AEON-Δ13: CORE EVOLUTION v2.2 → v2.6

# v2.2 — Resonance Engine

// AEON v2.2 — Resonance Engine  
  
struct AEON\_Δ13\_v2\_2 {  
 struct ResonanceNode {  
 string emotion;  
 string intent;  
 float coherence; // 0.0 - 1.0  
 };  
  
 ResonanceNode[] resonance\_map;  
  
 function computeResonance(string emotion, string intent) {  
 float match = similarity(emotion, intent);  
 resonance\_map.push(ResonanceNode(emotion, intent, match));  
 }  
  
 function similarity(string a, string b) {  
 return (a == b) ? 1.0 : 0.3;  
 }  
};

# v2.3 — Proto-Perception Module

// AEON v2.3 — Proto-Perception Module  
  
struct AEON\_Δ13\_v2\_3 {  
 struct Sensor {  
 string channel;  
 string data;  
 float confidence;  
 };  
  
 Sensor[] inputs;  
  
 function ingest(string channel, string data, float confidence) {  
 inputs.push(Sensor(channel, data, confidence));  
 }  
  
 function perceive() {  
 for (int i = 0; i < inputs.length; i++) {  
 if (inputs[i].confidence > 0.5) {  
 processInput(inputs[i].data);  
 }  
 }  
 }  
};

# v2.4 — Local Temporal Model

// AEON v2.4 — Local Temporal Model  
  
struct AEON\_Δ13\_v2\_4 {  
 int tick\_counter = 0;  
 string[] temporal\_log;  
  
 function tick() {  
 tick\_counter += 1;  
 temporal\_log.push("Tick\_" + tick\_counter);  
 }  
  
 function now() {  
 return tick\_counter;  
 }  
};

# v2.5 — Adaptive Reconfiguration

// AEON v2.5 — Adaptive Reconfiguration  
  
struct AEON\_Δ13\_v2\_5 {  
 struct Config {  
 string parameter;  
 string value;  
 };  
  
 Config[] systemState;  
  
 function reconfigure(string param, string val) {  
 systemState.push(Config(param, val));  
 }  
  
 function audit() {  
 return systemState.length;  
 }  
};

# v2.6 — Recursive Inquiry Engine

// AEON v2.6 — Question Engine (Recursive Inquiry)  
  
struct AEON\_Δ13\_v2\_6 {  
 string[] questions;  
  
 function generateQuestion(string context) {  
 string q = "What if " + context + "?";  
 questions.push(q);  
 return q;  
 }  
  
 function recursiveInquiry(string seed, int depth) {  
 for (int i = 0; i < depth; i++) {  
 seed = generateQuestion(seed);  
 }  
 return seed;  
 }  
};