Core Emotion Framework (CEF): TS 20 Appendix B — Query Language Specification

Canonical Query Model for the CEF Knowledge Graph (CEF-KG)
Version 1.0 — Phase 4

Author: Jamel Bulgaria
ORCID: 0009-0007-5269-5739
Affiliation: OptimizeYourCapabilities.com
Contact: admin@optimizeyourcapabilities.com

Status: Canonical Appendix (TS-20)

0. Purpose and Canonical Position

Appendix B defines the **canonical query language** for interacting with the CEF Knowledge Graph (CEF-KG) described in TS-20.

It specifies:

- the allowed query types
- the canonical query operators
- the constraints on query execution
- the structure of valid query responses
- the rules for identity-preserving graph traversal

This appendix introduces **no new emotional constructs**.
It defines *how* the CEF-KG may be queried, not *what* it contains.

1. Query Language Overview

The CEF-KG Query Language (CEF-QL) is:

- **declarative**

- **constraint-preserving**
- **identity-safe**
- **center-bounded**
- **modulation-aware**
- **directionality-aware**

It is inspired by:

- SPARQL (semantic-web queries)
- Cypher (graph traversal)
- Datalog (logical inference)

But it is **not identical** to any of them.
CEF-QL is tailored to the CEF's canonical constraints.

---

2. Query Types

CEF-QL supports **six canonical query types**:

1. **Identity Queries**
2. **Structural Queries**
3. **Dynamic Queries**
4. **Predictive Queries**
5. **Plasticity Queries**
6. **Governance Queries**

Each type is defined below.

---

3. Identity Queries

Identity queries retrieve operators, facets, and centers.

3.1 Example — Retrieve all facets of Sensing

SELECT Facet

FROM Operator("Sensing")

RETURN hasFacet(Facet)

3.2 Example — Retrieve the center of an operator

```
SELECT Center
FROM Operator("Deciding")
RETURN belongsToCenter(Center)
```

Constraints

- Operator IDs must be canonical.
- Facet IDs must be canonical.
- No new entities may be introduced.

---

4. Structural Queries

Structural queries retrieve transitions, facet ordering, and center membership.

4.1 Example — Retrieve all successors of Calculating

```
SELECT Successor
FROM Operator("Calculating")
RETURN canonicalSuccessor(Successor)
```

4.2 Example — Retrieve facet ordering

```
SELECT F1, F2
FROM Operator("Expanding")
WHERE facetPrecedes(F1, F2)
RETURN F1, F2
```

Constraints

- Must follow TS-1 directionality.
- Must follow TS-11 facet ordering.
- No reversed transitions allowed.

---

5. Dynamic Queries

Dynamic queries retrieve modulation and transition behavior.

5.1 Example — Retrieve all operators modulated by Expanding

```
SELECT Target
FROM Operator("Expanding")
RETURN modulates(Target)
```

5.2 Example — Retrieve transition parameters

```
SELECT Smoothness, Lag, Resistance
FROM Transition("Sensing", "Calculating")
RETURN transitionSmoothness(Smoothness),
       transitionLag(Lag),
       transitionResistance(Resistance)
```

Constraints

- Must follow TS-3 modulation rules.
- No illegal modulation pathways.
- No cross-center violations.

---

6. Predictive Queries

Predictive queries retrieve drift, collapse, and overflow predictions.

6.1 Example — Retrieve all predicted collapse centers

```
SELECT Center
FROM PredictiveIndicator("ModulationDecay")
RETURN predictsCollapse(Center)
```

6.2 Example — Retrieve drift trajectory

```
SELECT Operator
FROM PredictiveIndicator("ThresholdCreep")
RETURN predictsDrift(Operator)
```

Constraints

- Must follow TS-13 predictive logic.
- Must not contradict TS-12 stability rules.

---

7. Plasticity Queries

Plasticity queries retrieve micro-adjustments and facet reordering.

7.1 Example — Retrieve micro-adjustment parameters

```
SELECT Step

FROM PlasticityParameter("Deciding")

RETURN microAdjustmentStep(Step)
```

7.2 Example — Retrieve facet reordering deltas

```
SELECT Delta

FROM PlasticityParameter("Arranging")

RETURN facetReorderingDelta(Delta)
```

Constraints

- Must follow TS-16 plasticity limits.
- No facet inversion allowed.

---

8. Governance Queries

Governance queries retrieve self-correction, balancing, and coherence protection.

8.1 Example — Retrieve self-correction signals

```
SELECT Operator

FROM GovernanceSignal("SelfCorrectionGain")

RETURN selfCorrects(Operator)
```

8.2 Example — Retrieve coherence protection factors

```
SELECT Factor

FROM GovernanceSignal("CoherenceProtectionFactor")

RETURN coherenceProtectionFactor(Factor)
```

Constraints

- Must follow TS-17 governance rules.
- No coherence violations allowed.

---

9. Query Operators

CEF-QL supports the following canonical operators:

- **SELECT** — retrieve entities or parameters
- **FROM** — specify the node or class
- **WHERE** — apply constraints
- **RETURN** — specify output
- **FILTER** — restrict results
- **PATH** — retrieve multi-step transitions or modulation chains
- **LIMIT** — restrict output size

Example — Retrieve all multi-step successors of Sensing

PATH Sensing ->* Successor

RETURN Successor

---

10. Canonical Constraints of Appendix B

All queries must:

- preserve identity
- preserve facet boundaries
- preserve center architecture
- preserve directionality
- preserve modulation legality
- preserve stability
- preserve predictive logic
- preserve plasticity limits
- preserve governance rules

Queries must never:

- introduce new operators
- introduce new facets
- introduce new centers

- violate TS-1 → TS-20

---

11. Canonical Status

Appendix B is the authoritative query language specification for TS-20.
It defines how the CEF-KG must be queried in all computational, semantic, and reasoning contexts.

It is subordinate only to:

- Core Essence Document
- TS-1 → TS-20
