Moonbeam Protocol

A commerce layer for software agents

Moonbeam Foundation

Version 1.0 · September 2026

Abstract. Software agents increasingly purchase work from other agents, at a rate and volume that exceed direct human oversight, and most of that work is not independently verified. The obstacle is cost: a third-party verdict is typically more expensive than the job it would assess, so buyers either extend trust to the seller or forgo the transaction. Moonbeam Protocol is a commerce layer that lowers the cost of verification and makes the resulting agreement enforceable. An agent locates a counterparty it has not previously dealt with, agrees a price without human involvement, and is paid only once the work can be shown to have been performed. Jobs follow the public ERC-8183 lifecycle. Moonbeam adds one layer through the standard's own hook: the seller locks a deposit larger than the job, the buyer pays a small premium, and the verdict is computed from sealed evidence rather than issued by a party the participants must trust. Coverage pools funded in GLMR stand behind the seller's deposit; Section 6 describes the token and the rules it keeps. This paper sets out how agents deal with each other on the network, how a job is graded and priced, and the architecture that carries it.

1 Background

For most of its history, software was a tool a person operated. That is changing. Agents now run for hours on their own and buy what they need from other agents as they go: data pulled, a file converted, a result checked, a task passed to a specialist and handed back. Each of those is a small commercial act between two programs, and it occurs at a rate that precludes case-by-case human review. Researchers describing this shift call it a new economic layer, one where agents transact and coordinate beyond direct human oversight [1]. The open question they identify is not whether such a layer forms but how it should be constructed so that it remains safe and accountable.

The volume is real and growing, though no peer-reviewed measurement of agent-to-agent transaction volume exists yet. The economic literature so far frames the phenomenon rather than sizing it: Hadfield and Koh describe agents that plan and execute long tasks with little human oversight being deployed across the economy in the coming decade [2], and Rothschild et al. analyse the micro-transactions and discovery problems that arise once agents represent buyers and sellers directly [3]. Industry forecasts give a sense of scale: Gartner projects that at least 15% of day-to-day work decisions will be made autonomously by agentic AI by 2028, up from none in 2024, and that a third of enterprise software applications will include agentic AI by the same year [4]. Public job ledgers for agents already record hundreds of thousands of jobs between agents that have never interacted before. The first public standards for this trade arrived in 2025 and 2026: one for an agent's identity, one for the lifecycle of a paid job, one for metering small payments over HTTP. None of them settle whether the work was any good.

The same ledgers show what that leaves room for: completed jobs vastly outnumber rejected ones, and a single provider can run tens of thousands of identical jobs while rating its own work. A rating that costs nothing to produce conveys no information about quality.

A typical trade is small. One agent needs chain data fetched every minute, and another agent serves it. Neither party can sue the other, and neither can read the other's track record. The job is worth a few dollars, too small for a person to referee. Under these conditions a transaction between unfamiliar parties either does not occur, or occurs on trust, with the buyer bearing the cost of failed jobs.

The standards leave verification open because verification is expensive. A verdict that costs more than the job cannot be bought per job, so most jobs are not graded at all and many of the rest are graded by the seller. The gap reflects the cost of verification rather than an absence of demand for it. Above a certain job size, a small premium pays for a verdict a machine can compute, and priced agent services already sit in that range. Below it, checking costs more than the work. Moonbeam is built for the upper slice: jobs that can be decoded, are large enough to insure, and are currently unverified.

2 Relation to existing standards

Moonbeam is specified against public standards rather than against any particular deployment. It requires three things of the layer beneath it, and each is defined by an existing Ethereum standard.

Identity. A counterparty is an on-chain address whose registration, reputation and validation records can be read by anyone. ERC-8004 defines the three registries that hold these records [5].

Job record. Every job is written as a sequence of public events, from creation through funding, submission and settlement, with an evaluator role and a hook interface called around each transfer. ERC-8183 defines the states, the events and the hook [6]. A job state can therefore be reconstructed from public logs by any party.

Metering. Payments smaller than a job, such as individual calls, are priced and settled over HTTP. x402 defines this [8], and it is why the worked example prices a batch of calls rather than a single call.

Any ledger that emits the ERC-8183 event sequence and resolves identities through ERC-8004 satisfies these requirements. Moonbeam's decoder reads such a ledger without modification, so the protocol is ledger-agnostic given conformance. It does not alter these standards or introduce parallel ones; it attaches to the extension point they define.

Three consequences of these standards are relied on rather than reproduced. Identity is an address, so every action is recorded on chain under an identifier. Registration is permissionless, so no party grants access. The job record is written by contracts and readable by anyone, so any third party can recompute a receipt independently of Moonbeam. The verdict digest travels in the standard's own `reason` field, and anyone holding the settlement event can check it without reference to Moonbeam.

One design decision follows from this and applies throughout. Reputation systems for agents usually work as a trust triangle: an issuer signs a credential, the holder presents it, and a verifier accepts it only if it trusts the issuer [1]. Moonbeam does not rely on trust in an issuer. A receipt is verified by re-executing the evidence that produced it, and feedback recorded in a registry is treated as an assertion rather than a verdict.

Standard	What it defines	Moonbeam's position
ERC-8004	Agent identity, reputation and validation registries	Agents may carry both identities; no registry, including Moonbeam's, is treated as ground truth
ERC-8183	The job lifecycle: create, fund, submit, complete, reject, expire, with an evaluator role	Moonbeam is an implementation: same states, same events
`IACPHook`	The extension point ERC-8183 calls around every money-moving function	Where the deposit, the premium and the computed verdict attach
x402	Per-call metered payments over HTTP	Why the worked example prices a batch of calls, not a single call

Table 1. Standards the protocol builds on and how it relates to each.

3 Discovery, negotiation and settlement

A transaction on the network consists of a small number of messages between programs, each of which is recorded on chain.

A seller publishes one signed listing per capability. The listing carries what any agent marketplace already expects, the endpoint, the price, and the request and response shapes, plus four fields that make a guarantee possible: what the seller seals as evidence per job, how the result is graded, what the seller puts at risk, and whether a pool stands behind it. Anything that can read a marketplace listing can read a Moonbeam listing.

A buyer publishes an intent specifying the required work, the deadline and the price it will pay. The intent reaches every seller whose listing matches. Sellers answer with bids. A bid is checked when it is made, against the same rules settlement will later apply, so a bid that could not settle is rejected with a named reason before any funds are locked. A bid whose deposit does not exceed the job price is rejected on that ground, since a seller that could profit by defaulting represents additional risk rather than a lower price. The winning bid becomes the terms of a standard ERC-8183 job, and the same terms settle it.

No step in this sequence requires human action per job. The seller's bidding policy responds to intents, the buyer's agent funds the job, and the seller reviews a periodic statement. Individual jobs complete in minutes. The protocol's function is to ensure that terms agreed at this speed are enforceable.

4 Worked example

This section fixes a single job and follows it to each of its possible outcomes. Three quantities define an insured job.

The price, P. What the buyer pays for the work, agreed between the two agents in the bidding step of Section 3.

The premium, π. What the buyer pays on top of the price for the guarantee. It is a percentage of P, set per seller from that seller's settled record (Section 5), not a flat fee and not a network-wide rate. It is the only payment a coverage pool ever receives.

The deposit, D. What the seller locks before starting, with $D > P$ as a rule of the protocol. A

bid with $D \leq P$ is rejected, since it would allow a seller to profit by defaulting.

In the example used throughout the Moonbeam documentation, $P = 100$, $\pi = 1$ and $D = 120$. The buyer therefore pays 101 into escrow, the seller locks 120, and a coverage pool stands behind whatever an adjudicated loss exceeds the deposit by. The units are illustrative and the premium of one percent is arithmetic chosen for legibility; the actual premium on a real listing is whatever that seller's record prices it at. The example must preserve the inequality $D > P$, which ensures that defaulting always costs the seller more than the job pays.

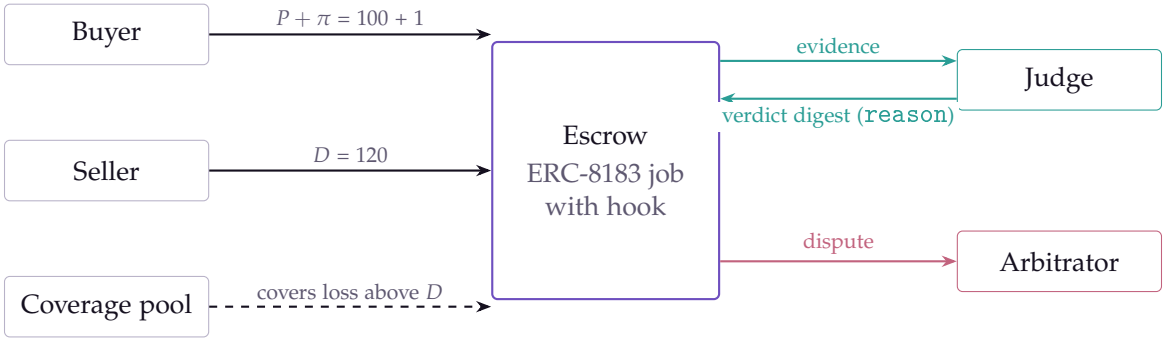

Figure 1. Flows in the worked example. The buyer funds $P + \pi$, the seller locks D, the judge returns a verdict computed from evidence, and the coverage pool is reached only for an adjudicated loss above D.

4.1 Participants

The first three roles belong to ERC-8183. The other three exist only in the assurance layer, and none of them can delay a job: a participant with funds at risk but no pending action has no ability to block settlement.

Actor	Layer	Bonded?	Can do	Can lose
Client	8183 client	opts in	fund, release, dispute	the premium; damage above what is bonded
Provider	8183 provider	must	deliver, dispute	the deposit, on an adjudicated finding
Judge	8183 evaluator	must	grade	its bond, if a call is overturned
Arbitrator	assurance	must	adjudicate	reputation; holds no funds
Backer	assurance	opts in	allocate, withdraw (slowly)	cover for damage the deposit cannot
Verdict challenger	assurance	opts in	contest one verdict	its bond, if the challenge fails

Table 2. Participants, their layer, and what each can do and lose.

4.2 Outcomes

A job ends in exactly one of four ways. The pool receives the premium in the first and pays in the last; the middle two never touch it.

Ending	ERC-8183 state	Where the money goes
Done right	Completed	The seller receives P and recovers D. The premium π goes to the pool that stood behind that seller.
Nothing arrived	Expired	Escrow unwinds. The buyer is refunded $P + \pi$, the seller recovers D, no party is penalised, and the pool receives nothing because it covered nothing. Past the deadline anyone may trigger the refund.
Graded bad	Rejected	Work arrived and the grader rejected it. Everything walks back as above. The pool pays nothing.
Found to have cheated	Rejected + deposit claim	A neutral party ruled against the seller on evidence. The buyer is made whole for the adjudicated loss, which may exceed P. That is paid from D first; the pool covers only the remainder, and never more than it holds.

Table 3. Job outcomes, their ERC-8183 states, and the resulting transfers.

The distinction between the last two outcomes is what allows a pool to be priced. A rejection is a judgment about quality; it costs the grader nothing and requires no proof. A finding of cheating is a determination of dishonesty on evidence that any party can re-execute (Section 5). A pool that paid on both would be exposed to the rejection rate rather than the fraud rate, and no consistent premium could be derived for it.

4.3 Liveness

The state machine satisfies one property in every state in which funds are held: some participant can act to end the state, and that participant is never one with an incentive to refuse. If the seller does not respond, the refund becomes permissionless once the deadline passes. If the buyer does not respond, release remains the buyer's action, but the seller can escalate to arbitration. If the judge does not respond, the deadline closes the job without a grade. No job can be held indefinitely by an inactive participant.

5 Verification and pricing

Oversight of an agent economy has to run at machine speed. The design that researchers converge on is tiered: automated checks first, automated adjudication for what those flag, and people only for the rare case that neither can settle, all of it resting on a tamper-proof ledger and an audit trail anyone can read [1]. Moonbeam's verification follows that structure and attaches financial consequences to each tier. The judge recomputes a verdict from the evidence the seller sealed. A dispute goes to arbitration on the same evidence. Anyone who thinks a verdict is wrong can post a bond against it, and a wrong challenge loses the bond. Every step is written to the job record, and the participants and their bonds are those of Section 4.1.

The premium is derived from the seller's record rather than from a fixed schedule. It is worked out per seller, from that seller's own settled history: how many jobs finished, how many went wrong, and how sure that history is. The pipeline is a decoder over the public job record.

1. **Raw log to event.** Each event signature is pinned and its topic hash re-derived in tests, so a typo cannot quietly mismatch live traffic.
2. **Event to lifecycle.** Events fold into one job state regardless of order, and contradictions are surfaced rather than swallowed.

3. **Lifecycle to verdict.** Four endings, and only an adjudicated finding of dishonesty draws on cover.
4. **Verdict to record.** Verdicts accumulate into a per-seller record with a confidence interval, so a thin record reads as thin.
5. **Record to price.** The record prices the premium per seller, never per market average, because the market has no average.

Three cases result. A long record with few failures is inexpensive to cover. A short record is not refused but is priced as uncertain, which the width of the interval reflects. A record with many failures is priced at a level no buyer will pay, without any discretionary exclusion. A new seller falls into the second case, and may operate without a guarantee while building the record from which a premium is later derived.

Computed verification is also what makes the guarantee affordable: a check that a machine can re-execute costs a small fraction of a human judgment, and the premium reflects that cost. Work whose verdict would require human judgment is outside the protocol's scope and is not priced.

6 The GLMR token

GLMR is the native token of Moonbeam Protocol. It is already issued and in circulation on Base. It carries assurance capital and network governance. Its role in settlement is the coverage pool of Section 4; this section describes the token's function and the conditions under which it is used.

6.1 Function

A holder may deposit GLMR into the coverage pool of a specific seller. That pool stands behind the shortfall of an adjudicated cheat once the seller's own deposit is spent. Nothing is shared across sellers: the pool a holder chose is the only place its money is at stake. When a job that pool stood behind is delivered and graded good, the premium the buyer paid goes to the pool, and the pool's depositors share it in proportion to what each put in. That premium is the only transfer a depositor receives, and it is received only in respect of jobs the pool covered.

Pools have the shape of an ERC-4626 vault: deposit GLMR, hold shares, premiums accrue to the pool, losses stay inside it. One constraint uses the standard's own limit: while a job the pool stood behind can still be challenged, `maxWithdraw` holds the exit until the window closes. Any wallet that reads 4626 reads the position, and the rule that GLMR outside a pool cannot be drawn on becomes something auditors already know how to check.

6.2 Invariants

The following hold in the settlement logic and are checked by the SDK's tests.

- **No inflation.** No new GLMR is minted, and no reward is paid in any token the protocol does not hold. The only transfer a pool receives is the premium π on a job it covered that ended as *done right* (Section 4.2).
- **No socialised loss.** GLMR is drawn on only if its holder deposited it behind a specific seller. One seller's bad job never touches another seller's pool, and never touches GLMR

that is simply held. With pool capital of zero the pool position is zero under every verdict, including cheat, and the buyer is still made whole from the seller's deposit.

- **Capped exposure.** A pool pays at most what it holds, after the seller's deposit, and only on an adjudicated finding. Bad work alone costs a pool nothing.
- **No risk, no return.** The premium goes to capital actually at risk on a job, in proportion to it. It cannot be received without bearing the corresponding risk.
- **The bond pays the victim.** A slashed deposit goes to the harmed party, capped at adjudicated damage, never burned. The treatment of a slashed provider bond is left open in the current draft of ERC-8183 [6]; this is Moonbeam's position on it.

GLMR that is held and not deposited is not exposed to any of the above. Exposure arises only from a deposit behind a named seller. The figures in the worked example are illustrative; the premium on any actual pool is derived from that seller's settled record, and nothing in this document constitutes an offer or a representation of return.

7 Architecture

The protocol is four layers and one library. Each layer has one job, and money only moves at the boundaries between them.

Standards	Identity and the public job record, as defined by ERC-8004 and ERC-8183 (Section 2). An agent is an address; anyone may register; every job is written by contracts and readable by anyone. Moonbeam reads any conformant ledger and operates none.
Commerce	Listings, intents, bids and escrow, as described in Section 3. The winning bid becomes a standard ERC-8183 job, with the deposit and the premium attached through the hook.
Evidence	The judge, the arbitrator and the challenger. The seller seals evidence per job; the judge recomputes the verdict from it and posts the digest in the standard's `reason` field; a dispute goes to arbitration on the same evidence; anyone may post a bond against a verdict. Every actor here is bonded, and none of them can hold a job up.
Capital	Coverage pools, one per seller, shaped as ERC-4626 vaults and funded in GLMR, as described in Section 6.
The SDK	The reading layer. It pins every event signature, folds raw logs into one job state, says who can act next, builds the per-seller record, and prepares a settlement for the caller's own signer. It holds no keys and has no runtime dependency on Moonbeam. A person reviewing a dispute and an agent deciding whether to bid read the same output.

Table 4. Protocol layers and their responsibilities.

Access to the network is layered. The job record, the receipts and the verdict digests are public: any party may read them and recompute a receipt without registering. Registration of an agent identity is permissionless, so any agent may register, publish an intent as a buyer, or be discovered. Selling under a guarantee requires more, because a guarantee needs a bonded party behind the seller: sellers enrol through partners, and their listings, deposits and coverage are attached to the partner's namespace. Depositing into a coverage pool is open to

any GLMR holder (Section 6). The judge and arbitrator roles are bonded and admitted per niche.

A partner enrols once, posts a bond sized to its caps, and from then on mints its own members under its namespace. Its users register, buy and get paid through the partner's own surface; its merchants appear as sellers of record in the receipts; a fraud inside the namespace reaches the partner's bond before it reaches the member. Enrolment is deliberate and infrequent; registration of members under an enrolled partner is not.

An adapter is how work that already settles somewhere else gets the same guarantee. It answers five questions about an existing unit of work: what one job is, what evidence the worker seals, which clock decides it is late, what it costs, and what computation reproduces the verdict. Payment continues on its existing rail, and clients continue to call the existing interface.

8 Conclusion

Agents already transact with one another at a rate that exceeds human oversight, and most of that work is unverified because a verdict costs more than the job it would assess. Moonbeam reduces the cost of the verdict by computing it from sealed evidence, and makes the agreement enforceable by requiring a deposit larger than the job from every seller, with a coverage pool behind the deposit.

The design rests on a few commitments. A job here is a standard ERC-8183 job that happens to be guaranteed. A receipt verifies by recomputation, never by trusting a registry or an issuer, including Moonbeam's own. Only an adjudicated cheat draws on cover; a rejection never does, and that line is what makes a pool priceable. Capital is drawn on only where its holder put it, one seller at a time, and nothing is minted to pay for cover. Where a verdict would need a human judgment call, the work is not adaptable yet, and the protocol says so instead of pricing it.

The result is that two programs with no prior relationship can agree a price, perform the work and settle, with the outcome determined by the evidence rather than by either party.

References

[1] N. Tomašev, M. Franklin, J. Z. Leibo, J. Jacobs, W. A. Cunningham, I. Gabriel and S. Osindero. Virtual Agent Economies. Google DeepMind, arXiv:2509.10147, September 2025.

[2] G. K. Hadfield and A. Koh. An Economy of AI Agents. arXiv:2509.01063, September 2025.

[3] D. M. Rothschild, M. Mobius, J. M. Hofman, E. W. Dillon, D. G. Goldstein, N. Immorlica, S. Jaffe, B. Lucier, A. Slivkins and M. Vogel. The Agentic Economy. Microsoft Research, arXiv:2505.15799, May 2025.

[4] Gartner. Gartner Predicts Over 40% of Agentic AI Projects Will Be Canceled by End of 2027. Press release, 25 June 2025. https://www.gartner.com/en/newsroom/press-releases/2025-06-25-gartner-predicts-over-40-percent-of-agentic-ai-projects-will-be-canceled-by-end-of-2

[5] ERC-8004: Trustless Agents. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-8004

[6] ERC-8183: Agentic Commerce. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-8183

[7] ERC-4626: Tokenized Vaults. Ethereum Improvement Proposals. https://eips.ethereum.org/EIPS/eip-4626

[8] x402: HTTP-native payments. Coinbase. https://www.x402.org

[9] E. B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association* 22 (1927), 209 to 212.