EcoForm Engineering Specification

1. Overview

EcoForm is the non-linear grammar and orthography subsystem within Kimera SWM. Its primary purpose is to represent, store, and manipulate structured grammatical constructs (e.g., nested syntactic patterns, non-linear parse trees) alongside orthographic transformations (e.g., script normalization, variant mappings). EcoForm serves as the foundation for:

- **Grammar Encoding: Capturing hierarchical, non-linear syntactic patterns in a flexible data structure.**
- **Orthographic Mapping: Managing script-level transformations (e.g., ligatures, diacritics, Unicode normalization) and linking them to grammatical units.**
- **Pattern Matching & Retrieval: Querying stored grammatical/orthographic constructs based on similarity or structural criteria.**
- **Integration with SWM: Exposing structured outputs to downstream SWM modules (e.g., Echoform, Geoid alignment) via defined APIs.**

EcoForm operates on discrete input events (token streams, symbol sequences), producing or updating "EcoForm units" that encapsulate both grammar and orthography metadata. Each unit retains a decaying activation profile, enabling later reactivation or merging based on new inputs.

---

2. Functional Requirements

1. **Creation & Initialization**
    - **Trigger Conditions:**
        1. **Incoming text or symbol stream submitted to the EcoForm Parser.**
        2. **Receipt of a structured linguistic event from upstream modules (e.g., token embeddings from Preprocessing).**
    - **Data Captured:**
        1. **EcoForm ID: Globally unique UUID.**

2. **Origin Context:** `{ module: String, cycle_number: Int, source_language: String }`.
3. **Grammar Payload:** Representation of non-linear parse tree (see Section 3.1).
4. **Orthography Vector:** Numeric or symbolic vector capturing script transformations (see Section 3.2).
5. **Activation Strength (AS_0):** Initial activation scalar ($0 < AS_0 \leq 1.0$).
6. **Decay Parameter (δ):** Coefficient controlling exponential decay of AS.
7. **Timestamp:** ISO 8601 creation time.

2. **Decay & Evaporation**
   - **Decay Law:** $AS(t) = AS_0 \cdot \exp(-\delta \cdot \Delta t)$ where Δt = current_time − creation_time (in seconds).
   - **Evaporation Threshold:** $\varepsilon_g = 0.05$ (units of activation). When $AS(t) < \varepsilon_g$, mark EcoForm unit as Evaporated and generate a Residual Schema (see Section 3.3).
   - **Archival:** After $T_{archive_g} = 2{,}592{,}000\ s$ (30 days), move Evaporated units to a cold-storage archive: retain only `Residual Schema` and minimal metadata.
3. **Query & Matching**
   - **Structural Query:** Given a target grammar pattern or orthography variant, return all Active or Evaporated EcoForm units whose Normalized Similarity Score (NSS) ≥ $\rho_g = 0.70$.
   - **Input:** `{ target_pattern: GrammarTree, orthography_pattern: Vector, max_age: Int (s) }`.
   - **Output:** List of `{ ecoform_id, AS_current, NSS }` sorted descending by NSS.
4. **Reactivation**
   - **Eligibility:** Only Evaporated units with $age \leq T_{reactivate_g} = 86{,}400\ s$ (24 h) and $NSS \geq \rho_g$.

- **Process:**

    1. **Boost Activation:** $AS_{new} = AS_{current} \cdot \alpha_g$ **where** $\alpha_g = 0.50$.
    2. **Merge Grammar Payloads: Blend stored non-linear trees (e.g., weighted union or structural overlay) with incoming pattern.**
    3. **Update Timestamp: Set** $last\_reactivation\_time = now$**, update status to Active.**

5. **Orthography Normalization & Transformation**

    - **Normalization Rules:**

        1. **NFC/NFD Unicode standard.**
        2. **Script-specific mappings (e.g., Arabic diacritic stripping, Latin ligature splitting).**

    - **Thresholds:**

        1. **Diacritic Sensitivity (**Δ_1**): If diacritic variance ≤ 0.02, treat as same underlying form; otherwise, tag as distinct.**
        2. **Ligature Variance (**Δ_2**): Similarity threshold for mapping ligatures to base form = 0.85.**

    - **Transformation API: Provide methods to convert between variant forms and canonical forms, preserving links to associated grammar payloads.**

6. **Memory Stitching & Merging**

    - **Input: Multiple Evaporated EcoForm IDs can be stitched into a Composite Grammar Unit.**

    - **Process:**

        1. **Extract Residual Schemas from each EcoForm.**
        2. **Compute Weighted Merge Tree: For each node in the grammar trees, combine children based on normalized weights** $w_i = AS_i \ / \ \Sigma \ AS$.
        3. **Generate new EcoForm Unit with** $AS_0 = \Sigma \ AS_i \cdot \beta_g$ **where** $\beta_g = 0.25$.

4. **Link new unit to all source EcoForm IDs as “stitch_source.”**

7. **APIs & Integration**
    - **CreateEcoForm: Accept raw input, parse into grammar/orthography constructs, store new EcoForm unit.**
    - **GetEcoFormStatus: Return `{ AS_current, age, status, creation_time, last_reactivation_time }`.**
    - **QueryEcoForms: Return list based on `target_pattern`, `orthography_pattern`, `max_age`.**
    - **ReactivateEcoForm: Input `{ ecoform_id, new_pattern, new_context }`. Perform eligibility check and reactivation.**
    - **StitchEcoForms: Input `{ source_ids: [UUID], target_language: String }`. Return new composite EcoForm ID.**

---

3. Memory Structures & Data Schemas

3.1 Grammar Payload Schema

Each EcoForm unit contains a Grammar Tree representing nested, non-linear syntactic constructs. The schema is defined by:

```
grammar_tree:
  node_id: UUID
  label: String          # e.g., "NP", "VP", "Det", "Noun"
  children: [ GrammarNode ]  # List of child nodes (possibly empty)
  features:              # Key–value pairs for node attributes
    pos_tag: String
    morphological_tags: [String]
    subscript_index: Int   # For non-linear references (e.g., cross-links)
```

- **Non-Linear References: Each node may include cross-links to other nodes via `subscript_index`, enabling DAGs rather than strict trees.**
- **Feature Vector: A fixed-length vector `GV` (Dimension `D_g = 128`) encoding syntactic features (one-hot or learned embeddings).**

3.2 Orthography Vector Schema

Orthographic representation is stored as a fixed-length vector capturing script and variant features:

```
orthography_vector:
  script_code: String       # e.g., "Latn", "Arab", "Cyrl"
  unicode_normal_form: String # "NFC", "NFD"
  diacritic_profile: Vector[D_o=32]  # Float array capturing diacritic presence weights
  ligature_profile: Vector[D_l=32]   # Float array for common ligature patterns
  variant_flags:            # Boolean flags or small ints indicating variant types
    has_cedilla: Boolean
```

has_breve: Boolean

is_hyphenated: Boolean

- **D_o = 32, D□ = 32 are fixed by configuration.**
- **Normalization State: Encoded as 2-bit value internally (0 = none, 1 = NFC, 2 = NFD, 3 = NFKC).**

3.3 EcoForm Registry Schema

All EcoForm units are stored in the EcoForm Registry (e.g., in-memory DB or document store). Each record has:

Field Name	Type	Description
`ecoform_id`	UUID (String)	Unique identifier.
`origin_context`	JSON Object	`{ module: String, cycle_number: Int, source_language: String }`.
`grammar_tree`	JSON Object	Serialized Grammar Payload (see Section 3.1).
`grammar_vector`	Float[D_g]	Dense vector encoding grammar features.
`orthography_vector`	JSON Object	Orthography Vector (see Section 3.2).
`initial_AS`	Float	Initial Activation Strength ($0 < AS_0 \leq 1.0$).

`AS_current`	Float	Current activation strength (updated each cycle).
`decay_rate`	Float	Exponential decay coefficient δ.
`status`	String	Enum { “Active”, “Evaporated”, “Archived” }.
`creation_time`	ISO 8601 String	Timestamp of creation.
`last_reactivation_time`	ISO 8601 String // null	Timestamp of last reactivation; null if never reactivated.
`evaporation_time`	ISO 8601 String // null	Timestamp when AS fell below ε_g; null if still Active.
`residual_schema`	JSON Object // null	`{ grammar_vector_residual: Float[D_r], orthography_residual: Vector[D_r2] }` captured on evaporation.
`metadata`	JSON Object	Arbitrary key–value pairs (e.g., confidence scores, audit_tags).

- D_g = 128 is the dimensionality of grammar feature vectors.
- Residual Dimensions:
  - `D_r = 64` (compressed grammar vector),
  - `D_r2 = 16` (compressed orthography vector).

3.3.1 Indexes & Partitioning

- **Primary Key:** `ecoform_id`.
- **Secondary Indexes:**
  - `(status, AS_current)` to retrieve Active units above a threshold.
  - `origin_context.source_language` for language-specific queries.
  - `grammar_vector` and/or precomputed LSH buckets for approximate similarity lookup.
  - `creation_time` / `evaporation_time` for age-based filtering.
- **Shard Strategy:** Consistent hashing on `ecoform_id` into N_g = 4 shards, each handling ~100,000 Active units.

---

4. Routing Logic

EcoForm processing is managed by a Routing Engine that directs incoming events to appropriate handlers. The following describes the routing flow:

1. **Input Ingestion**
   - **Sources:**
     - **Text Preprocessor: Streams of tokenized words, each tagged with a language code.**
     - **Symbolic Event Bus: Emitted grammar/orthography constructs from external modules (e.g., Lexical Analyzer).**
   - **Routing Rule: All inputs with `source_language` $\in$ `SWM_supported_languages` are forwarded to EcoForm Parser.**
2. **EcoForm Parser**
   - **Step 1: Orthography Normalizer**
     - **Apply Unicode Normalization Form C (NFC) by default.**

        - Strip or annotate diacritics if `diacritic_profile_variance ≤ Δ₁ = 0.02`.
        - Map ligatures to base characters if `ligature_similarity ≥ Δ₂ = 0.85`.
    - Step 2: Grammar Tree Builder
        - Invoke a non-linear parser (e.g., custom CYK extension) to produce a Grammar Tree.
        - Compute `grammar_vector` (D_g = 128) via a lookup embedding or computed features.
    - Step 3: EcoForm Creation
        - Compute `initial_AS` based on input confidence (e.g., upstream parser confidence score).
        - Set `decay_rate = δ_g` from configuration (default `δ_g = 0.003`).
        - Call CreateEcoForm API with assembled data (see Section 5.1).
3. Decay Scheduler
    - Runs every `EvaporationCheckInterval = 60 s`:
        - For each Active EcoForm unit:
            1. Compute `Δt = now - last_update_time`.
            2. Update `AS_current ← AS_current · exp(-δ_g · Δt)`.
            3. If `AS_current < ε_g (0.05)`:
                - Mark status = “Evaporated”.
                - Set `evaporation_time = now`.
                - Generate `residual_schema` via compress() (see Section 3.3).

4. If `now - creation_time ≥ T_archive_g (2,592,000 s)` and status = “Evaporated”:
    - Move unit to archive store (retain only `residual_schema`, `decay_rate`, `evaporation_time`, and `origin_context`).
    - Update status = “Archived”.

4. Query Dispatcher
    - Receives query requests (`target_pattern`, `orthography_pattern`, `max_age`).
    - Filters backend shards based on `status ∈ {Active, Evaporated}` and `age ≤ max_age`.

Computes Normalized Similarity Score (NSS) as:

NSS = w_1 · cosine(grammar_vector, target_grammar_vector)

+ w_2 · cosine(orthography_vector.embedding, target_orthography_vector)

- where $w_1 = 0.60$, $w_2 = 0.40$.
- Returns matches with `NSS ≥ ρ_g (0.70)`, sorted by NSS descending.

5. Reactivation Service
    - Listens for reactivation events requesting `ecoform_id` and `new_pattern`.
    - Verifies that unit’s status = “Evaporated” and `age ≤ T_reactivate_g`.
    - Calculates `NSS_reactivate` between stored `residual_schema` and `new_pattern`.
    - If `NSS_reactivate ≥ ρ_g`, perform reactivation (see Section 2.4). Otherwise, reject.

6. Stitching Orchestrator
    - Accepts lists of source EcoForm IDs.

- Validates that each source is Evaporated and `age ≤ T_archive_g`.
- Invokes StitchMemory routine (see Section 2.6) to produce a new composite EcoForm unit.

---

5. API Specification

EcoForm exposes a set of JSON-over-HTTP endpoints. Authentication is enforced via mTLS, and all requests require a valid service certificate authorized under SWM trust domain.

5.1 CreateEcoForm

POST /ecoform/create

- **Headers:**
  - `Content-Type: application/json`
  - **mTLS Client Certificate present**

Request Body:

```
{
 "origin_context": {
  "module": "String",
  "cycle_number": Integer,
  "source_language": "String"
 },
 "grammar_payload": {
  "node_id": "UUID",
  "label": "String",
  "children": [ /* recursive GrammarNode objects */ ],
  "features": {
```

```
    "pos_tag": "String",
    "morphological_tags": ["String", ...],
    "subscript_index": Integer
   }
  },
  "grammar_vector": [ Float, ..., Float ],        // length = D_g = 128
  "orthography_vector": {
   "script_code": "String",
   "unicode_normal_form": "String",
   "diacritic_profile": [ Float, ..., Float ],  // length = D_o = 32
   "ligature_profile": [ Float, ..., Float ],   // length = D_l = 32
   "variant_flags": {
    "has_cedilla": Boolean,
    "has_breve": Boolean,
    "is_hyphenated": Boolean
   }
  },
  "initial_AS": Float,       // 0 < initial_AS ≤ 1.0
  "decay_rate": Float,       // default δ_g = 0.003
  "creation_time": "ISO_8601 String"
}
```

-

Response Body (HTTP 200):

```
{
  "ecoform_id": "UUID",
  "status": "Active",
```

"initial_AS": Float

}

-
- **Error Codes:**
    - `400 Bad Request` **if required fields are missing or vectors have incorrect dimensions.**
    - `401 Unauthorized` **if mTLS certificate is invalid.**
    - `500 Internal Server Error` **on storage errors.**

5.2 GetEcoFormStatus

GET /ecoform/{ecoform_id}/status

- **Path Parameter:**
    - `ecoform_id`**: UUID string.**

Response Body (HTTP 200):

{

"ecoform_id": "UUID",

"status": "Active" | "Evaporated" | "Archived",

"AS_current": Float,

"age_seconds": Integer,

"creation_time": "ISO_8601 String",

"last_reactivation_time": "ISO_8601 String" | null,

"evaporation_time": "ISO_8601 String" | null

}

-

- **Error Codes:**
    - `404 Not Found` **if** `ecoform_id` **does not exist in Active or Archive.**
    - `500 Internal Server Error` **on lookup failures.**

5.3 QueryEcoForms

POST /ecoform/query

Request Body:

```
{
 "target_grammar_vector": [ Float, ..., Float ],     // D_g = 128
 "target_orthography_vector": {
   "script_code": "String",
   "unicode_normal_form": "String",
   "diacritic_profile": [ Float, ..., Float ],      // D_o = 32
   "ligature_profile": [ Float, ..., Float ],       // D_l = 32
   "variant_flags": {                           // optional
     "has_cedilla": Boolean,
     "has_breve": Boolean,
     "is_hyphenated": Boolean
   }
 },
 "max_age_seconds": Integer,    // e.g., 86400 for 24 h
 "min_NSS": Float               // default = 0.70
}
```

-

Response Body (HTTP 200):

```
{
 "matches": [
  {
   "ecoform_id": "UUID",
   "AS_current": Float,
   "NSS": Float
  },
  ...
 ]
}
```

-
- **Behavior:**
  - **Filter out units with `status == "Archived"` or `age_seconds > max_age_seconds`.**
  - **Compute `NSS = 0.60·cosine(grammar_vector, target_grammar_vector) + 0.40·cosine(orthography_embedding, orthography_embedding_target)`.**
  - **Only include matches where `NSS ≥ min_NSS`.**
- **Error Codes:**
  - **`400 Bad Request` if vectors are malformed or missing.**
  - **`500 Internal Server Error` on query timeouts or backend errors.**

5.4 ReactivateEcoForm

POST /ecoform/{ecoform_id}/reactivate

- **Path Parameter:** `ecoform_id`.

Request Body:

```
{
 "new_grammar_vector": [ Float, ..., Float ],  // D_g = 128
 "new_orthography_vector": { /* same schema as in CreateEcoForm */ },
 "required_language": "String"                 // must match origin_context.source_language
}
```

-

Response Body (HTTP 200) if successful:

```
{
 "ecoform_id": "UUID",
 "status": "Active",
 "AS_current": Float,
 "last_reactivation_time": "ISO_8601 String"
}
```

-
- **Behavior:**
    - Verify `status == "Evaporated"`.
    - Compute `age_seconds = now - creation_time`; fail if `age_seconds > T_reactivate_g (86400)`.
    - Compute `NSS_reactivate` against stored `residual_schema`. If `NSS_reactivate < ρ_g`, return `409 Conflict`.
    - Perform:
        - `AS_current ← AS_current · α_g (0.50)`.

        - Merge stored `grammar_tree` with new vector by structural overlay or weighted union.
        - Merge stored `orthography_vector` with new by component-wise max similarity.
        - Update `last_reactivation_time = now`.
        - Set `status = "Active"`.
    - Return updated status.
- Error Codes:
    - `404 Not Found` if `ecoform_id` does not exist.
    - `409 Conflict` if eligibility criteria not met (e.g., `status != "Evaporated"`, `age_seconds > 86400`, or `NSS_reactivate < 0.70`).
    - `400 Bad Request` for malformed vectors.
    - `500 Internal Server Error` on internal errors.

5.5 StitchEcoForms

POST /ecoform/stitch

Request Body:

{

"source_ids": ["UUID1", "UUID2", ...],

"target_language": "String"

}

-

Response Body (HTTP 200) on success:

{

"new_ecoform_id": "UUID",

"status": "Active",

"initial_AS": Float

}

-
- **Behavior:**
    - **For each `id ∈ source_ids`:**
        - **Verify existence and `status == "Evaporated"`.**
        - **Verify `age_seconds ≤ T_archive_g`; otherwise, cannot stitch.**
    - **For each source: retrieve `residual_schema`:**
        - **`residual_grammar_vector` (D_r = 64)**
        - **`residual_orthography_vector` (D_r2 = 16)**
        - **Retrieve `AS_current` (at evaporation).**
    - **Compute normalized weights $w_i = AS_i / \Sigma_j AS$.**

Compute Composite Grammar Vector:

$comp\_grammar\_vector[k] = \Sigma_i (w_i \cdot residual\_grammar\_vector_i[k]),$ for $k \in [1..D_r]$

-
- **Compute Composite Orthography Vector similarly over D_r2.**
- **Build a Composite Grammar Tree by merging node sets:**
    - **For nodes with identical `node_id` in multiple sources, unify children lists; for conflicting labels, prefer the one with higher `AS`.**
- **Set $initial\_AS = (\Sigma_i AS_i) \cdot \beta_g (0.25)$.**
- **Set $decay\_rate = \delta_g (0.003)$.**
- **Create new EcoForm via `CreateEcoForm`, using:**

        - `origin_context = { module: "EcoFormStitcher", cycle_number: current_cycle, source_language: target_language }`
        - `grammar_payload = Composite Grammar Tree`
        - `grammar_vector = comp_grammar_vector_padded(D_g=128)` (zero-pad or upsample from D_r=64)
        - `orthography_vector = Composite Orthography Vector` (upsample/zero-pad from D_r2=16)
        - `initial_AS`, `decay_rate`, `creation_time = now`.
    - For each `id ∈ source_ids`, update its metadata: add `"stitched_into": new_ecoform_id`.
- **Error Codes:**
    - `400 Bad Request` if `source_ids` is empty or invalid.
    - `404 Not Found` if any `id` does not exist.
    - `409 Conflict` if any source is not `Evaporated` or `age_seconds > T_archive_g`.
    - `500 Internal Server Error` on storage failures.

---

6. Threshold Values & Configuration

EcoForm's runtime parameters are defined in a YAML configuration file loaded at startup. All numeric thresholds and dimensions are explicitly listed below.

```
ecoform_config:
  # Activation & Decay
  epsilon_activation: 0.05       # ε_g: below this AS_current → Evaporated
  decay_rate_default: 0.003      # δ_g: decay coefficient per second
  t_reactivate_max: 86400        # 24 h in seconds
  alpha_boost: 0.50              # AS boost factor on reactivation
```

beta_stitch: 0.25 # Scaling factor for initial_AS in stitched unit

t_archive: 2592000 # 30 days in seconds

 # Matching & Similarity

nss_threshold: 0.70 # ρ_g: minimum NSS for query/reactivation

weight_grammar_nss: 0.60 # w_1 in NSS computation

weight_orthography_nss: 0.40 # w_2 in NSS computation

 # Orthography Normalization

diacritic_variance_threshold: 0.02 # Δ_1

ligature_similarity_threshold: 0.85 # Δ_2

 # Vector Dimensions

D_g: 128 # Grammar vector dimension

D_o: 32 # Diacritic profile dimension

D_l: 32 # Ligature profile dimension

D_r: 64 # Residual grammar vector dimension

D_r2: 16 # Residual orthography vector dimension

 # Sharding & Scaling

num_shards: 4

max_active_per_shard: 100000

 # Scheduler Intervals (in seconds)

evaporation_check_interval: 60

metrics_report_interval: 300