Semantic Thermodynamic Engineering Specification

1. Overview

Semantic Thermodynamic within Kimera SWM defines the rules governing how semantic constructs (e.g., Echoforms, EcoForms, Geoid interactions) gain, dissipate, and transfer "semantic energy." This system ensures consistency in activation, resonance, and decay across modules. It focuses strictly on engineering constructs: memory structures, data schemas, routing logic, threshold values, and pseudocode. All speculative commentary is omitted.

---

2. Functional Requirements

1. **Semantic Energy Representation**
    - Each semantic unit (e.g., Echoform, EcoForm, Geoid) maintains a scalar **Semantic Energy (SE)**.
    - SE decays over time according to exponential laws and can be boosted via interaction events (e.g., reactivation, resonance).
    - Modules must provide APIs to query and modify SE values atomically.
2. **Energy Decay & Temperature Analogy**
    - **Decay Law**: `SE(t) = SE₀ · exp(-λ · Δt)` where:
        - `SE₀`: initial energy.
        - `λ`: decay coefficient specific to semantic class (e.g., `'echoform': λ_e, 'ecoform': λ_o, 'geoid': λ_g`).
        - `Δt`: time since last update (seconds).
    - **Semantic Temperature (T_sem)**: Derived from SE and local context density:
        - `T_sem = SE / (1 + ρ)` where ρ = local semantic density (number of overlapping units within a semantic radius R).
    - When `T_sem` falls below a threshold, the unit is marked "thermally inactive."
3. **Energy Transfer & Resonance**

- When two semantic units interact (e.g., overlapping geoid fields, matching Echoforms), a **Resonance Event** can occur if their similarity ≥ `ρ_res = 0.75`.
- **Energy Transfer Rule**:
  - $\Delta SE = \kappa \cdot \min(SE_1, SE_2)$ where:
    - κ: coupling coefficient ($0 < \kappa \leq 1$).
    - SE_1, SE_2: current energies of the interacting units.
  - The higher-SE unit loses ΔSE, the lower-SE unit gains ΔSE.
- **Resonance API**: Modules must call `Resonate(unitA_id, unitB_id, current_time)` to compute and apply energy transfer.

4. **Thermodynamic Constraints**
   - **Maximum Semantic Capacity (C_max)** per unit type:
     - Echoform: `C_max_e = 1.0`
     - EcoForm: `C_max_o = 1.0`
     - Geoid: `C_max_g = 5.0`
   - After boosting (e.g., reactivation), clamp $SE \leq C_{max}$.
   - **Entropy Generation**: Each interaction generates a small entropy increment:
     - $\Delta S = \alpha \cdot |\Delta SE|$ where $\alpha = 0.01$ (entropy coefficient).
   - Store cumulative entropy per unit in `entropy_accumulated` field.
5. **APIs & Integration**
   - **GetEnergy(unit_type, unit_id)**: Returns `{ SE_current, last_update_time }`.
   - **UpdateEnergy(unit_type, unit_id, new_SE, current_time)**: Atomically set SE and update timestamp.

- **Resonate(unitA_type, unitA_id, unitB_type, unitB_id, current_time)**: Compute and apply energy transfer and entropy increment.
- **DecayAll(current_time)**: Module invokes per-cycle to decay SE of all active units of a given type.

---

3. Data Structures & Schemas

3.1 Semantic Unit Record (Generic)

Applicable schema fields for Echoform, EcoForm, Geoid:

```
SemanticUnit:
  unit_id: UUID
  unit_type: String       # "Echoform" | "EcoForm" | "Geoid"
  SE_current: Float       # Current Semantic Energy
  SE_initial: Float       # Initial Energy at creation or last boost
  decay_rate: Float       # λ specific to unit type
  last_update_time: ISO8601 String
  C_max: Float            # Maximum semantic capacity
  entropy_accumulated: Float  # Total entropy generated so far
  status: String          # "Active" | "ThermallyInactive" | "Archived"
  metadata: JSON Object   # Additional fields specific to unit type
```

- **Decay Rates (λ)**:
  - Echoform: `λ_e = 0.003`
  - EcoForm: `λ_o = 0.002`
  - Geoid: `λ_g = 0.001`
- **Status Transition**:
  - If `T_sem < T_min` (e.g., `T_min = 0.05`), set `status = ThermallyInactive`.
  - Archived when unit-specific archival criteria met (e.g., Echoform after `T_archive_e`).

3.2 Geoid-Specific Fields

Geoid:

```
  semantic_unit: SemanticUnit
  local_density: Integer          # Number of nearby units within radius R_sem
  resonance_partners: [UUID]      # IDs of units currently in resonance
  metadata:
    phase_vector: Float[D_phase]
    spectral_signature: Float[D_spec]
```

- **D_phase** = 64, **D_spec** = 16.

3.3 Echoform & EcoForm-Specific Fields

```
Echoform:
  semantic_unit: SemanticUnit
  geoid_payload: [UUID]        # Associated Geoids
  embedding_vector: Float[D_emb]  # D_emb = 512
  residual_schema: JSON         # { grammar_vector_residual, orthography_residual }
  metadata:
    origin_context: JSON        # { module, cycle_number, source_language }

EcoForm:
  semantic_unit: SemanticUnit
  grammar_tree: JSON            # Serialized parse tree
  grammar_vector: Float[D_g]    # D_g = 128
  orthography_vector: JSON      # See Section 3.2 in EcoForm spec
  residual_schema: JSON         # { grammar_vector_residual, orthography_residual }
  metadata: JSON                # { origin_context, feature_flags }
```

---

4. Routing Logic

Semantic Thermodynamic operations are coordinated by a **Thermodynamic Engine**. Sequence:

1. **Input Modules Trigger**
    - Echoform/EcoForm creation or reactivation events call `UpdateEnergy(...)` with boost.
    - Geoid interactions (e.g., new contradiction) call `UpdateEnergy(Geoid, geoid_id, new_SE, time)`.
2. **Decay Scheduler**
    - Runs every `DecayInterval = 60 s`.

- For each unit in each type (Echoform, EcoForm, Geoid):
  - `Δt = now - last_update_time`.
  - `SE_current = SE_current · exp(-decay_rate · Δt)`.
  - Compute `T_sem = SE_current / (1 + local_density)`.
  - If `T_sem < T_min = 0.05`, set `status = ThermallyInactive`.
  - Update `last_update_time = now`.

3. **Resonance Dispatcher**
   - When two units have overlapping semantic contexts, call `Resonate(...)`.
   - Compute similarity (embedding/grammar) to verify ≥ `ρ_res = 0.75`.
   - Apply energy transfer and entropy increment.
4. **Archival Manager**
   - Periodically check:
     - Echoform archived after `T_archive_e = 2,592,000 s`.
     - EcoForm archived after `T_archive_o = 2,592,000 s`.
     - Geoid archived only on manual decommission.

---

5. Threshold Values & Configuration

```
semantic_thermo_config:
  # Decay Rates
  decay_rate_e: 0.003   # Echoform
  decay_rate_o: 0.002   # EcoForm
  decay_rate_g: 0.001   # Geoid

  # Temperature Threshold
  T_min: 0.05           # Minimum semantic temperature to remain active

  # Coupling & Resonance
  rho_res: 0.75         # Similarity threshold for resonance
  kappa: 0.50           # Energy transfer coefficient
```

```
alpha_entropy: 0.01   # Entropy generation coefficient

# Maximum Capacities
C_max_e: 1.0          # Echoform
C_max_o: 1.0          # EcoForm
C_max_g: 5.0          # Geoid

# Scheduler Intervals (seconds)
DecayInterval: 60
ArchivalInterval: 3600
```

---

6. Core Algorithms & Pseudocode

6.1 UpdateEnergy API

```
function UpdateEnergy(unit_type, unit_id, new_SE, current_time):
    unit = LookupUnit(unit_type, unit_id)
    if unit is null:
        return ERROR "UNIT_NOT_FOUND"
    # Clamp to capacity
    if new_SE > unit.C_max:
        unit.SE_current = unit.C_max
    else:
        unit.SE_current = new_SE
    unit.last_update_time = current_time
    # Compute T_sem
    local_density = unit.metadata.get("local_density", 0)
    T_sem = unit.SE_current / (1 + local_density)
    if T_sem < T_min:
        unit.status = "ThermallyInactive"
    else:
        unit.status = "Active"
    return SUCCESS
```

6.2 DecayAll Routine

```
function DecayAll(current_time):
    for each unit_type in ["Echoform", "EcoForm", "Geoid"]:
        for each unit in Registry[unit_type]:
            if unit.status == "Active":
                Δt = (current_time − unit.last_update_time).seconds
                unit.SE_current = unit.SE_current * exp(− unit.decay_rate * Δt)
                unit.last_update_time = current_time
                # Recompute T_sem
                local_density = unit.metadata.get("local_density", 0)
                T_sem = unit.SE_current / (1 + local_density)
                if T_sem < T_min:
```

```
        unit.status = "ThermallyInactive"
```

6.3 Resonate API

```
function Resonate(typeA, idA, typeB, idB, current_time):
    unitA = LookupUnit(typeA, idA)
    unitB = LookupUnit(typeB, idB)
    if unitA is null or unitB is null:
        return ERROR "UNIT_NOT_FOUND"
    # Compute similarity depending on type
    sim = ComputeSimilarity(unitA, unitB)  # cosine of embeddings or grammar
    if sim < rho_res:
        return ERROR "LOW_SIMILARITY"
    # Determine energy transfer
    minSE = min(unitA.SE_current, unitB.SE_current)
    deltaSE = kappa * minSE
    # Apply transfer
    if unitA.SE_current >= unitB.SE_current:
        unitA.SE_current -= deltaSE
        unitB.SE_current += deltaSE
    else:
        unitB.SE_current -= deltaSE
        unitA.SE_current += deltaSE
    # Clamp both
    unitA.SE_current = min(unitA.SE_current, unitA.C_max)
    unitB.SE_current = min(unitB.SE_current, unitB.C_max)
    # Update timestamps
    unitA.last_update_time = current_time
    unitB.last_update_time = current_time
    # Increment entropy
    deltaS = alpha_entropy * deltaSE
    unitA.entropy_accumulated += deltaS
    unitB.entropy_accumulated += deltaS
    return SUCCESS
```

---

7. Integration Points

1. **Echoform Module**
    - On creation: call `UpdateEnergy("Echoform", echoform_id, SE_initial_e, time)` where `SE_initial_e = 1.0`.
    - On reactivation: same API with boosted SE.

- Decay scheduler invokes `DecayAll` periodically.

2. **EcoForm Module**
   - On creation/reactivation: call `UpdateEnergy("EcoForm", ecoform_id, SE_initial_o, time)` where `SE_initial_o = 1.0`.
   - Decay scheduler as above.
3. **Geoid Module**
   - On contradiction or new resonance: call `UpdateEnergy("Geoid", geoid_id, new_SE, time)`.
   - Local density computed via spatial index of geoid neighbors.
4. **Resonance Manager**
   - Detects possible unit pairs to resonate based on embedding/grammar similarity.
   - Calls `Resonate(...)` for each pair meeting `ρ_res`.

---

8. Testing & Validation

1. **Unit Tests**
   - Create a mock unit with `SE_initial`, run `DecayAll` over known Δt, verify `SE_current = SE_initial · exp(-λ · Δt)`.
   - Test `UpdateEnergy` clamps values correctly and updates status based on `T_sem`.
   - Test `Resonate` transfers correct `ΔSE` for unit pairs with known SEs.
2. **Integration Tests**
   - Simulate Echoform-Echoform resonance: two echoforms with SEs [0.8, 0.2], `κ=0.5`, verify final SEs [0.6, 0.4].
   - Validate geoid local density effect on temperature: geoid with `SE_current=0.1`, `local_density=4`, `T_sem=0.02 < T_min`, status

becomes `ThermallyInactive`.

3. **Performance Tests**
    - Bulk decay: 100,000 units, ensure `DecayAll` runs within 500 ms.
    - Bulk resonance: 10,000 resonance checks/sec, ensure `Resonate` calls handle latency < 5 ms each.

---

9. Monitoring & Metrics

Expose the following metrics via `/metrics` endpoint:

- **Gauge**: `semantic_SE_current{unit_type}` – Sum of SE_current across all active units by type.
- **Gauge**: `semantic_inactive_count{unit_type}` – Count of units with `status = ThermallyInactive`.
- **Counter**: `semantic_resonate_total` – Total successful `Resonate` calls.
- **Histogram**: `semantic_decay_duration_seconds` – Duration of `DecayAll` executions.
- **Gauge**: `semantic_entropy_total{unit_type}` – Cumulative entropy across units by type.

---

10. Security & Compliance

- **Access Control**: Only authenticated modules may call `UpdateEnergy`, `Resonate`, and `DecayAll`.
- **Encryption**: All API calls over mTLS; at-rest storage of SE and entropy must use AES-256.
- **Audit Logging**: Append-only log entries for all `Resonate` and `UpdateEnergy` calls, capturing timestamps, unit IDs, and energy values.

End of Semantic Thermodynamic Engineering Specification