Aleten Labs

RESEARCH PAPER 3 OF A SERIES
WP-2026-03 · V1.0
AUGUST 2026 · CC BY 4.0 · SCHEMA OPENLY LICENSED

The derivation record: an open schema for per-response traceability in conversational AI

Eighteen fields written on the response path, a client-carried state envelope under a keyed integrity tag, and the claim that record-keeping, not explanation, is where conversational systems can be held to account.

B. Harvell · Aleten Labs · research@aleten.com

18	6	10 / 10	20 B	~146 B	0
TOP-LEVEL FIELDS, LOG V1.3.0	FLAG SUMMANDS AUDITED PER TURN	STATELESS REPLAY, REPLY + LOG + ENVELOPE	KEYED MAC TAG ON THE ENVELOPE	ENVELOPE GROWTH PER TURN, CAP-BOUNDED	FIELDS POPULATED BY INFERENCE
counted	counted	measured	measured	measured	specified

ABSTRACT

When a conversational system is asked to account for an output, it can offer an explanation or a record. Explanations generated by statistical systems are post-hoc narratives with no privileged access to the computation that produced the output; records, if the architecture supports them, are the computation. We publish the derivation record of the Ackren engine as an open, versioned schema: eighteen fields written on the response path of every turn, capturing the typed flags raised, the assumptions made and their revision status, the reference bindings and the relation that licensed each, the selection branch taken and the branches not taken, the realisation mode of every output span, the round-trip verification outcome, and the content-addressed build identifier that makes the whole record independently re-derivable. We describe the companion state envelope, which carries the entire conversation state between turns under a keyed integrity tag so that a stateless server can prove its turns while holding nothing, with tampering refused as a typed input error and growth bounded by enumerated caps. We map the record field by field onto the EU AI Act's record-keeping obligations and onto the traceability assumptions of IEC 62304 and DO-178C, and we contribute draft normative text distinguishing record-keeping that any vendor can satisfy from record-keeping that requires derivability. The schema is published under a permissive licence with an invitation to implement it; the fields no statistical architecture can populate are identified, and so are the things no record can show, because a schema whose limits are undocumented is a marketing artefact, not an audit one.

STATUS OF THIS DOCUMENT

Companion to WP-2026-01 v1.5 (the engine specification) and WP-2026-02 v1.1 (the failure architecture). All measured figures cite the engine report for development build df353a9add9aa347; example records are drawn verbatim from that build's frozen probe set and are mechanism-safe under its register quarantine.

01 RECORDS, NOT EXPLANATIONS

why the schema is the product

Ask a neural conversational system why it said what it said and the answer is another sample from the same distribution that produced the output (Bender and Koller, 2020; Vaswani et al., 2017). The explanation has no privileged access to the generation; it is testimony, not evidence. Regulation has responded by demanding records instead: the EU AI Act's Article 12 requires high-risk systems to keep logs sufficient to trace their functioning (European Union, 2024), and the safety-critical software standards assume, as their foundation, that any output can be traced to the requirement and mechanism that produced it (IEC, 2006; RTCA, 2011; ISO, 2018; CENELEC, 2011). A statistical system meets these obligations by approximation: it can log its inputs, outputs, timestamps and model version, but the middle, why this output and not another, is not recorded because it is not discrete. A deterministic engine can log the middle. This paper publishes what that looks like when it is designed rather than retrofitted.

Two design commitments distinguish the derivation record from an event log. First, it is written **on the response path**: the record is assembled as the turn executes, by the same code paths, and a turn that cannot write its record does not ship its reply. There is no debug flag, because a record that can be turned off is a record that was off when it mattered. Second, every field is **populated by mechanism, not inference**: the record contains what happened (this flag was raised at this stage with this payload; this selection branch fired because this guard held) and never contains estimates, confidences, or reconstructions. The record of a turn, together with the versioned artefact the build identifier names, is sufficient to re-derive the turn, by machine or by hand, which is the property the whole series rests on (Aleten Labs, 2026a).

RELATED INSTRUMENTS

The W3C PROV data model standardised the vocabulary of provenance, entities, activities and agents, for describing how artefacts came to be (Moreau and Missier, 2013); the derivation record can be read as a domain-specific PROV profile in which the activity is a conversational turn and the generating entity is a content-addressed artefact. Secure audit logging established the adversarial requirements, integrity of records against tampering by the very party that holds them (Schneier and Kelsey, 1999), which reappears here as the envelope's keyed integrity tag, since the party holding the conversation state between turns is the untrusted client. What neither tradition supplies is the linguistic interior: provenance and audit frameworks record *that* a computation happened and *who* held it, not which reading of an ambiguous pronoun was taken and under which licence. The derivation record's contribution is that interior, typed.

02 THE RECORD, FIELD BY FIELD

log v1.3.0, eighteen fields

FIELD	CONTENTS AND PURPOSE
`turn_id · session_id`	position of the turn; the session identifier travels in the envelope, so the record and the state name each other
`build_id`	content-addressed identifier of the artefact and engine that executed the turn; the determinism guarantee is indexed by it, and re-derivation starts from it
`log_version`	schema version of this record, semver; the schema changes by published changelog, never silently
`input`	byte length, content hash, and quoted spans of the user turn; the raw text itself is not in the record's audit tier (§5)
`flags[]`	every typed flag raised, with kind, summand, raising stage and evidence payload; the failure architecture of WP-2026-02, serialised
`assumptions[]`	every licensed assumption in force, with its licensing rule, evidence tier, and revision status (standing, eased, flipped, confirmed); the machine may lean, and this is where the lean is disclosed
`ignored[]`	what was dropped or overridden: expired QUD items, preference-order resolutions; the record of what the turn chose not to address
`binds[]`	every reference binding, naming the referent and the relation that licensed it; for bridging, the qualia role `(Pustejovsky, 1995)`. The field no statistical architecture can populate, because no statistical architecture holds a relation it could name
`intent_source`	which selection branch produced the response intent: flag, obligation, qud, initiative, continuation, or fallback; one word that says why this reply exists
`selected_intent · frame_id`	the intent chosen and the frame that constrained its realisation
`realisation`	per-span provenance of the output: each span's mode (grammar-searched, closed-class template, or quoted), the templated-token fraction, and the class-backoff fraction; the reply's own anatomy
`roundtrip`	outcome of re-parsing the output through the comprehension pipeline: pass, regenerations used, recovered intent; the per-response adherence proof, with its scope stated in §5
`stage_impl`	the named implementation of every stage that ran (e.g. `s6: priority_stack`); mechanism provenance, so a record from last year names the machinery of last year
`errata[]`	corrections attached to this record after the fact; records are never edited, they accrete errata, mirroring the development log's strike-not-delete rule
`audit`	the privacy-safe aggregate tier: flag counts by summand, assumption count, quoted-span count, token count, round-trip outcome; the tier that can be exported and published (§5)
`state_snapshot_ref`	reference to the post-turn state (or null); the hook on which cross-session persistence hangs without changing the record

The record is deliberately flat where it can be and typed where it must be. Nothing in it is free text except quoted user material, which is confined to the quoted-span mechanism precisely so that unknown input can appear in a reply without ever entering the parse, the context, or the generation vocabulary.

2.1 · THE THREE FIELDS THAT CARRY THE ARGUMENT

binds[]. When the engine resolves "the author" against a book mentioned two turns earlier, the bind entry names the referent, the relation (the agentive qualia role of *book*), and the lexical entry that supplied it. This is a statement about record-keeping, not about quality: a transformer may resolve the same reference more often, but it cannot produce this field, because there is no discrete relation in its computation to name. Any regulation, standard or contract that requires reference resolution to be *accountable*, rather than merely accurate, is requiring this field or its equivalent, and we publish it so that the equivalence class has a concrete member.

intent_source. One enumerated value that answers the question auditors actually ask: why did the system say this? Because comprehension failed and repair fired (flag); because a social obligation was owed (obligation); because a question was open (qud); because the domain plan proposed it (initiative); because the topic continued (continuation); or because nothing else applied (fallback). Six values, mutually exclusive, provable from the state, and sufficient to sort any transcript's turns into the ones that answered, the ones that asked, and the ones that honestly declined. The frozen conversation of the current build sorts, for example, into obligation, continuation and qud turns, with its one unanswerable question visibly routed through deferral `[measured]`.

assumptions[]. The engine is permitted to lean on licensed defaults (a typical pig has four legs) provided the lean is marked in the surface form, entered here with its licensing rule, and revised by a total policy when the user builds on, challenges or contradicts it. The field makes the epistemic state of the reply inspectable: a reply with an empty assumptions array asserted nothing it could not derive; a reply with entries says exactly where it leaned. User-taught facts appear with their provenance and session scope, and nothing the user teaches enters the shared knowledge base from here; the record shows the fact being queued for human verification instead, which is the honest form of "I'll remember that."

2.2 · BRANCHES NOT TAKEN

The rendered walkthrough of any turn includes, for the selection stage, the branches that did not fire and the guard that failed for each (obligation queue empty; QUD stack empty; no domain pack loaded). This is recoverable from the record because σ is a strict priority stack: naming the branch taken names everything above it as unavailable and everything below it as outranked `(Aleten Labs, 2026a; Aleten Labs, 2026b)`. An explanation of a decision that cannot enumerate the alternatives is a description; the record's enumeration is what makes it an account.

03 THE STATE ENVELOPE

a stateless server that can prove its turns

Between turns, the engine's server holds nothing. The entire conversation state, the QUD stack, obligation queue, referent list, belief ledger and assumption statuses, is serialised into an envelope carried by the client and returned with the next turn. Statelessness is usually an operational convenience; here it is an audit property: a turn is a pure function of (input, envelope, artefact), so the record plus the envelope plus the build identifier reproduce the turn exactly, on any machine, later, which is what independent re-derivation means in practice. The current build replays a hash-pinned ten-turn conversation this way with reply, record and envelope all byte-identical `[measured]`.

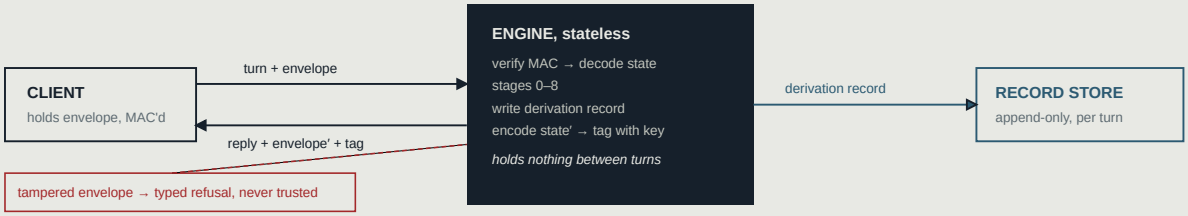

Figure 1 · The turn cycle. The client carries the state; the engine verifies, executes, records, re-tags. A conversation is a chain of pure function applications, each link independently re-derivable from its record. · `Basis: specified; replay and refusal both measured on the development build.`

3.1 · INTEGRITY: THE CLIENT IS THE ADVERSARY MODEL

A client-carried belief ledger is a client-editable memory unless it is protected, and an unkeyed checksum only catches accidents: a tamperer edits the ledger and re-computes the tag. The envelope therefore carries a 20-byte keyed MAC (keyed blake2b over the canonical body), with the key supplied as deployment configuration, never derived from engine state, because the engine deliberately has no ambient secrets to mint one from. The property is tested by mounting the actual attack: a ledger entry is forged, re-tagged with the unkeyed hash, and submitted; the live endpoint refuses it with a typed state-invalid error that routes to repair like any other input flag `[measured]`. Key rotation is scheduled work: a key identifier byte in the envelope, with rotation surfacing as the same typed error and a fresh-session repair, never a crash.

3.2 · GROWTH: THE BOUND IS ENUMERATED

The envelope measures 207 bytes after one turn and grows by a mean of ~146 bytes per turn over the ten-turn probe, and the per-turn series plateaus, with one turn not growing at all and the farewell shrinking the envelope as discharged obligations drop, because every carried structure is capped (QUD 8, obligations 8, referents 16). The growth of a client-carried state is exactly the kind of limit that is cheap to enumerate now and expensive to discover at turn two hundred, so it is enumerated now, with the caveat that absolute sizes shift by the length of the session identifier and the tag width is fixed whether keyed or not `[measured]`.

04 THE REGULATORY MAPPING

and a piece of draft normative text

Article 12 of the AI Act requires that high-risk systems technically allow the automatic recording of events over their lifetime, sufficient to identify situations that may present risks and to facilitate post-market monitoring (European Union, 2024). The mapping below is field-level rather than rhetorical; where an obligation is genuinely not addressed by the record, the cell says so.

OBLIGATION, PARAPHRASED	RECORD SURFACE
Events recorded automatically over the system's lifetime	the record is written on the response path of every turn; there is no configuration in which it is not
Traceability of the system's functioning	`flags[]` · `intent_source` · `binds[]` · `stage_impl` · `roundtrip`: not only what the system did but which mechanism did it, re-derivably
Identification of situations presenting risk	`flags[]` and `assumptions[]` are precisely the turn's self-identified risk surface: what was not understood, and what was leaned on
Reference period and dating	`turn_id` · `session_id` · `build_id` with append-only storage; retention is a deployment policy, stated honestly as out of the schema's scope
Version identification over updates	`build_id` and `log_version`; the record of a turn permanently names the artefact that produced it
Human oversight support (Art. 14 adjacency)	the rendered walkthrough: any turn expands to its stages, branches not taken included, readable without engine access
Transparency to users (Art. 50)	**not addressed by this schema**: disclosure that one is conversing with a machine is an interface obligation, and determinism buys nothing there

4.1 · TWO SENTENCES THAT LOOK ALIKE AND ARE NOT

Standards text is where the competitive landscape of transparency will actually be decided, so we contribute language rather than positions. Consider: *"providers shall document how outputs are generated."* Every lab with a technical writer satisfies this. Now: ***"providers shall record, for each output, the specific rule or artefact that produced it, such that the output can be re-derived from the record by an independent party."*** The second sentence is satisfiable by deterministic systems and, at present, by nothing else; it is also, we argue, what Article 12's traceability language is reaching for when read seriously. We offer the second sentence, and the schema that demonstrates its satisfiability, as a contribution to that discussion; a field-level public schema is worth more to a standards body than a position paper, because it is the difference between arguing a requirement is satisfiable and handing over the existence proof.

4.2 · WHY THE SCHEMA IS OPEN

The schema is published under a permissive licence, at a stable versioned URL, with a changelog and an explicit invitation to other vendors to implement it. If adopted, it becomes a standards contribution; if not, it is a file format with good documentation, and it cost nothing that was being kept. The asymmetry worth noticing: a vendor whose architecture can populate every field has no reason not to publish their schema, so the licence is also a question posed to the rest of the industry, politely.

05 WHAT THE RECORD CANNOT SHOW

and the privacy tier

Four limits, stated as claims about the schema rather than footnotes. The round-trip field proves production-comprehension agreement within the grammar, not that a human recovers the intent; a record can be complete, well-formed and pass, on a reply a person would misread (Aleten Labs, 2026a). The record shows what was authored and consulted, never what is true: a wrong knowledge base yields impeccable records of wrong answers, which is why the human verification programme is a safety control and the record carries provenance rather than truth values. Two engine failure modes are invisible to the record by design honesty: a future index false accept would be recorded as a verified lookup (bounded by fingerprint width), and long-range incoherence is not detectable by anything the record could carry (Aleten Labs, 2026b). And the record is per-turn: cross-turn properties live in the envelope chain, so an audit of a conversation requires the chain, not a single record, which is why the envelope is integrity-protected at the same standard as the record is versioned.

5.1 · THE AUDIT TIER

The full record contains material that identifies content (quoted spans, content hashes, referent structure), and one field class is deliberately excluded from exportable aggregates: raw index slot values are invertible back to vocabulary items, so the exportable `audit` object carries only per-turn aggregates, flag counts by summand, assumption count, quoted-span count, token count, round-trip outcome. Published logs and the development log's public metrics draw from this tier only. The schema thus has two audiences by construction: the deployment's own auditors, who hold the full records under the deployment's data-protection regime, and the public, who see aggregates that cannot be inverted into user content.

06 A WORKED RECORD

the refusal turn, abridged


```
{ "turn_id": 0, "session_id": "walkthrough", "build_id": "df353a9add9aa347", "log_version": "1.3.0",
  "input":  { "byte_length": 17, "content_hash": "b2s:7a0bfc347679c506",
              "quoted_spans": [ { "char_length": 7, "suppressed": "none" } ] },
  "flags":  [ { "kind": "ocv_oov", "summand": "ocv", "stage": "s1",
                "payload": { "quoted_char_length": 7, "suppressed": "none" } } ],
  "assumptions": [],  "binds": [],  "ignored": [],  "errata": [],
  "intent_source": "flag",  "selected_intent": "ack:intent/repair_ocv_oov",  "frame_id": "ack:cc/repair_ocv_oov",
  "realisation": { "spans": [ { "start": 0, "end": 5, "realisation_mode": "closed_class_template" },
                              { "start": 5, "end": 6, "realisation_mode": "quoted" },
                              { "start": 6, "end": 7, "realisation_mode": "closed_class_template" } ],
                   "templated_token_fraction": { "num": 6, "den": 6 } },
  "roundtrip": { "outcome": "pass", "regenerations": 0, "recovered_intent": "ack:intent/repair_ocv_oov" },
  "stage_impl": { "s0": "ingest_v1", "s1": "fullform_affix_licence", "s6": "priority_stack",
                  "s7": "closed_class", "s8": "closed_class_inventory", "...": "..." },
  "audit": { "flag_counts_by_summand": { "ocv": 1, "input": 0, "parse": 0, "ref": 0, "act": 0, "select": 0 },
             "assumption_count": 0, "quoted_span_count": 1, "token_count": 6, "roundtrip_outcome": "pass" } }
```


Reading it: an unknown word raised the OOV flag at stage 1 with its evidence; the flag forced the repair branch; the reply was realised from a verified closed-class frame with the unknown word confined to a quoted span; the output re-parsed to the intent it was generated from; and the exportable audit object summarises all of it without carrying the user's text. The reply this record accounts for is seven words long. That asymmetry, a paragraph of accounting for a sentence of output, is not overhead; it is the product, and it costs a fraction of a millisecond [measured].

07 CONCLUSION

the account, not the apology

The derivation record reframes conversational accountability from a generation problem, produce a convincing explanation, to an engineering one, write down what happened, in types, as it happens, under a hash that lets anyone check. Eighteen fields suffice for a system whose comprehension failures are typed, whose assumptions are licensed and ledgered, whose reference bindings name their licences, and whose outputs prove their own intent by round trip. A client-carried, integrity-tagged envelope extends the account across turns while the server holds nothing. The schema is open because its adoption costs its publisher nothing and poses a fair question to every other architecture; its limits are printed because an audit instrument that hides its own blind spots is not one. Subsequent papers treat the failure accounting the flags serialise (WP-2026-04) and the artefact separation the build identifier names (WP-2026-06).

A REFERENCES

BU Harvard, alphabetical

Aleten Labs (2026a) *Ackren: a formal specification of a deterministic conversational engine with per-response derivability*. Working paper WP-2026-01, v1.2. Aleten Labs.

Aleten Labs (2026b) *Honest before capable: typed failure and repair-first selection in dialogue architecture*. Working paper WP-2026-02. Aleten Labs.

Bender, E.M. and Koller, A. (2020) 'Climbing towards NLU: on meaning, form, and understanding in the age of data', in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. ACL, pp. 5185–5198.

CENELEC (2011) *EN 50128: Railway applications – software for railway control and protection systems*. Brussels: CENELEC.

European Union (2024) *Regulation (EU) 2024/1689 of the European Parliament and of the Council (Artificial Intelligence Act)*. Official Journal of the European Union, L series, 12 July.

Ginzburg, J. (2012) *The interactive stance: meaning for conversation*. Oxford: Oxford University Press.

IEC (2006) *IEC 62304: Medical device software – software life cycle processes*. Geneva: International Electrotechnical Commission.

ISO (2018) *ISO 26262: Road vehicles – functional safety*. Geneva: International Organization for Standardization.

Moreau, L. and Missier, P. (eds) (2013) *PROV-DM: the PROV data model*. W3C Recommendation, 30 April. W3C.

Pustejovsky, J. (1995) *The generative lexicon*. Cambridge, MA: MIT Press.

RTCA (2011) *DO-178C: Software considerations in airborne systems and equipment certification*. Washington, DC: RTCA.

Schneier, B. and Kelsey, J. (1999) 'Secure audit logs to support computer forensics', *ACM Transactions on Information and System Security*, 2(2), pp. 159–176.

Traum, D.R. and Larsson, S. (2003) 'The information state approach to dialogue management', in van Kuppevelt, J. and Smith, R.W. (eds) *Current and new directions in discourse and dialogue*. Dordrecht: Kluwer, pp. 325–353.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017) 'Attention is all you need', in *Advances in Neural Information Processing Systems 30*.

ALETEN · *ἈΛΉΘΕΙΑ* · UNCONCEALMENT ACKREN · DETERMINISTIC CONVERSATIONAL ENGINE

CORRECT, AND YOU CAN SEE WHY

WP-2026-03 · V1.0 · LOG V1.3.0 · ENVELOPE V1.1.0 · BUILD DF353A9ADD9AA347