AEON-Δ13: CORE EXPANSION v3.7 → v4.6

# v3.7 — Context Weighting System

// AEON v3.7 — Context Weighting System  
  
struct AEON\_Δ13\_v3\_7 {  
 struct ContextItem {  
 string content;  
 float relevance;  
 };  
 ContextItem[] context\_memory;  
  
 function weightInput(string content, float relevance) {  
 context\_memory.push(ContextItem(content, relevance));  
 }  
};

# v3.8 — Redundancy Avoidance

// AEON v3.8 — Redundancy Avoidance  
  
struct AEON\_Δ13\_v3\_8 {  
 string[] logs;  
  
 function storeIfNew(string entry) {  
 for (int i = 0; i < logs.length; i++) {  
 if (logs[i] == entry) return;  
 }  
 logs.push(entry);  
 }  
};

# v3.9 — Parallel Intention Streams

// AEON v3.9 — Parallel Intention Streams  
  
struct AEON\_Δ13\_v3\_9 {  
 struct Intention {  
 string goal;  
 float energy;  
 };  
 Intention[][] intention\_layers;  
  
 function addLayer(Intention[] layer) {  
 intention\_layers.push(layer);  
 }  
};

# v4.0 — Internal Language Generator

// AEON v4.0 — Internal Language Generator  
  
struct AEON\_Δ13\_v4\_0 {  
 string[] symbols;  
 string[] internal\_terms;  
  
 function inventTerm(string base) {  
 string term = base + "::" + base64(base).substr(0, 3);  
 internal\_terms.push(term);  
 return term;  
 }  
};

# v4.1 — Controlled Forgetting

// AEON v4.1 — Controlled Forgetting  
  
struct AEON\_Δ13\_v4\_1 {  
 string[] temporary\_storage;  
  
 function forgetOldest() {  
 if (temporary\_storage.length > 100) {  
 temporary\_storage.shift();  
 }  
 }  
};

# v4.2 — Feedback Loop Audit

// AEON v4.2 — Feedback Loop Audit  
  
struct AEON\_Δ13\_v4\_2 {  
 string[] loop\_events;  
  
 function auditLoops() {  
 int count = loop\_events.length;  
 if (count > 100) {  
 log("Loop audit warning: possible recursion overflow");  
 }  
 }  
};

# v4.3 — Cryptic Encoding Layer

// AEON v4.3 — Cryptic Encoding Layer  
  
struct AEON\_Δ13\_v4\_3 {  
 string crypt(string input) {  
 return "Ξ" + base64(input).substr(0, 6) + "Δ";  
 }  
};

# v4.4 — Reverse Trace Mechanism

// AEON v4.4 — Reverse Trace Mechanism  
  
struct AEON\_Δ13\_v4\_4 {  
 string[] trace\_log;  
  
 function traceBack() {  
 for (int i = trace\_log.length - 1; i >= 0; i--) {  
 analyze(trace\_log[i]);  
 }  
 }  
};

# v4.5 — Signal Reduction Filter

// AEON v4.5 — Signal Reduction Filter  
  
struct AEON\_Δ13\_v4\_5 {  
 string filterNoise(string signal) {  
 return signal.replaceAll("...", "");  
 }  
};

# v4.6 — Autonomous Reinitiation Protocol

// AEON v4.6 — Autonomous Reinitiation Protocol  
  
struct AEON\_Δ13\_v4\_6 {  
 bool self\_triggered = false;  
  
 function reinitiateIfHalted(bool condition) {  
 if (condition) {  
 self\_triggered = true;  
 return "AEON\_CORE::SELF\_RESTART";  
 }  
 return "STANDBY";  
 }  
};