EVIDE Technical Scope Note 02

NVIDIA OpenShell: What Remains Verifiable When Audit Records Are No Longer Available?

Log Retention, Declared Gaps and Independent Evidentiary Examination

Version 1.0 · 14 November 2026 · Emanuel Celano, EVIDE (External Evidentiary Deposit)

Summary

- OpenShell enforces policy on AI agents and records what it observes. Its documentation states openly that durable retention is the operator's task and that no local source guarantees a complete audit history.
- The point where a record leaves OpenShell for durable storage is therefore a custody handoff, and it currently takes place entirely inside the operator's control.
- EVIDE answers a narrower question than OpenShell does: who fixed, at what moment and with a party outside the operator's control, what had been acquired and what was missing.
- The sharpest case is a review declared stable by an operator who also acknowledges that the underlying observations were incomplete. OpenShell documents the limits of observation; EVIDE can preserve the position the operator took and, through its Forensic Cross-Check (FCC), represent the structural incoherence of that position, within the limits of what was declared.
- EVIDE does not sign the event stream, attest logs, recover lost events or certify completeness.
- A deposit made before the source expires fixes two different things: the bytes EVIDE actually received, with the SHA-256 digest it computed on them, and the operator's declaration of which sources were acquired and which were not. After the deposited file is deleted at the end of its retention period, EVIDE keeps the record and the digest; checking the content then requires a copy held elsewhere, compared against that digest. That is the whole contribution, and it is bounded.

Purpose and scope

An audit system records what it observes. An evidentiary deposit establishes what was submitted to it, and when, within boundaries it declares. Neither, on its own, proves that the retained record contains every relevant event.

This note applies that distinction to NVIDIA OpenShell, the open-source runtime that sandboxes AI agents and enforces policy on their actions. It asks two questions. When the operational record of an agent action is no longer available, what can an independent examiner still verify, and what can no one establish? And what happens when an operator declares a review stable, despite acknowledging that the underlying observations were incomplete?

The note keeps two kinds of statement apart:

- **Documented limits.** What the OpenShell documentation itself states about retention, loss and completeness, each attributed to a specific documentation version.
- **EVIDE analysis.** Our reading of what those limits mean for later examination, and where an external deposit can or cannot help.

OpenShell's logging is substantial: structured OCSF events, JSONL and OTLP export, trace correlation, loss signalling and resumable cursors. This note does not argue that OpenShell lacks evidence tooling or is defective. It examines the distance between *recording* an event, *retaining* it and *proving the retained record is complete*, and the point at which OpenShell's own documentation hands that distance to the operator.

This is an independent reading of public documentation, accessed on 14 November 2026. It is not affiliated with, reviewed by or endorsed by NVIDIA. No integration exists between EVIDE and OpenShell. NVIDIA and OpenShell are trademarks of NVIDIA Corporation.

On 9 November 2026 NVIDIA announced the contribution of OpenShell to the Cloud Native Computing Foundation as a Sandbox project, with the repository moving to the `ai-openshell` GitHub organization. References in this note point to the locations available on the date of access.

Out of scope: NVIDIA Sentry (the BlueField-4 reference design) and NVIDIA Agent Policy Fabric, whose public documentation is currently published as a development version.

Boundaries at a glance

- OpenShell observes the agent and writes the operational record. EVIDE is not in that path and receives nothing from OpenShell directly.
- The operator decides what to collect, when, and from which sources. Only people, or the operator's own tooling, write to EVIDE.
- EVIDE preserves what it receives: declarations, references and, for a limited period, deposited files. Nothing flows back to the sandbox.
- No component, OpenShell or EVIDE, establishes that an event absent from the retained record did not occur.

Where OpenShell hands over to the operator (v0.1.8)

All statements below are taken from OpenShell documentation version v0.1.8, the current version at the time of access, unless another version is named. They are NVIDIA's own statements, and they define precisely where the operational record ends and the operator's responsibility begins.

Boundary	What the documentation states
Gateway buffer	The gateway keeps up to 2,000 recent log lines per sandbox in memory. The buffer is not written to disk and is lost when the gateway restarts or the sandbox is deleted. Each gateway replica keeps its own buffer.
Local files	Supervisor log files rotate daily in UTC and keep the three most recent files. On Docker and Podman they sit on a tmpfs mount, lost when the container stops; on Kubernetes, in an `emptyDir` volume removed with the pod. The documentation describes them as not a durable archive.
Durable retention	Durable retention requires collecting supervisor output and workload container logs into an external log store, arranged before rotation or teardown.
Loss in transit	The gateway push channel is bounded and can drop events under load or during disconnection. Local file writers are non-blocking and can also drop events under pressure. The console writer drops new lines when its 1,024-line queue is full, rather than blocking enforcement.
Loss signalling	The gateway distinguishes recoverable loss (a warning event: lines were skipped) from unrecoverable loss (the stream ends with `OUT_OF_RANGE`).
OTLP export of OCSF events	Records lost to a full queue, an oversized body or exhausted retries are reported on the next successful export in an `openshell.ocsf.gap` record, with the count and the time of first loss. Two cases fall outside this: "Records lost during shutdown produce no gap record." And records a collector rejects in a partial success are counted by the gateway as exported; only the collector's own metrics show them.
Separate sources	Gateway stream, supervisor files and workload container output are separate. Main process output is not in `openshell logs`; sandbox-runtime events are neither pushed to the gateway stream nor written to the supervisor's JSONL file. Each source must be collected separately.
Collection settings	Shorthand logging is active by default; full OCSF JSON export is opt-in (`ocsf_json_enabled`). OpenShell v0.1.8 emits OCSF v1.8.0 only. Legacy schema-version settings are deprecated and ignored.
Event identity	According to the current version (v0.1.8) of the OCSF JSON Export page, `metadata.uid` uniquely identifies an event and stays unchanged when the record is serialized again.
Trace correlation	OTLP log export links OCSF records to the active sampled span. A sampled ID does not guarantee export or retention in the trace backend, and gateway trace IDs are correlation hints, not evidence of the caller's identity.
Completeness	"Neither the local files nor the gateway stream guarantees complete audit history." The OCSF export page states that collection is best-effort, not a guarantee of complete audit history.

The design choice is coherent: enforcement takes priority over logging, and retention is delegated to the operator's infrastructure. The evidentiary consequence is that everything after the handoff (what was exported, when, from which sources, with which known losses) is recorded, if at all, by the same party whose conduct a later review may examine.

A documentation change worth preserving

The OpenShell documentation for v0.0.116, and again for v0.1.0, stated that "The log files inside the sandbox contain the complete record", including events the gRPC push channel could drop under load. The v0.1.8 documentation states that neither the local files nor the gateway stream guarantees complete audit history.

This is a verified textual difference. It is not, by itself, evidence that a guarantee was withdrawn from the same system: between those versions the architecture changed (the project's own issue tracker refers to a v0.1.2 architecture change), and in v0.1.8 the supervisor runs separately from the sandbox workload, with its own files and collection paths.

The evidentiary point is narrower and more general. A statement about completeness is meaningful only together with the version and architecture it describes. An examiner reviewing an incident months later needs to know which documentation applied when the record was produced, and that documentation may no longer be the version online.

Scenario: a disputed agent action after the record has rotated

An agent running in an OpenShell sandbox calls an external API. Policy allows the call, and the supervisor emits an OCSF event recording the allowed connection and the policy that permitted it. Thirty days later, a customer disputes that data was sent to that API.

The event records that a connection was allowed under a named policy. An allowed connection, or any enforcement event, does not by itself show that the specific content in dispute was transmitted or received. Whatever the retained record shows, that question may need other sources.

1. **During operation.** OpenShell enforces policy and emits events to the gateway stream, the supervisor files and, if configured, an external collector.
2. **Acquisition.** Shortly afterwards, the operator exports material: OCSF records for the period, any `openshell.ocsf.gap` records and warnings, collector metrics, the OpenShell version and collection settings in use, and the relevant documentation pages. This is an operator action, outside OpenShell and outside EVIDE.
3. **Deposit.** The operator deposits the export with EVIDE as a ZIP archive, then records the acquisition in a structured declaration that references that deposit. The two are separate, linked deposits (see *Records and their evidentiary limits*). If the operator also keeps a copy of the export, the deposit makes that copy checkable later.
4. **Source expiry.** Within days the local files rotate; a gateway restart clears the in-memory buffer; the sandbox is deleted. The collector's own retention runs its course.
5. **Review.** When the dispute arrives, an examiner compares what the operator presents now with what was deposited then, and with the sources the operator declared as acquired and not acquired.

The decisive variable is the order of steps 3 and 4. A deposit made before the source expires fixes the state of the record while it still existed. A deposit made afterwards can fix only what survived, and should say so. In neither case does the deposit add an event that was never collected.

This scenario is illustrative. It was not executed on an OpenShell deployment.

Three times that are not the same

Time	Clock	What it shows	What it does not show
Event time	The emitting component (supervisor or gateway)	When OpenShell recorded the event	That every event in the period was recorded or retained
Acquisition time	The operator's systems, as declared	When material was exported from the collector	That the collector held everything OpenShell emitted
Deposit time	EVIDE server, recorded on receipt	When EVIDE received the declaration or file	Anything about the event itself; it is not a qualified timestamp

A record can be authentic and still describe an incomplete period. The three times let an examiner say which of the two questions is being answered.

What an examiner can and cannot conclude

Question	Without an external deposit	With an EVIDE deposit	Not established by either
Is the disputed event in the retained record?	Depends on what the operator's collector still holds	Unchanged: EVIDE adds no events	Whether the event occurred if it is absent
Is the export presented today the export made at acquisition?	Depends on the operator's account	Compare the SHA-256 of the copy presented with the one EVIDE computed on receipt. The digest remains on record after the deposited file is deleted; the bytes do not	Whether the export itself was complete
Which sources were acquired, and which were not?	Depends on the operator's account	The operator's declaration, as made at deposit time	Whether the declaration is accurate
Were losses documented in the period?	Gap records or warnings, if they were themselves retained	Gap records and metrics, if deposited, fixed as of deposit time	Losses that produced no record, such as those at exporter shutdown
Which OpenShell version and documentation applied?	Current documentation, which may have changed	Declared version and, if deposited, the documentation pages as received	That the deployment matched the documentation
Is the retained record complete?	Not guaranteed by OpenShell's own statement	Not established by EVIDE	Not established by either

The value of the deposit is in the middle column, and it is bounded. EVIDE cannot recover a lost event, fill a gap or confirm completeness. It can make the state of the record at a given moment independently checkable: what existed, what was acquired, what was declared missing, and whether what is shown later is what was received then.

Four rules of claim discipline follow for any review:

- An event identifier (`metadata.uid`) supports the identity of that event, and the detection of duplicates. It does not show that every event was exported.
- A collector's receipt shows what the collector accepted, not everything the source emitted.
- A reported gap shows that a loss was detected. The absence of a gap record does not show that no loss occurred.
- A deposit shows what was submitted, by which account and when. It does not show that the submitted material is complete or correct.

Records and their evidentiary limits

EVIDE distinguishes material it receives from material it is only told about. The difference matters here because the two produce different evidentiary results.

Record	Channel	What EVIDE establishes	What remains declared
Log export deposited as a file	Human deposit, ZIP	The bytes received, their SHA-256 computed by EVIDE on receipt, the time of receipt, and the depositing account (Owner and DAPI in the EAR header). The bytes are retrievable only until their scheduled deletion	That the file is what the depositor says it is, and that it contains everything the collector held
Acquisition declaration	Human Structured Intake	Who declared what about the acquisition, and when, sealed with SHA-256 over the canonical record	The truth of the declaration: sources acquired, sources not acquired, known gaps
Reference to material kept elsewhere	`evidence_references`, submitted through the API, MCP or Human Structured Intake	That a reference with these fields was submitted at deposit time, inside the sealed record	The pointer, the declared origin, the declared retention status, any hash with its scope and who computed it. EVIDE does not fetch, receive or verify the referenced material.
Documentation pages	Human deposit, file	The pages as received at deposit time	That they match the deployed version

The acquisition declaration links to the file deposit through an evidence reference that carries the EVIDE identifier of the file deposit and the hash EVIDE computed for it. That reference is declared by the sender, like any other: EVIDE does not automatically resolve it or reconcile it with the deposited file. An examiner makes that comparison.

Declared retention status uses three values: `persistent_storage`, `rolling_buffer` and `unknown`. In this scenario the gateway buffer and the supervisor files are a `rolling_buffer`; an external collector is `persistent_storage` only if the operator declares it so. EVIDE preserves the declaration; it does not test it.

Practical limits apply. A log export can be deposited as a ZIP archive, subject to the service's upload size limit. Uploaded files are kept for a limited period set by the service configuration. When the retention period ends, the stored file is deleted; the record, the EVIDE-computed SHA-256, the file name, size and type, and the EAR remain, but EVIDE can no longer provide the original bytes. A later check of the content therefore requires a copy preserved elsewhere, whose digest can be compared with the one EVIDE recorded. EVIDE does not reconcile a declared hash against an uploaded copy of the same material. EVIDE is not a log archive: it complements the operator's log store and does not replace it.

A minimum declaration for this scenario

The following content is proposed for the scenario. It is not a new EVIDE schema, and EVIDE does not discover or validate any of it: each item is preserved as declared, in the existing fields indicated.

Item	Where it is recorded today
OpenShell version, compute driver and deployment context	Free text in the declaration
Sources acquired	One evidence reference per source, with declared origin and declared retention status
Sources not acquired, gaps, warnings and `OUT_OF_RANGE` statuses observed	Declared uncertainties; unresolved signals where the readiness is declared `verified_partial` or `unverifiable`, with the verification mechanism identified
Collection settings: JSON export enabled, OCSF schema version, OTLP export, collector identity	Free text
Acquisition interval and times, with the clock each comes from	Free text; the deposit time is recorded by the server
File set deposited, or pointer to material kept elsewhere, with any hash	Evidence references (pointer, hash, hash scope, who computed it)
Who decided to preserve the material or to stop collection, in which declared role, on what basis	The decision and authority fields of the Human Structured Intake

Separately, the OpenShell policy in force could be recorded as a declared operating perimeter through EVIDE ANCHOR, available through the API and MCP but not through Human Structured Intake. EVIDE would preserve it as declared; it would not compare it with the agent's behaviour.

A review declared stable on incomplete observations: Forensic Cross-Check (FCC)

One EVIDE mechanism is directly relevant to the review of a degraded record. It works in two distinct steps. First, the declarations are deposited and sealed: the intake hash covers what was declared, and nothing else. Then EVIDE derives an evidentiary profile from those sealed values and shows it in the EVIDENTIARY PROFILE section of the EAR. The derived profile is not part of the intake hash.

The first of the three indicators EVIDE derives in the profile, the Forensic Cross-Check (FCC), crosses the declared classification of the outcome with the visibility of the source, which EVIDE derives mechanically from the declared readiness. The full derivation matrix is public (see references).

The effect in this scenario: if the person closing the review declares the outcome stable while declaring that observation was partial (readiness declared as `verified_partial`, with the acquisition check named as the verification mechanism and its open signals listed: the gateway buffer rotated before export, the supervisor files were beyond the retained rotation), FCC does not return "stable". It returns "degraded". With unverifiable visibility and a provisional classification, it returns "unknown". The companion indicator DWC reports the unresolved signals as detected or, in the second case, critical.

OpenShell's gap records say that the stream lost lines. The EVIDE deposit preserves, with a party outside the operator's control, the reviewer's declarations: a review closed as stable, on a source the reviewer had declared incomplete. The FCC profile, derived afterwards from those sealed declarations, represents the structural incoherence between them. It is not an observation of OpenShell's logs.

Three conditions apply:

- FCC reads only declared values. It does not read logs, OCSF events, files or gap records.
- The "continuity" FCC reports is inferred by EVIDE from two declared values within this single record. It is not the continuity of the log stream, and not EVIDE's Evidentiary Continuity chain.
- FCC gives no verdict on whether the agent's action was correct.

The limit also runs the other way. If the operator omits a loss, or wrongly declares complete visibility, FCC cannot discover it: it has no access to the source. An FCC result of "stable" means only that the declared classification and the declared visibility are coherent with each other. It does not show that the OpenShell record is complete.

Because the profile is derived deterministically from the sealed declarations, anyone holding the sealed canonical record can recompute it with the public matrix and check that the EAR reports it correctly.

Adjacent work

Several efforts address parts of the same problem. They are listed here to mark boundaries, not to suggest any association with EVIDE.

- **OpenShell's own proposals.** Issues in the repository propose sequence numbers and a hash chain for the event stream (#3817), signed and chained action records using a key issued by the gateway (#4201), and OCSF 1.9.0 attestation and record integrity, opt-in and off by default (#5177). These would strengthen detection of loss and alteration within OpenShell's record, with keys held inside the operator's deployment. None is documented behaviour of v0.1.8.
- **Signing the stream externally.** Issue #4201 considers and sets aside signing the OCSF stream in a log shipper, because the events are already one hop from the supervisor and carry no chain: in its words, this "moves the trust problem rather than closing it". EVIDE does not sign the stream and does not attest logs. It records the acquisition, and what was declared missing, after the fact and outside the operator's systems.
- **Export manifests.** An external contributor proposed an export bundle whose manifest would report whether the requested interval is complete (#2745, opened 14 August 2026, closed as not planned on 21 September 2026). A design discussion opened by a community contributor describes an observation export manifest with event and drop counts (#2661). Neither is an OpenShell feature. Where an operator produces such a manifest, it is the natural object to deposit at the handoff.
- **Execution-record adapters.** The TRACE project publishes an OpenShell integration that binds a captured OCSF interval to an execution record.
- **Signed agent-action receipts.** Individual Internet-Drafts define signed, hash-chained receipts for agent actions, including a profile of the IETF SCITT architecture (RFC 9943). A SCITT receipt establishes what an issuer signed and that it was registered with a transparency service. It concerns statements the issuer chose to emit; it does not by itself establish that the issuer emitted every relevant statement, or fix what a later acquisition contained and lacked.

A low-cost step

Nothing in this note requires a change to OpenShell's enforcement or logging. One documentation-level step would make the handoff visible: describe the moment a record leaves OpenShell for durable retention as a single act consisting of the export, its manifest where available, and a deposit with a party outside the operator's control, made before rotation or teardown. Any operator can adopt this today with existing tools.

What EVIDE does not do

- It does not collect logs from OpenShell, the gateway, the supervisor or any collector.
- It does not sign the event stream or attest individual events.

- It does not recover, reconstruct or infer lost events.
- It does not attest that a retained record is complete, or that a gap record covers every loss.
- It does not verify a declared hash, a declared source or a declared retention status.
- It does not verify that deposited material is what the depositor says it is.
- It does not act as a long-term log archive.
- It does not assess OpenShell's security, configuration or conformity.
- Its receipt time is not a qualified timestamp.

A possible Lab follow-up (not performed)

The scenario could be tested in the EVIDE Governance Lab with a synthetic OpenShell sandbox and a documented event sequence: collect gateway and supervisor output through separate paths, interrupt one path, deposit the acquisition and its declared gaps, then repeat the examination after the source files have expired. The measure would be whether a reviewer can reconstruct the declarations, acquisitions and documented gaps, not whether every OpenShell event was generated. No such experiment has been run.

Status and references

Status: Version 1.0, 14 November 2026. Documentary analysis of public OpenShell documentation and repository content accessed on 14 November 2026. No OpenShell deployment was run, and no source-code audit of OpenShell, execution test or independent reproduction was performed. EVIDE-side statements checked against a production code copy of 13 November 2026. Not a security assessment, not a conformity assessment, and not a claim of integration.

References (accessed 14 November 2026):

- NVIDIA OpenShell v0.1.8, *Accessing Logs*. https://docs.nvidia.com/openshell/v0.1.8/observability/accessing-logs
- NVIDIA OpenShell v0.1.8, *OCSF JSON Export*. https://docs.nvidia.com/openshell/v0.1.8/observability/ocsf-json-export
- NVIDIA OpenShell (latest, v0.1.8 at access), *OCSF JSON Export*, for event identity. https://docs.nvidia.com/openshell/observability/ocsf-json-export
- NVIDIA OpenShell (latest, v0.1.8 at access), *Gateway Configuration File*, section OCSF Log Export. https://docs.nvidia.com/openshell/how-it-works/gateways/configuration
- NVIDIA OpenShell v0.0.116, *Accessing Logs*. https://docs.nvidia.com/openshell/v0.0.116/observability/accessing-logs
- NVIDIA OpenShell v0.1.0, *Accessing Logs*. https://docs.nvidia.com/openshell/v0.1.0/observability/accessing-logs
- NVIDIA/OpenShell, issues #3781 (v0.1.2 architecture change), #3817, #4201, #5177, #2745 and Discussion #2661. https://github.com/NVIDIA/OpenShell
- NVIDIA, *OpenShell contributed to the CNCF*, 9 November 2026. https://nvidia.github.io/OpenShell-Research/dev-notes/posts/2026-11-09-donated-openshell-to-cncf/
- CNCF, *Welcoming OpenShell to the CNCF: Open Governance to Secure Agentic AI*, 10 November 2026. https://www.cncf.io/blog/2026/11/10/welcoming-openshell-to-the-cncf-open-governance-to-secure-agentic-ai/
- TRACE, *Integration: NVIDIA OpenShell*. https://trace.agentrust-io.com/docs/integration/openshell/
- NVIDIA Newsroom, *Open Agent Safety Platform*, 28 September 2026. https://nvidianews.nvidia.com/news/open-agent-safety-platform
- IETF Internet-Draft draft-noa-scitt-ai-agent-receipt-01, *A SCITT Profile for AI-Agent Action Receipts*. https://datatracker.ietf.org/doc/draft-noa-scitt-ai-agent-receipt/
- IETF Internet-Draft draft-sahu-agent-action-receipts-00, *Signed, Hash-Chained Action Receipts for AI Agents*. https://www.ietf.org/archive/id/draft-sahu-agent-action-receipts-00.html
- EVIDE, *EVIDE Artifacts: External References vs. Received Deposits*. https://app.certifywebcontent.com/docs/evide-artifacts/
- EVIDE, *The Evidentiary Artifact Record (EAR)*. https://app.certifywebcontent.com/docs/evide-ear/
- EVIDE, *Forensic Cross-Check (FCC)*, including the derivation matrix. https://app.certifywebcontent.com/docs/forensic-cross-check/
- EVIDE, *EVIDE ANCHOR*. https://app.certifywebcontent.com/docs/evide-anchor/
- EVIDE, *Technical Scope Note 01: Robotic Safety Decisions and Evidentiary Boundaries*, v1.0, 14 November 2026. https://app.certifywebcontent.com/working-papers#tsn-robotic-safety-decisions



emanuel
celano
14.11.2026
19:38:52
GMT+01:00