed to anchor AI-assisted or human decisions to an independent, cryptographically verifiable record. Each intake transforms a closed decision into a portable evidentiary object that can be validated outside the originating system, without requiring operational access to the source system, its logs, or its infrastructure.

This independence is what allows the evidentiary object to be verified without reliance on the originating system.

"EVIDE does not standardise how a decision is made. It standardises the minimum evidentiary object that can survive outside the source system."

EVIDE does not:

- verify whether the declared authority was legitimate
- validate the correctness of the decision
- introduce admissibility logic
- replace governance frameworks
- operate inside the decision pipeline

Boundary Semantics

EVIDE operates at a specific and bounded layer. Understanding what it does and does not do prevents misuse and misinterpretation.

- EVIDE records closure state, not runtime execution
- EVIDE anchors the externally visible responsibility state at the moment a decision becomes final
- EVIDE preserves admissibility visibility without enforcing admissibility
- EVIDE is evidentiary, not supervisory
- EVIDE does not replay decisions - it anchors closure objects
- EVIDE does not verify declared oversight - it makes the declaration persistent, portable, and independently verifiable

Behavioral completion vs. Responsibility closure

A finalized decision without a bound authority represents behavioral completion only - the system produced an outcome, but no accountable actor is formally attached to it.

A finalized decision with a bound authority and declared human oversight represents responsibility closure - the outcome is not only produced, but explicitly attributable.

The Layer 2 → Layer 3 boundary requires responsibility closure, not behavioral completion.

Identity-Bound Access Model

EVIDE access provisioning requires DAPI identity verification. This requirement exists because EVIDE operates as an evidentiary responsibility layer. The intake infrastructure must be able to attribute API activity to a verifiable identity source, even when deposited payloads contain declared or non-verified authority states.

DAPI therefore functions as the identity-bound access layer of the EVIDE intake infrastructure, separating:

- infrastructure access attribution
- payload-declared authority
- evidentiary responsibility visibility

Access provisioning details are confidential and available to certified API partners only. Contact info@certifywebcontent.com to request access.

1. Schema Structure

The EVIDE intake payload is a JSON object submitted via POST. It is canonicalized and hashed deterministically before storage. The current schema version is "2.1".

REQUIRED FIELDS

- `evide_schema` - must be "2.1", the current and only value accepted by the API. Earlier values ("1.8", "1.9", "2.0") are historical and no longer accepted.
- `source_system` - name of the originating system
- `source_reference` - unique decision identifier in the source system
- `source_timestamp_utc` - ISO 8601 UTC timestamp of decision production
- `decision.type` - decision category (e.g. `candidate_evaluation`, `risk_classification`)
- `decision.status` - final state, use "finalized"
- `decision.closure_timestamp_utc` - UTC timestamp of closure
- `decision.summary` - natural-language description of the decision
- `authority.id` - authority identifier in the source system
- `authority.role` - role of the authority
- `authority.dapi_number` - must match the DAPI identity bound to the API key used for the request. Cross-checked server-side against the key's owning identity at intake time.
- `intervention.classification_context` - upstream governance structure visibility (v1.7). The `intervention` object itself is required.

NOTABLE OPTIONAL FIELDS

- `authority.verification` - DAPI reference code linking the authority to a verified identity. Without it, `evidentiary_profile.identity` returns "declared" instead of "claimed".
- `intervention.type` - override / approval / rejection / escalation
- `content_hash` - lets the client declare, before submission, the hash it expects EVIDE to compute over the canonicalized payload. EVIDE does not compare this value against its own computation; the client can independently compare its declared `content_hash.value` with the `intake_hash` returned by EVIDE. Only `SHA-256` is accepted as the algorithm, since that is the only algorithm EVIDE's own `intake_hash` ever uses.
- `fedis_requested` - triggers FEDIS forensic certification workflow
- `chain.parent_evide_id` - links to a previous intake, forming a verifiable evidence chain. Must reference an existing record in the same evidentiary domain, in an accepted status — validation is strict, with no silent fallback to a new chain
- `chain.chain_type` - free text describing the continuation (e.g. `litigation_hold`, `internal_investigation`). No fixed enum
- `chain.matter_reference` - optional; if declared on both sides it must match, and is inherited automatically from the parent when omitted
- `handoff` - boundary state declaration for L3 intake. Required. `boundary_readiness` is a structured object with `status`, `readiness_gate`, `visibility_surface`, and `unresolved_signals`. See Section 3.

- `evidence_references` - array of declared external artifact references (screenshots, video, sensor logs, documents), anchored without EVIDE storing the underlying file. Optional integrity hash. (v2.1)
- `declarations` - array of declared operational perimeter statements (environment, privileges, purpose, tools, prohibited operations, agent configuration) an agent was authorized within, before it acted (EVIDE ANCHOR). EVIDE preserves the declaration only, never its correctness. (v2.1)

TOP-LEVEL FIELD WHITELIST

Only the fields listed above are accepted at the top level of the payload. A field the server does not recognize is refused with `422 invalid_field` rather than silently ignored — an unrecognized field would otherwise be canonicalized into the intake hash without ever having been validated. Three fields described in earlier public materials — `human_oversight`, `object_class`, and `created_at_utc` — were never read or validated by the server and are no longer accepted. They remain in the Schema Evolution history below as a record of when each was introduced. Validation runs in three tiers: the top-level whitelist above (tier 1); structural checks — required fields, types, enums, and the rules for `content_hash` and the ESB buffer object (tier 2); and semantic checks that cross-reference something outside the payload itself — `authority.dapi_number` against the API key's owning identity, `chain.parent_evide_id` against the referenced record, and the ESB minimum observation window (tier 3).

MINIMAL PAYLOAD EXAMPLE

```
"evid schema": "2.1",
"source_system": "AquariuOS",
"source_reference": "CDR-2026-00421",
"source_timestamp_utc": "2026-04-07T09:15:00Z",
"decision": {
  "type": "candidate_evaluation",
  "status": "finalized",
  "closure_timestamp_utc": "2026-04-07T09:15:00Z",
  "summary": "AI recommendation overridden by human authority"
},
"authority": {
  "id": "user_87421",
  "role": "HR Reviewer",
  "dapi_number": "DAPI-0000000001"
},
"intervention": {
  "classification_context": {
    "threshold_status": "not_defined"
  }
},
"handoff": {
  "boundary_readiness": {
    "status": "candidate",
    "readiness_gate": null,
    "visibility_surface": null,
    "unresolved_signals": []
  },
  "reconstruction_independence": "declared",
  "submission_status": "not_submitted",
  "acceptance_status": "not_claimed"
}
```

Full schema reference: app.certifywebcontent.com/json

2. classification_context (v1.7 + v1.8)

The `classification_context` object exposes whether the classification act was supported by an upstream taxonomy and threshold structure at the moment of recording. It does not validate the threshold - it makes the presence or absence of such a structure externally visible.

THRESHOLD_STATUS VALUES

- `met` - threshold was defined and the decision satisfied it
- `not_met` - threshold was defined but not satisfied
- `unknown` - threshold may exist but state is not verifiable at deposit time
- `not_defined` - no threshold was formally defined for this classification

Governance gap signal: `threshold_status`: "not_defined" does not indicate an operational error. It signals that at the moment of decision, no formally defined governance structure existed - a governance gap made externally visible, not a system failure.

THRESHOLD_AUTHORITY (V1.8)

Exposes whether the threshold had a structurally attributable source of authority at closure. `attribution_status` is required when `threshold_authority` is present.

- `attributed` - single identifiable authority defined at closure
- `fragmented` - threshold defined across multiple sources, no single attributable authority
- `implicit` - threshold present in practice but not formally defined or attributable
- `unknown` - threshold authority not verifiable at the moment of closure

3. handoff (v2.0)

The `handoff` object defines the boundary state of a closure object before EVIDE accepts it for external anchoring. It is required. `boundary_readiness` is a structured object with four canonical states (promoted from a plain string in v2.0).

- `boundary_readiness` - in v2.0 a structured object with: `status` (`candidate` | `verified` | `verified_partial` | `unverifiable`), `readiness_gate.identifier`, `readiness_gate.scope_reference`, `visibility_surface`, `unresolved_signals`. In v1.9 a string: `candidate` or `verified`.
- `reconstruction_independence` - must be declared for EVIDE intake.
- `submission_status` - must be `not_submitted` when sent by the upstream system.
- `acceptance_status` - must be `not_claimed` when sent by the upstream system.

Boundary rule: no downstream step should require reconstruction of intent from audit logs, human memory, or system-specific context. The object must be self-contained and independently verifiable before submission.

After a successful intake, EVIDE returns both `evidentiary_profile` (dimensional read of the object) and a `handoff` confirmation object with `submission_status`: "submitted" and `acceptance_status`: "accepted" — confirming the upstream object crossed the EVIDE intake boundary. The original submitted payload is stored and hashed without modification - the evidentiary fingerprint always reflects what the upstream system declared, not what EVIDE computed after intake.

boundary_readiness and runtime conditions — architectural note

EVIDE anchors closure states, not runtime states. It does not monitor conditions after anchoring and does not introduce runtime governance.

The distinction between "candidate" and "verified" carries a precise architectural meaning: it declares whether the runtime conditions that grounded the closure state were independently confirmed as stable at the moment of boundary crossing.

A system cannot evaluate its own boundary readiness and declare it “verified”. That evaluation must come from a gate that is independent of the system that produced the decision. If no such independent confirmation exists, “candidate” is the correct and complete value — not a fallback.

This distinction is what prevents EVIDE from anchoring a closure state that was already in runtime degradation at the moment of submission.

Note: “verified” does not guarantee correctness of the decision. It confirms only that boundary readiness conditions were independently assessed as stable at the moment of crossing.

EVIDENCE STABILIZATION BUFFER (ESB) - V2.1

For decisions that require observation over time before the evidentiary boundary is crossed - a condition that may evolve before closure, rather than being final at submission - EVIDE offers an additional intake profile: the Epistemic Stabilization Buffer. It is part of EVIDE Schema v2.1 and does not require a separate payload schema. Operational endpoint details for ESB are available to certified API partners only - see the Identity-Bound Access Model section above.

4. Canonicalization and Hashing

Before computing the SHA-256 hash, the system canonicalizes the JSON: keys are sorted alphabetically at all levels and the result is minified. This guarantees that two semantically identical payloads always produce the same hash regardless of key submission order.

ALGORITHM

- Remove the content_hash node if present
- Recursively sort all keys alphabetically at every nesting level
- Serialize to minified JSON (UTF-8, no escaped slashes)
- Compute SHA-256 of the resulting string

Key ordering inside classification_context: taxonomy_reference → threshold_authority → threshold_reference → threshold_status. This order must be respected when computing content_hash on the source side.

Integrity guarantee: the intake hash returned in the API response is the authoritative fingerprint. The source-declared content_hash is optional but extends the integrity chain to cover the pre-intake transit as well.

The API response (v2.0) includes `evidentiary_profile` — a server-computed object with `profile_version`: "1.1". It is not part of the submitted payload and is not included in the canonical hash.

EVIDE does not reduce an evidentiary object to a single validity state. It exposes the known, partial, degraded, or unverifiable dimensions of that object at the moment of deposit.

Dimension	Question	Source field	Possible states
identity	Who is linked to the record?	<code>authority.verification_presence</code>	claimed / declared
authority	Who assumes decision responsibility?	<code>authority.id + authority.role</code>	declared / null
classification	Is the classification stable?	<code>intervention.classification_status</code>	stable / provisional / contested
threshold	Was a decision threshold defined?	<code>classification_context.threshold_status</code>	met / not_met / unknown / not_defined
threshold_authority	Who owns the threshold?	<code>threshold_authority.attribution_status</code>	attributed / fragmented / implicit / unknown
boundary_readiness	Was the object stable at crossing?	<code>handoff.boundary_readiness.status</code>	candidate / verified / verified_partial / unverifiable
runtime_visibility	Was the runtime condition observable?	<code>handoff.boundary_readiness.visibility_surface</code>	confirmed / partial / unverifiable / null
trace_reference	Can the upstream trace be located?	<code>intervention.trace.access</code>	available / restricted / unavailable
continuity	Does the declared observational visibility structurally support the classification?	<code>inferred: classification x runtime_visibility (Forensic Cross-Check)</code>	{mode: inferred, state: stable / degraded / broken / unknown}
dwc	Decision Wave Compression : is oversight throughput exceeded?	<code>inferred (DWC)</code>	{mode: inferred, state: not_detected detected critical unknown}
fac	Formal Accountability Collapse: is accountability attributable?	<code>inferred (FAC)</code>	{mode: inferred, state: not_detected detected critical unknown}

5. Schema Evolution

EVIDE follows a strict backward-compatible evolution model. Each version adds optional fields or response attributes without breaking existing integrations.

Version	Addition
1.0	Base schema - decision, authority, intervention, human_oversight, chain
1.1	created_at_utc, object_class, content_hash
1.2	intervention.rationale_type
1.3	intervention.taxonomy_version - enables classification replay in audit
1.4	fedis_requested - triggers FEDIS forensic certification workflow
1.5	authority_verification_status (response) - claimed vs declared. Consolidated into evidentiary_profile.identity in v2.0.
1.6	intervention.classification_status - stable / provisional / contested
1.7	intervention.classification_context - upstream governance structure visibility
1.8	classification_context.threshold_authority - threshold ownership attributability
1.9	handoff - boundary-ready schema, universal handoff contract for L3 intake. Required for evid_schema 1.9. boundary_readiness as string: candidate verified.
2.0	handoff.boundary_readiness promoted from string to structured object. Four canonical states: candidate, verified, verified_partial, unverifiable. Adds readiness_gate (identifier + scope_reference), visibility_surface, unresolved_signals. intervention.classification_context required for full evidentiary_profile dimensional computation. decision.status must be "finalized". Response includes evidentiary_profile (server-computed, profile_version 1.1) with eleven evidentiary dimensions. Dimension 9 is inferred via Forensic Cross-Check (classification x runtime_visibility) and expresses the structural coherence of a single closure — exposed under the historical API field name continuity, unrelated to Evidentiary Continuity across a chain.
2.1	extensions and evidence_references arrays - declaring external artifacts (screenshots, video, sensor logs, documents) anchored without EVIDE storing the underlying file. Any integrity hash is declared by the submitter and never independently verified by EVIDE. Adds chain.matter_reference and strict server-side validation of chain continuations. The extensions array is opt-in in both directions: an optional block present in the payload but not listed there is rejected, and an extension listed there with no corresponding content is rejected as well. The additions are non-breaking at the payload structure level. Also adds the declarations array (EVIDE ANCHOR) for declaring the operational perimeter an agent was authorized within, before it acted - same opt-in extensions mechanism, same non-verification principle. "2.1" is the current and only evid_schema value accepted by the API.

6. Interoperability

EVIDE is designed to receive structured closure objects from any system - AI pipelines, workflow engines, governance platforms, runtime agent systems - without requiring integration inside the decision pipeline.

The minimal boundary requirements for the source system are:

- an explicit closure state (not only behavioral completion)
- an identified authority bound to that closure
- an explicit responsibility declaration
- a stable reference to the upstream trace
- a coherent classification context (even if initially internal)

SOURCE IDENTITY FIELD MAPPING

- `evidence_id` → `source_reference` (primary object identity)
- `event_id` → `intervention.trace.reference` (runtime event identity)
- `trace_reference` → `intervention.trace.reference` (upstream trace continuity)

Reference implementation: RANKIGI × EVIDE Interface Mapping (v0.4.1) - app.certifywebcontent.com/docs/rankigi-evide-interface/

7. FEDIS Compatibility

FEDIS (Forensic Evidence Declaration & Integrity Statement) transforms an EVIDE intake record into a legally presentable forensic artifact. It is not a default system artifact but a formal evidentiary output generated through a certification process, triggered by setting `fedis_requested`: true in the intake payload.

A FEDIS-ready EVIDE record must include:

- cryptographically fixed closure state (SHA-256 intake hash)
- independent UTC intake timestamp
- declared authority bound to the closure
- classification context sufficient for external interpretation
- stable upstream trace reference

FEDIS extends the EVIDE record with a qualified eIDAS digital signature, RFC 3161 timestamping, SHA-256 and SHA-512 fingerprints, formal chain of custody declaration, and sworn statement - making the same record usable in audit, regulatory, and judicial contexts without reprocessing.

Architecture in three layers

- Source system - produces the decision and surfaces the closure object
- EVIDE - anchors the closure object externally with integrity and timestamp
- FEDIS - transforms the anchored record into a legally presentable artifact

One closure. Three layers. One independent record.

*For operational access, contact
Dr. Emanuel Celano
Founder & Chief Architect at EVIDE
info@certifywebcontent.com*