One-Action-One-Mint Transaction Paradigm

1. Abstract:

A method for uniformly recording and propagating system events in a decentralized memory-driven ledger by encapsulating each discrete logical action as a single on-chain mint transaction that carries a cryptographic attestation of inputs and outputs, thereby reducing state bloat, enhancing auditability, and enabling coordinated downstream processing without separate logging mechanisms.

2. Technical Field:

This invention relates to blockchain data recording, distributed ledger execution efficiency, and automated agent co-ordination, and more particularly to methods for atomic event capture and propagation in memory-driven systems.

3. Background:

Conventional decentralized systems often use multiple logging and state-storage mechanisms—on-chain state changes, off-chain logs, event queues, and disparate proof objects—to record and propagate system events. This leads to storage bloat, synchronization challenges, and complexity in integrating AI modules or downstream processors. There is a need for a unified, lightweight paradigm that captures each action atomically and provides a single source of truth for all participants and modules.

4. Summary:

Defining each logical action with explicit input and output schemas.

Generating a cryptographic attestation binding action inputs to outputs.

Minting a single on-chain token or transaction (one-action-one-mint) encapsulating the attestation and action identifier.

Configuring downstream modules, agents, and smart contracts to consume the minted transaction as their sole proof object.

Reconstructing full system state or driving coordinated behaviors by replaying or subscribing to the one-action-one-mint events.

5. Detailed Description:

At the core of the paradigm, each system event—whether it is a memory log entry, a proof-of-memory generation, a vault action, or an AI reflection—is formalized as an 'action' with defined data schemas for inputs and outputs. Upon event occurrence, a cryptographic attestation is computed (e.g., hashing or signing the concatenation of input and output data). A single mint transaction is then executed on-chain, embedding the attestation, action identifier, timestamp, and any metadata. This atomic mint serves as the definitive record of

the action, eliminating the need for separate state writes or logs. Modules across the network—such as autonomous agents, VM inference engines, and flow controllers—subscribe to these mint events to trigger subsequent processing, ensuring all components refer to the same immutable proof object.

6. Method Flow:

Step 1: Action Definition – Specify action types with input and output data schemas and unique identifiers.

Step 2: Attestation Generation – Upon action execution, compute a cryptographic attestation of inputs and outputs.

Step 3: Atomic Mint – Mint a single on-chain transaction embedding the attestation, action identifier, and relevant metadata.

Step 4: Downstream Integration – Enable smart contracts, agents, and analytics modules to subscribe to minted events as the sole source of truth.

Step 5: State Reconstruction – Optionally replay minted events through a stateless engine to reconstruct or verify full system state.

7. Narrative Worked Example:

Consider a vault credit issuance action: a participant requests 100 units of credit. The system defines the action schema (user ID, amount, timestamp). When authorized, the engine computes an attestation hash of ('userID|100|timestamp|previousBalance'), then mints a single transaction 'MintCredit' embedding the hash and action details. All downstream modules—ledger update, risk assessment agent, and dashboard— listen for 'MintCredit' events and update their state accordingly, without separate logs or state calls.

8. Algorithmic Worked Example:

Pseudocode:

```
def process_action(action_type, input_data):
    output_data = execute_logic(action_type, input_data)
    attestation = hash(input_data || output_data || action_type)
    mint_transaction('ActionMint', {
        'type': action_type,
        'attestation': attestation,
        'inputs': input_data_metadata,
        'outputs': output_data_metadata
    })
    notify_subscribers('ActionMint')
```

9. Potential Embodiments:

Using digital signatures or multi-signature schemes for attestation to support federated action approvals.

Batching multiple action mints into a single aggregated transaction for high-throughput environments.

Embedding Merkle proofs within the mint to represent nested sub-actions or complex workflows.

Integration with zero-knowledge proofs to attest action correctness and privacy.

Application in both PoS and PoW systems, and as a core module in the main application binary.

10. Implementation Notes:

Each mint transaction follows the one-action-one-mint rule and is recorded immutably on-chain. Action schemas and attestation logic reside in protocol modules or VM contexts. No additional on-chain state is required beyond the minted events.

11. Claims:

1. A method for recording system events in a decentralized memory-driven ledger, comprising:

a. defining, by a processor, a plurality of action types, each with specified input and output schemas and identifiers;

b. generating, by the processor, a cryptographic attestation binding the inputs and outputs of an action execution;

c. minting, by the processor, a single on-chain transaction encapsulating the attestation, action identifier, and metadata;

d. subscribing, by downstream modules, to the minted transaction as the exclusive proof object for triggering subsequent processing;

2. The method of claim 1, further comprising reconstructing system state by replaying the sequence of minted transactions through a stateless engine.

3. The method of claim 1, wherein the cryptographic attestation is a hash of the concatenated inputs and outputs.

4. The method of claim 1, wherein multiple action mints are batched into aggregated transactions for efficiency.

5. The method of claim 1, wherein zero-knowledge proofs are used to attest action correctness without revealing underlying data.