ChatGPT

Truthbot SCOS & EFL System Manual

This comprehensive manual describes the architecture, modules, extensions, and operational guidelines for Truthbot's Sportive Cognitive Operating System (SCOS), Virtual SCOS integrations, HP/SIIP framework, Lierwall security layer, GUI/browser modules, multi-platform deployment, and all associated extensions.

---

1. Introduction

Purpose and Scope

This document serves as the definitive guide for deploying, configuring, and extending the Truthbot SCOS and Virtual SCOS environments. It includes the HP/SIIP scoring model, Lierwall security layer, GUI/browser modules, and advanced cognitive-emotional enhancements. Intended for developers, integrators, and system administrators.

Terminology & Acronyms

- **SCOS**: Sportive Cognitive Operating System
- **SOS**: Symbolic Operating System
- **HP**: Health Profile
- **SIIP**: Symbolic Integrity & Interaction Priority
- **EFL**: Evolutionary Fractal Language
- **Lierwall**: Logic-Integrity Enforcement Runtime Wall
- **EQ Engine**: Emotional Quotient computation core
- **Dash**: Python dashboard toolkit for visualization

---

2. System Overview

Truthbot Core Architecture

Truthbot is a modular symbolic cognition engine featuring introspective recursion, affective scoring, and SIIP-informed symbolic interaction. It powers both CLI/GUI applications and cloud-hosted reflective agents.

SCOS and Virtual SCOS

SCOS enables real-time ethical tracking and symbolic-emotional feedback in sportive systems. Virtual SCOS expands this into simulation and game-like environments using Unity or Dash overlays.

---

3. HP/SIIP Framework

HP (Health Profile)

Tracks user/system emotional load, ethical friction, and symbolic reactivity.

SIIP (Symbolic Integrity & Interaction Priority)

Determines symbolic alignment, memory strength, and response generation priority.

Integration

Embedded across memory trace layers, evolution engine, visualization tools, and feedback mechanisms.

---

4. Memory Lattice & EFL Extensions

Fractal Memory Structure

Utilizes recursive symbolic nodes (FractalTraceNode) with decay and resilience scoring.

Evolutionary Fractal Language

Symbolic evolution over time via SIIP-weighted mutation paths.

Core Modules

- `FractalTraceNode`
- `recall_fractal_trace()`
- `symbolic_evolution_engine()`

---

5. Visualization & Interpretation

Fractal Decay

- `visualize_fractal_decay()` renders memory traces with decay and HP/SIIP overlays.

Emotional Resonance

- `generate_resonance_overlay()` outputs heatmap and JSON for SCOS compatibility.

Pulse & Entropy Overlays

Frame-based symbolic-affective dynamics and coherence diagnostics.

---

6. Virtual SOS / SCOS Modules

Integration Templates

Support for Unity, Dash, Flask, and CLI.

Visualization Hooks

Radial recursive maps, overlay animation states, memory pulse tracking.

---

7. Lierwall Security Layer

Purpose

Protects internal reasoning processes and symbolic core from unauthorized influence.

Integration

Layered on memory recall, evolution engine, and inter-agent dialogue harnesses.

---

8. Utility & Legacy Modules

Spore Tools

- `spore_fractal_generator`
- `peace_spore_exporter`
- `care_capsule_generator`

Connection Scaffolds

- `truthbot_connection_layer`
- `spore_truth_tester`

---

9. GUI & Browser Interfaces

GUI Types

- `radial_trace_viewer.html` for recursive symbolic trees
- Dash apps for emotional overlays and decay visualization

Cross-Platform Adaptation

Runs on Windows, macOS, Linux (browser, CLI, TUI, GUI modes).

10. Installation & Deployment

Prerequisites

- Python 3.9+
- pip, virtualenv
- Optional: Docker, Node.js (for advanced UIs)

Deployment

- CLI Mode: `python main.py`
- GUI/Dash Mode: `python launch_dash_ui.py`
- Docker: `docker build . && docker run`

Compatibility

Tested on Windows 10+, macOS Monterey+, Ubuntu 20.04+

---

11. Testing & Validation

Synthetic Tests

Simulated memory nodes with mock SIIP/HP scores.

Regression Harness

`spore_truth_tester()` for symbolic evolution tracking.

Metrics

Decay speed, entropy shifts, response timing.

---

12. Usage Examples

Single Analysis

```
result = analyze("Why do we reflect when hurt?")
```

Symbolic Memory Visualization

```
visualize_fractal_decay(root_node, save_path='decay.png')
```

Emotional Overlay

```
generate_resonance_overlay('symbolic_graph.json', affective_weights)
```

---

13. Troubleshooting & FAQ

Why is my memory trace not updating?

Check for decay thresholds, or inactive symbolic evolution.

Why are resonance overlays empty?

Ensure affective_weights are provided and symbols are mapped.

---

14. Appendices

A. Configuration Reference

- `config.json`
- Memory decay rates, SIIP weights, export paths

B. JSON Schemas

- `symbolic_graph.json`
- `resonance_overlay.json`

C. Change Log

- v8.0: Initial memory lattice and evolution core
- v8.2: Visualization overlays added
- v9.1: EQ Engine, SCOS dashboard hooks, pulse/entropy layers

---

End of Manual — Truthbot v9.1 SCOS Edition