

EFMCore API Specification

Overview

The `EFMCore` class implements the integrated control loop for the Entropica Forensic Model (EFM), coordinating capsule introspection, projection-based forecasting, forensic tracing, and autonomous response.

This API enables integration into robotic swarms, distributed agents, or intelligent systems requiring symbolic introspection and fault-tolerant autonomy.

Class: `EFMCore`

```
class EFMCore:  
    def __init__(self, bim, csl, ctm):  
        ...
```

Parameters

Name	Type	Description
<code>bim</code>	<code>BehaviorIntegrityModule</code>	Tracks symbolic deviation between steps
<code>csl</code>	<code>CausalSymbolicLearner</code>	Learns and applies causality motifs from failure history
<code>ctm</code>	<code>CognitiveTraceMatrix</code>	Stores symbolic trace graph

Method: `tick()`

```
def tick(self, capsule_state: dict) -> Tuple[str, List[str]]:  
    ...
```

Description

Executes a single lifecycle loop for a capsule.

Returns

Output	Type	Description
trace_level	str	Adjusted trace depth (L1-L4) for capsule memory fidelity
actions	List[str]	Triggered DSL actions such as partition, rollback, or escalate

DSL Action Predicates

```
if capsule.DriftRisk > 0.85:  
    capsule.partition(mode="safe")  
    lineage.notify_drift(parent_id)
```

Supported DSL Actions

Action	Description
partition()	Isolates the capsule for recovery
rollback()	Reverts to last stable symbolic state
escalate()	Reports hereditary risk to cluster-level node

Subcomponents

TPE : TemporalProjectionEngine

- Forecasts system collapse (TTF) based on symbolic coherence degradation.

CAC : CognitiveApertureController

- Adjusts trace resolution using alpha_dyn based on TTF and symbolic pressure.

RPC : ReflectiveProjectionCheck

- Performs self-assessment of symbolic lineage and triggers DSL actions.

IPE : InternalProjectionEngine

- Analyzes capsule ancestry to detect hereditary drift and calculate entropy.

Example Lifecycle

```
capsule_state = {  
    "id": "c.4041",  
    "entropy": 0.93,  
    "stability": 0.14,  
    "parent_id": "c.4029"  
}  
  
trace_level, actions = efm.tick(capsule_state)
```

Returns:

```
trace_level = "L4"  
actions = ["partition", "notify_drift"]
```

Notes for Integration

- The EFMCore expects real-time or batch inputs from a symbolic system.
 - Designed for deployment within robotic agents, financial AI, LLM monitoring, and distributed policy governance.
 - Best used in tandem with Plonky2-based ZK-SP layer and d-CTM clustering (Booklet 4).
-

TODO (Optional Expansions)

- Add `export_proof()` method to connect to ZK-SP aggregator
 - Add `register_node()` to support d-CTM swarm synchronization
 - Define `EFMResponse` schema for RESTful API deployment
-

License

MIT / Entropica Foundation 2025