Comment on *EMV® Agentic Payments – Framework for Specifications* v1.0 DRAFT

Narrative companion to the completed EMVCo feedback form (12 comments). Section and page references are to the v1.0 DRAFT (August 2026).

To: EMVCo, LLC — Agentic Payments working group
From: Decionis, Inc. (Festus B. Jejelowo, founder) — festus@decionis.com
Re: Public comment on *EMV® Agentic Payments – Framework for Specifications*, v1.0 DRAFT
Date: 6 September 2026

1. Who is commenting, and in what capacity

Decionis operates an independent execution authority for actions initiated by AI agents and other automated actors: a proposed action is evaluated against the accountable organisation's own policy before it commits, the answer is one of ALLOW, ESCALATE or BLOCK, and each evaluation leaves a signed, portable record that a third party can verify without access to our systems. The same mechanism gates agent checkouts today through the Agentic Commerce Protocol and the Agent Payments Protocol, and maps onto the Universal Commerce Protocol's own primitives.

We are not an EMVCo member, subscriber or associate, and we do not operate on card rails. We comment as an implementer of the function the draft calls Fulfilment evaluation against a Payment Mandate and its Constraints (§4.3.5), and of the shared state it depends on (§5.5), because the hard cases the draft names — agent-managed recurrence, budget-based Fulfilment, partial execution and mixed outcomes under a single Intent (§5.4), and post-transaction support (§5.1) — are the cases where we have had to decide, in production, what "within bounds" means at the moment an action commits. Everything we assert about our own mechanism below is public and checkable: a technical note with a DOI (https://doi.org/10.5281/zenodo.22081954), a reproducible corpus of signed decision records with a DOI (https://doi.org/10.5281/zenodo.22312956), and an open verifier on npm (`@decionis/verify`).

2. Summary of positions

1. **Intent must be bound to the exact transaction at commit time, not only registered before it.** Registering intent solves discovery; it does not solve the gap between "what the consumer authorised" and "what actually clears". We recommend the framework define a pre-commit revalidation step against the transaction's exact attributes, and require its outcome to be recorded (§ 3.1).
2. **Reserve a third outcome.** "Within intent" and "outside intent" leave no room for the most common agent failure — a small, explainable overrun of a cumulative budget. A held/escalated outcome that returns control to the consumer or their

delegate for a bounded window, and records the human's answer, prevents the framework from forcing every edge case into approve-or-decline (§ 3.2).

3. **An unreachable intent evaluation should hold an agent-initiated payment, not wave it through.** Fail-closed as the default for agent-initiated transactions, with an indicator that distinguishes "intent could not be evaluated" from "outside intent" (§ 3.3).
4. **The evaluation should leave verifiable, portable evidence — not a log entry.** A signed record naming the intent reference, the constraint set and its version, the exact attributes evaluated, the verdict and any human answer, verifiable against a published key by a party who does not run the evaluator (§ 3.4).
5. **Intent over time needs state-current, atomic consumption.** Cumulative budgets and recurring authorisations must be evaluated against the authoritative record of what has already been consumed, and consumption must be atomic, or two concurrent agent transactions will each fit the remaining budget (§ 3.5).
6. **Agentic Transaction Indicators should carry a reference to the evaluation record**, not only a flag that an agent was present (§ 3.6).
7. **Know Your Agent identifies; it does not authorise.** The framework should keep agent identity and intent authority as distinct layers with distinct evidence (§ 3.7).

3. Detailed comments

3.1 Bind intent to the action at commit time (§4.3.5 Fulfilment, p. 29–30; §6.7.6 Constraint Structure and Evaluation, p. 48; §5.6, p. 35)

The framework's own scenarios — recurring purchases, cumulative budgets, post-transaction activity — are scenarios in which the world changes between registration and commit: prices move, a budget is partly consumed by another channel, a recurring instruction is amended. A check performed when intent is registered is therefore evidence that the action was acceptable when evaluated, not evidence that the exact effect that committed was still authorised.

We recommend the framework specify, as a distinct step, **pre-commit revalidation**: immediately before the authorisation request is submitted, the referenced intent is re-evaluated against the transaction's exact attributes (amount and currency, merchant or acceptor, timing, the recurrence or budget position, and any condition attached to the consumer's authorisation), and the result is part of the record. Our implementation revalidates payload digest, target, policy version, material signals, any human approval receipt, validity window and a single-use nonce at exactly this point, and the technical note above shows what happens without it: in a deterministic harness, request-time authorisation alone allowed stale execution in half of contested attempts, and replay and substitution in all of them, while execution-bound authorisation allowed none. (Those figures were measured in an in-process harness, not over a network, and are cited only for the shape of the result.)

3.2 A third outcome: held for the consumer, with a bounded window (§5.4 Lifecycle Model, p. 34; §4.1 Execution Mode, p. 22–23; §6.9.2, p. 49)

Card authorisation is binary. Agent-initiated payments are not, and the framework has the chance to say so. The case that occurs most often in practice is not fraud but a small, explainable overrun — a basket that clears a cumulative budget by a few percent because of a shipping fee the agent did not anticipate. Forcing that case into decline-or-approve either blocks legitimate purchases or trains consumers to authorise generously, which erodes the meaning of intent.

We recommend an explicit **escalation outcome**: the transaction is held, the consumer (or a delegate the consumer named at registration) is asked a specific question with the specific overrun, the answer is time-bounded and single-use, and the answer is recorded as evidence inside the same record as the evaluation rather than as a separate approval. In our implementation the answer is a verified-presence receipt bound to the approver, the approved action digest, inherited conditions and an expiry, and the held action is then re-evaluated with that receipt as an input — a human approval is evidence within a new evaluation, not a transferable override. We note that UCP already reserves an equivalent primitive (an outstanding action that prevents a checkout from completing until it is processed), which suggests the concept is portable across protocols.

3.3 Unreachable evaluation: hold by default for agent-initiated payments (§6.7.6, p. 48; §5.5 Shared State and Replay Considerations, p. 34–35; §8, p. 54)

When the party evaluating intent cannot be reached, the framework should state the default. For a present cardholder with bounded exposure, availability may reasonably win. For an agent acting without a human present, and with the framework's own scenarios of cumulative budgets, we recommend the default be to **hold**, and that the resulting indicator distinguish "intent could not be evaluated" from "outside intent", so issuers and merchants can measure the two separately and no outage is later read as a policy decision.

3.4 Evidence a third party can verify (§5.6 Orders as Units of Fulfilment, p. 35; §5.1 Post-Transaction Support, p. 32; §5.3, p. 33; §6.9.2, p. 49; §6.7.3, p. 47)

Post-transaction activity — disputes, chargebacks, audits — is where the framework's value is tested, and a log entry held by the party that made the decision is weak evidence there. We recommend that each intent evaluation produce a **signed, portable record** containing at minimum: the intent reference; the constraint set and its version (so the rule can be pinned, not described); the exact transaction attributes evaluated; the verdict; any human answer and who gave it; and a timestamp inside the signed envelope. The signing keys should be published as a key set with a rotation grace window, so that a merchant, issuer, network or auditor can verify the record without contacting the evaluator.

Two properties of such a record are worth distinguishing in the framework text. A **signature** proves the record was not altered after the fact. **Re-derivation** — re-running the recorded inputs through the recorded constraint set and obtaining the same verdict — proves the answer was the one the constraints required. A framework that specifies only the first will get records that are tamper-evident and still wrong. Our public corpus exists

to make this distinction checkable: it contains signed synthetic records including the negative cases a verifier must refuse (a mutated verdict fails both digest and signature; a claim grafted from one record onto another breaks the signature and is not reported), and the private key is deliberately published so a reader can regenerate the corpus rather than trust fixtures we shipped. The corpus proves the verifier; only a live key set proves a production record, and the repository documents that check separately.

3.5 Intent over time: state-current evaluation and atomic consumption (§5.5, p. 34–35; §4.3.3 Constraints, p. 28–29; §6.3.2, p. 42; §6.7.6, p. 48)

A cumulative budget is a shared mutable resource. If two agent transactions are evaluated concurrently against the same remaining budget, both fit, and the consumer has authorised neither outcome. We recommend the framework require that (a) the evaluator name the authoritative state it evaluated against — the intent service's own ledger of prior consumptions, with a position reference — and (b) consumption of budget or of a recurrence slot be **atomic and single-use**, so that a second evaluation against the same position fails rather than succeeds. The correctness criterion we use is policy-state serializability, introduced by Peng and Wu (2026): each committed effect remains explainable against a state consistent with the effects ordered before it. Our implementation claims the nonce and the mutable scope atomically at commit; unrelated scopes stay independent so throughput is not sacrificed for correctness.

3.6 Agentic Transaction Indicators should reference the evaluation (§8 Agentic Transaction Indicators, p. 54–55)

A flag that an agent was involved is useful for reporting. A pointer to the signed evaluation record — its identifier and where its key set is published — turns the indicator into something a receiving party can act on: an issuer can decline a transaction whose record it cannot verify, a merchant can attach the record to a dispute response, and post-transaction activity can reference exactly what was evaluated. We recommend the indicator reserve an optional field for that reference now, even if its format is specified later.

3.7 Know Your Agent is a precondition, not an authorisation (§3 Know Your Agent, p. 20–21; §5.2, p. 33)

Identifying the agent, attesting its attributes and establishing that a consumer delegated to it are necessary and are the province of identity. None of them decides whether this specific transaction, right now, is within the consumer's intent. We recommend the framework keep the two as separate layers with separate evidence — an agent identity assertion, and an intent evaluation record that references it — so that a valid agent credential is never read as authorisation of the action. In one line: access is not authority, and detecting an agent is not authority either.

4. Suggested framework text

In the draft's own vocabulary (Fulfilment, order, Mandate Pair, Constraints, Autonomous Fulfilment artefact), and knowing the framework is informative, offered as candidate wording:

1. "Fulfilment evaluation is performed against the final values immediately before the payment authorisation request is submitted, against the current lifecycle and state information (§5.5), and its outcome is materialised as an Autonomous Fulfilment artefact (§5.6)." (§4.3.5, §6.7.6)
2. "A Fulfilment evaluation yields one of: within bounds; outside bounds; or held pending a bounded, single-use act of Consumer consent over the closed values, recorded in the same artefact." (§5.4)
3. "Where the state needed to evaluate a stateful Constraint is unavailable at Fulfilment time in autonomous mode, the Fulfilment is not treated as within bounds; the outcome is recorded as unevaluable, and the Agentic Transaction Indicator distinguishes it from outside bounds." (§6.7.6, §8)
4. "An Autonomous Fulfilment artefact carries the Intent and order references, the identifiers and version of the Constraints evaluated, the final values, the state position used, the outcome, any Consumer consent used to resolve a held order and its signer, and a timestamp inside the signed envelope; it is signed by the evaluating party with a key resolvable through a published JWKS endpoint, as §6.7.3 already assumes for Layer 1." (§5.6, §6.9.2)
5. "Consumption of a budget amount or a recurrence occurrence is atomic and single-use per order, the evaluation references the state position it used, and that state is held by a participant other than the Agent." (§5.5)

5. Questions for the working group

1. Which party is expected to operate the intent evaluation — the intent service itself, the issuer, the wallet, or a third party the consumer or merchant designates — and may that party be architecturally separate from the agent and from the rail it governs?
2. In dispute handling, what standing does an evaluation record have when it shows an outside-intent or held outcome that was nonetheless submitted for authorisation?
3. Will the framework specify key distribution for evaluation records, or leave it to participants?
4. Section 6 describes Verifiable Intent as maintained externally and routes structural feedback to that repository. When a framework-level semantic (an outcome category, a state position, the content of a Fulfilment artefact) needs a field in a VI mandate or Layer 3 credential to be carried end to end, which body owns that change — and will the framework define those semantics independently of VI, so that a non-VI Intent mechanism can conform?

6. What Decionis operates today, stated with its limits

Operational: pre-execution verdicts (ALLOW / ESCALATE / BLOCK) against the organisation's own policy; execution-bound Decision Dossiers carrying payload digest, target, policy snapshot, validity window and single-use nonce; pre-commit revalidation with atomic nonce and scope claim; an escalation lifecycle in which a named person's verified approval is bound to the action and re-evaluated; a fail-closed default for unbounded machine-initiated actions; Ed25519-signed records verifiable against a

published key set (`https://api.decionis.com/.well-known/decision-dossier-jwks.json`); a public corpus (DOI 10.5281/zenodo.22312956) and an open verifier (`npx -y @decionis/verify`); Agentic Commerce Protocol webhook signature verification and Agent Payments Protocol mandate verification in production; a verdict mapping onto UCP primitives.

Not operational, and not claimed: any participation in card-network authorisation flows or intent APIs; membership, certification or endorsement by EMVCo or by any protocol named above; inbound verification of UCP requests (the mapping is outbound only); network-latency figures — the benchmark cited in §3.1 ran in an in-process harness and is offered for the shape of the result, not as a performance claim.

7. Artifacts

- Jejelowo, Festus B. "The Execution Verifiability Gap: Why Model Governance Cannot Authorize Consequential Actions." Decionis Research, version 1.0, 24 August 2026. https://doi.org/10.5281/zenodo.22081954
- Jejelowo, Festus B. "Agent-Safe Pipeline: Reproducible Execution-Authority Reference Architecture and Decision Dossier Corpus." Zenodo, version v0.1.3, 5 September 2026. https://doi.org/10.5281/zenodo.22312956
- Verifier: https://www.npmjs.com/package/@decionis/verify — reproduce the corpus checks with `npx -y @decionis/verify --file dossiers/vectors/owned-execution-bound.json --jwks dossiers/corpus-jwks.json`
- Reference architecture and threat model: https://github.com/decionis/agent-safe-pipeline

We would welcome the opportunity to walk the working group through the mechanism, and to test the framework's Fulfilment-evaluation semantics against a running evaluator once the draft is stable.

— Decionis, Inc.