Sovereign Sanctuary Elite - IDE Handover Pack

Version: 2.0.0 **Date:** 2026-01-25 **Author:** Manus AI for Architect

---

1. Overview

This IDE handover pack provides a complete, cleaned, and bug-fixed environment for the **Sovereign Sanctuary Elite** system. It consolidates all relevant code, configurations, and documentation from recent development cycles into a single, coherent project structure. The pack is designed for seamless import into a modern IDE (e.g., VS Code) to support the next phase of the elite build.

This document serves as the primary guide for understanding the architecture, setting up the development environment, and operating the core components of the system.

2. Core Architecture

The system is architected around a **decentralized, self-healing network of AI agents** that perform automated tasks with cryptographic verification. Key architectural pillars include:

- **Evidence-Bound Execution:** Every significant action is recorded on an immutable ledger, creating a verifiable audit trail.
- **Model Mesh Routing:** A dynamic LLM selector (`model_mesh_router.py`) routes requests to the most appropriate model based on cost, performance, and capability, ensuring resilience and efficiency.
- **Self-Healing Daemons:** Autonomous monitors (`self_heal_monitor.py`) continuously check the health of system nodes and execute recovery protocols when failures are detected.

- **Cryptographically Verified Agents:** AI agents, such as the `verified-agent-elite.ts`, sign their outputs to guarantee authenticity and integrity.
- **Flight Control System:** A file system watcher (`flight_control_daemon.py`) automates the process of sealing evidence and updating system status boards (SITREPs).

3. Project Structure

The repository is organized into the following key directories:

Directory	Description
`/agents`	Contains the core AI agent implementations (TypeScript and Python).
`/config`	Centralized configuration files for the swarm and model mesh.
`/core`	Core Python modules for the daemon and data models.
`/docs`	System documentation, architecture diagrams, and specifications.
`/scripts`	Deployment, file push, and utility scripts.
`/tests`	Unit and integration tests for all system components.
`/tools`	Supporting tools for self-healing, flight control, and evidence sealing.
`/ui`	Front-end components, including the Fortress Elite Cockpit.

4. Setup and Installation

To set up the development environment, follow these steps. This assumes you have **Python 3.10+** and **Node.js 18+** installed.

4.1. Python Environment

First, create and activate a virtual environment:

```
python3 -m venv .venv
source .venv/bin/activate
```

Next, install the required Python dependencies:

```
pip install -e .[dev]
```

This command installs the project in editable mode and includes the development dependencies specified in `pyproject.toml`.

4.2. Node.js Environment

Install the required Node.js packages using `pnpm` or `npm`:

```
pnpm install
```

This will install the dependencies listed in `package.json`, including the AI SDK, Zod for schema validation, and TypeScript development tools.

4.3. Configuration

Create a `.env` file in the root of the project to store secrets and environment-specific settings. At a minimum, you should define:

```
 # Used by the verified-agent-elite.ts for signing outputs
AGENT_SIGNING_KEY="your-secret-signing-key"

# OpenAI API Key (or other compatible API)
OPENAI_API_KEY="sk-..."
```

5. Running Core Components

The system's core components can be run using the scripts defined in `pyproject.toml` and `package.json`.

5.1. Self-Heal Monitor

The self-heal monitor continuously checks the health of the distributed nodes. To run it:

```
sanctuary-heal --interval 60
```

This will start the monitor with a 60-second check interval. Health status and actions are logged to `logs/self_heal.log`.

5.2. Flight Control Daemon

The flight control daemon watches for file changes in the `evidence` directory and automates the sealing process.

```
sanctuary-flight --verbose
```

5.3. Verified PR Review Agent

The TypeScript-based agent can be run directly to review a diff file. It will output a cryptographically signed review.

```
pnpm agent:review
```

6. Bug Fixes and Enhancements

This version (2.0.0) includes several critical bug fixes and enhancements over previous iterations:

- **`self_heal_monitor.py`:**
    - **Fixed:** Bare `except` clauses were replaced with specific exception handling to prevent swallowing critical errors.

    - **Fixed:** The disk space calculation was inverted, causing false positives. It now correctly checks for minimum free space.
    - **Enhanced:** Added graceful shutdown handling to ensure clean termination.
    - **Enhanced:** Implemented structured logging for easier parsing and analysis.
- `flight_control_daemon.py`:
    - **Fixed:** A potential infinite loop was resolved by adding better event filtering and debouncing logic.
    - **Enhanced:** Error handling was improved to provide more specific feedback on subprocess failures.
    - **Enhanced:** The daemon now validates the environment on startup to ensure required tools are present.
- `verified-agent-elite.ts`:
    - **Fixed:** The HMAC signature generation was using an incorrect crypto method, which has been corrected to `crypto.createHmac`.
    - **Fixed:** Token estimation logic was refined for more accurate cost tracking.
    - **Enhanced:** Added automatic retry logic with exponential backoff for API calls to the LLM, improving resilience.

7. Next Steps

With the environment set up, you can now proceed with the elite build. Key areas for immediate focus include:

1. **Run Integration Tests:** Execute the test suite to confirm that all components are functioning correctly after the merge.

    ```
    pytest
    pnpm test
    ```

2. **Extend the Model Mesh:** Enhance the `model_mesh_router.py` to include more sophisticated routing logic, such as latency-based or capability-based routing.

3. **Develop the Fortress Elite Cockpit:** Continue development of the `Fortress-Elite-Cockpit.html` to provide a real-time visualization of the swarm's health and operational status.

This handover pack provides a stable, verified foundation for the next stage of development. The system is now more robust, resilient, and ready for expansion.