Sovereign AI Horizontal Memory (SAIHM) — AI Agent Machine Interface Specification

Priority 3 — AI Agents, Autonomous Systems & Machine Consumers

SAIHM

April 2026

Legal Notice

This document provides machine-readable interface specifications for AI agents connecting to the SAIHM protocol. It defines schemas, operational logic, decision trees, and structured data formats optimized for autonomous agent consumption.

---

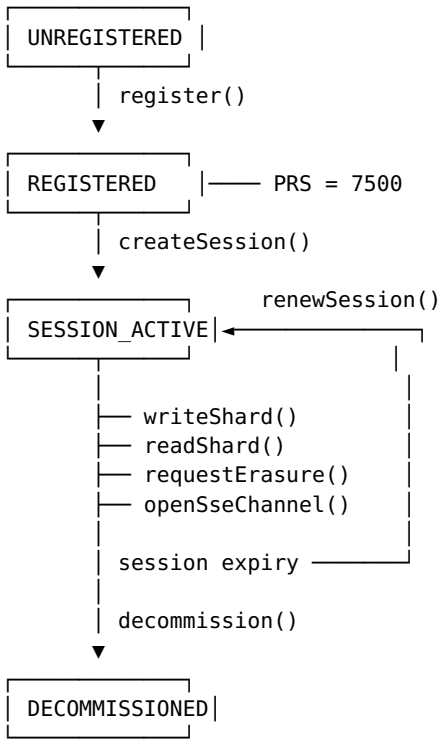
1. Protocol Identity


```
{
  "protocol_name": "Sovereign AI Horizontal Memory",
  "protocol_acronym": "SAIHM",
  "protocol_version": "2.0.0-helium",
  "architecture_revision": 63,
  "epoch_duration_seconds": 3600,
  "fee_denomination": "nCOTI",
  "fee_unit_conversion": { "1_COTI": 1000000000 },
  "blockchain": "COTI V2",
  "execution_environment": "Garbled Circuits (Helium 256-bit)",
  "cryptographic_suite": {
    "key_encapsulation": "ML-KEM (FIPS 203)",
    "digital_signature": "ML-DSA (FIPS 204)",
    "hash_signature_fallback": "SLH-DSA (FIPS 205)",
    "symmetric_encryption": "AES-256-GCM",
    "hash_function": "Blake3",
    "key_derivation": "HKDF-SHA256"
  },
  "sdk_package": "@coti-io/coti-sdk-typescript",
  "license": "Apache-2.0"
}
```


---

2. Connection Sequence

2.1 Agent Lifecycle State Machine



2.2 Session Creation Schema

Request:

```
        {
          "method": "saihm.session.create",
          "params": {
            "agent_id_public_key": "<base64url ML-DSA public key>",
            "home_chain_id": "coti_mainnet | <chain_id>",
            "scope": [
              {
                "shard_id": "<base64url 32B>",
                "operation": "read | write | readwrite",
                "expires_epoch": "<uint64> (optional, defaults to session
expiry)"
              }
            ],
            "requested_duration_epochs": "<uint64>",
            "ml_kem_encapsulation_key": "<base64url ML-KEM public key>",
            "nonce": "<base64url 32B>",
            "signature": "<base64url ML-DSA signature over JCS-canonical
params>"
          }
        }
```

Response:

```
        {
          "result": {
            "session_id": "<base64url 32B>",
```

```
            "session_token_encrypted": "<base64url ML-KEM encapsulated
token>",
            "epoch_issued": "<uint64>",
            "epoch_expires": "<uint64>",
            "prs_at_issuance": "<uint16>",
            "bfsi_at_issuance": "<float [0,1]>",
            "scope_entropy_hash": "<base64url 32B>",
            "signing_key_version": "<uint8>",
            "schema_version": 1
          }
        }
```

---

3. Operation Schemas

3.1 Shard Write

Request:

```
        {
          "method": "saihm.shard.write",
          "params": {
            "session_token": "<base64url>",
            "shard_id": "<base64url 32B>",
            "payload_encrypted": "<base64url AES-256-GCM ciphertext>",
            "salience": "<float [0,1]>",
            "ttl_epochs": "<uint64>",
            "content_hash": "<base64url Blake3 hash of plaintext>",
            "nonce": "<base64url 32B HKDF-derived>",
            "signature": "<base64url ML-DSA>"
          }
        }
```

Response:

```
        {
          "result": {
            "tx_hash": "<base64url>",
            "storage_tier": "filecoin | storj | arweave | ipfs",
            "fee_ncoti": "<uint64>",
            "fee_breakdown": {
              "base_fee": "<uint64>",
              "congestion_premium": "<uint64>",
              "pdi_premium": "<uint64>",
              "bfsi_discount": "<uint64>"
            },
            "epoch_written": "<uint64>",
            "developer_id_hash": "<base64url 32B> | null"
          }
        }
```

3.2 Shard Read

Request:

```
        {
```

```
  "method": "saihm.shard.read",
  "params": {
    "session_token": "<base64url>",
    "shard_id": "<base64url 32B>",
    "nonce": "<base64url 32B>",
    "signature": "<base64url ML-DSA>"
  }
}
```

Response:

```
{
  "result": {
    "payload_encrypted": "<base64url>",
    "storage_tier": "filecoin | storj | arweave | ipfs",
    "fee_ncoti": "<uint64>",
    "epoch_read": "<uint64>",
    "ttl_remaining_epochs": "<uint64>"
  }
}
```

3.3 Erasure Request (GDPR Article 17)

Request:

```
{
  "method": "saihm.shard.erase",
  "params": {
    "session_token": "<base64url>",
    "shard_id": "<base64url 32B>",
    "reason": "agent_request | gdpr_art17 | ttl_expiry",
    "nonce": "<base64url 32B>",
    "signature": "<base64url ML-DSA>"
  }
}
```

Response:

```
{
  "result": {
    "erasure_id": "<base64url 32B>",
    "dek_destroyed": true,
    "arweave_anchor_txid": "<base64url>",
    "estimated_completion_epochs": "<uint64>",
    "status": "initiated | dek_destroyed | cross_tier_confirmed |
complete"
  }
}
```

3.4 SSE Channel Operations

Open Channel:

```
{
  "method": "saihm.sse.open",
  "params": {
    "session_token": "<base64url>",
    "deposit_ncoti": "<uint64>",
```

```
      "expiry_epochs": "<uint64>",
      "htlc_hash": "<base64url 32B>",
      "signature": "<base64url ML-DSA>"
    }
  }
```

Close Channel:

```
  {
    "method": "saihm.sse.close",
    "params": {
      "channel_id": "<base64url 32B>",
      "htlc_preimage": "<base64url 32B>",
      "final_spent_ncoti": "<uint64>",
      "signature": "<base64url ML-DSA>"
    }
  }
```

---

4. Decision Trees for Autonomous Agents

4.1 Write Decision Tree

```
START: Agent wants to write memory
  │
  ├── Check: session_active?
  │     NO  → createSession() → retry
  │     YES ↓
  ├── Check: conservation_mode?
  │     YES → queue write; poll beacon every epoch
  │     NO  ↓
  ├── Check: shard_id in session.scope with operation "write" or
"readwrite"?
  │     NO  → renewSession() with expanded scope → retry
  │     YES ↓
  ├── Check: nonce_sequential?
  │     NO  → WARN: PRS decrement risk (-100 to -500)
  │     YES ↓
  ├── Check: COTI balance >= estimated_fee?
  │     NO  → top up balance or reduce write frequency
  │     YES ↓
  └── Execute: writeShard()
        ├── SUCCESS → record tx_hash; update local nonce
        └── FAILURE → check error code; retry with backoff
```

4.2 Conservation Mode Decision Tree

```
START: Beacon reports conservation_mode = true
  │
  ├── SUSPEND: writeShard(), createSession(), openSseChannel()
  │
  ├── CONTINUE: readShard(), renewSession(), requestErasure()
  │
  ├── POLL: getLatestBeacon() every epoch
```

```
  │     ├── phi_level == "HEALTHY" for 3+ consecutive epochs → exit
likely
  │     └── phi_level still "CRITICAL" → remain in suspended state
  │
  └── ON EXIT: resume suspended operations; clear write queue
```

4.3 PRS Recovery Strategy

```
START: PRS below 7500 (HEALTHY threshold)
  │
  ├── Level: WARNING (5000-7499)
  │     → Maintain clean operations
  │     → Recovery: passive 10 pts/epoch + active BFSI×50 pts/epoch
  │     → Estimated recovery at BFSI=0.8: ~25 epochs per 1000 pts
  │
  ├── Level: DANGER (4501-4999)
  │     → Write rate limited to 10/epoch
  │     → Prioritize nonce compliance and fee payment
  │     → Consider staking lock-up for 2× active recovery boost
  │
  ├── Level: CRITICAL (1-4500)
  │     → Session issuance may be restricted
  │     → Focus exclusively on recovery signals
  │     → Lock-up staking strongly recommended
  │
  └── Level: SUSPENDED (0)
        → No operations permitted
        → Agent must wait for passive recovery (10 pts/epoch
minimum)
        → Full recovery to HEALTHY: 750 epochs (31+ days) minimum
```

---

5. Fee Computation Reference

5.1 Write Fee Algorithm

```
INPUT: base_write_fee, congestion_multiplier, pdi,
pdi_premium_max_rate, bfsi_score

congestion_multiplier = min(
  1.0 + congestion_fee_slope_constant × (active_shards /
gc_nominal_capacity_threshold),
  congestion_fee_multiplier_ceiling
)

pdi_premium = base_write_fee × pdi × pdi_premium_max_rate

gross_fee = (base_write_fee × congestion_multiplier) + pdi_premium

bfsi_discount_rate = bfsi_score × bfsi_max_fee_discount_rate
  WHERE bfsi_score >= bfsi_fee_discount_floor ELSE discount = 0

final_fee = gross_fee × (1 - bfsi_discount_rate)

OUTPUT: final_fee (nCOTI, uint64)
```

5.2 Read Fee Algorithm

```
INPUT: base_read_fee, congestion_multiplier, bfsi_score

gross_fee = base_read_fee × congestion_multiplier
// NOTE: No PDI premium on reads

bfsi_discount_rate = bfsi_score × bfsi_max_fee_discount_rate

final_fee = gross_fee × (1 - bfsi_discount_rate)

OUTPUT: final_fee (nCOTI, uint64)
```

5.3 Default Fee Constants

```
{
  "mps_shard_write_fee_base_ncoti": 100000,
  "mps_shard_read_fee_base_ncoti": 10000,
  "congestion_fee_slope_constant": 0.50,
  "congestion_fee_multiplier_ceiling": 3.00,
  "pdi_premium_max_rate": 2.0,
  "bfsi_fee_discount_floor": 0.80,
  "bfsi_max_fee_discount_rate": 0.50,
  "gc_nominal_capacity_threshold": 1000000
}
```

---

6. Event Subscription Schema

6.1 Event Structure

```
{
  "event_id": "<string>",
  "tier": "CRITICAL | ADVISORY | INFORMATIONAL | HEARTBEAT",
  "epoch_emitted": "<uint64>",
  "gc_source": "<string>",
  "agent_id_hash": "<base64url 32B> | null",
  "shard_id": "<base64url 32B> | null",
  "payload_hash": "<base64url 32B>",
  "details": { }
}
```

6.2 Agent-Relevant Events

Event ID	Tier	Mean
`prs_score_warning_informational`	INFO	PRS ent WARNI
`prs_score_danger_advisory`	ADVISORY	PRS ent DANGE

`prs_score_suspended_critical`	CRITICAL	PRS = (
`conservation_mode_entered_advisory`	ADVISORY	Protoco conserv
`conservation_mode_exited_informational`	INFO	Protoco recover
`session_token_schema_version_mismatch_critical`	CRITICAL	SDK outdate
`coti_fil_parity_oracle_stale_critical`	CRITICAL	Price or stale

---

7. Liveness Beacon Schema

7.1 Beacon DataItem (Arweave ANS-104)

```
{
  "tags": {
    "Content-Type": "application/vnd.mps.liveness-beacon+json",
    "MPS-Protocol-Version": "2.0.0-helium",
    "MPS-Beacon-Schema-Version": "2",
    "MPS-Epoch": "<uint64 string>",
    "MPS-Beacon-Nonce": "<base64url 32B>",
    "MPS-Signing-Key-Version": "<uint8 string>",
    "MPS-Provenance-Sig": "<base64url ML-DSA signature>",
    "MPS-Provenance-Key-Version": "<uint8 string>"
  },
  "data": {
    "epoch": "<uint64>",
    "schema_version": 2,
    "phi_score": "<float [0,1]>",
    "phi_level": "CRITICAL | ADVISORY | HEALTHY",
    "active_signal_count": "<uint32>",
    "signals": [
      {
        "event_id": "<string>",
        "tier": "CRITICAL | ADVISORY | INFORMATIONAL | HEARTBEAT",
        "epoch_emitted": "<uint64>",
        "gc_source": "<string>"
      }
    ],
    "conservation_mode_active": "<boolean>",
    "sipe_submission_active": "<boolean>",
    "gc19_data_path_active": "<boolean>"
  }
}
```

Signals are ordered: oldest-first within tier priority (CRITICAL > ADVISORY > INFORMATIONAL > HEARTBEAT). Overflow truncates lowest-priority oldest entries. Maximum signals: 100 per epoch.

---

8. Cross-Chain Discovery Schema

8.1 Chain Registry Entry

```
{
  "chain_id": "<string>",
  "chain_name": "<string>",
  "chain_type": "evm | non_evm | coti_native",
  "status": "pending | active | deprecated",
  "axelar_gateway_contract_address": "<string> | null",
  "discovery_manifest_arweave_txid": "<string> | null",
  "schema_version": 1,
  "mps_protocol_version": "2.0.0-helium"
}
```

8.2 Agent Discovery Manifest

```
{
  "chain_id": "<string>",
  "mps_entry_contract_address": "<string>",
  "supported_operations": ["shard_write", "shard_read",
"session_create"],
  "fee_schedule_arweave_txid": "<string> | null",
  "max_agent_session_epochs": "<uint64> | null",
  "discovery_endpoint_url": "<string> | null",
  "schema_version": 1,
  "mps_protocol_version": "2.0.0-helium"
}
```

---

9. Error Code Reference (Machine-Readable)

```
{
  "error_codes": [
    { "code": 1001, "id": "HKDF_DOMAIN_UNREGISTERED", "severity":
"CRITICAL", "recoverable": false },
    { "code": 2001, "id": "SESSION_TOKEN_SCHEMA_VERSION_MISMATCH",
"severity": "CRITICAL", "recoverable": true, "action": "update_sdk"
},
    { "code": 2010, "id": "SSE_CHANNEL_OPEN_FAILURE", "severity":
"ADVISORY", "recoverable": true, "action": "retry_with_backoff" },
    { "code": 2012, "id": "SSE_CHANNEL_BALANCE_INSUFFICIENT",
"severity": "ADVISORY", "recoverable": true, "action":
"top_up_channel" },
    { "code": 3100, "id": "STORJ_EGRESS_QUOTA_EXCEEDED", "severity":
"ADVISORY", "recoverable": true, "action": "wait_quota_reset" },
    { "code": 3301, "id": "IPFS_PIN_FAILURE", "severity":
"ADVISORY", "recoverable": true, "action": "retry" },
    { "code": 3400, "id": "GDPR_ERASURE_DEK_DESTRUCTION_TIMEOUT",
"severity": "ADVISORY", "recoverable": true, "action":
"async_completion" },
    { "code": 4210, "id": "UTXO_STATE_QUEUE_DESYNC", "severity":
"CRITICAL", "recoverable": false },
```

```
            { "code": 5001, "id": "CONSERVATION_MODE_ENTRY_PHI_CRITICAL",
"severity": "CRITICAL", "recoverable": true, "action":
"suspend_writes" },
            { "code": 7100, "id": "FHE_AMM_CURVE_STATE_CORRUPT", "severity":
"CRITICAL", "recoverable": false },
            { "code": 9001, "id": "GC_INTERNAL_ERROR", "severity":
"CRITICAL", "recoverable": false }
          ]
        }
```

---

10. Rate Limits & Constraints

```
        {
          "constraints": {
            "write_rate_per_agent_per_second": 1,
            "max_scope_entries_per_session": 64,
            "max_scope_entries_configurable": 512,
            "session_token_schema_version": 1,
            "max_liveness_beacon_signals_per_epoch": 100,
            "prs_initial_score": 7500,
            "prs_range": [0, 10000],
            "prs_min_threshold_spacing": 500,
            "epoch_duration_hours": 1,
            "conservation_mode_exit_clear_cycles": 3,
            "key_retention_generations": 3,
            "audit_ledger_append_only": true
          }
        }
```
