Technology Overview

HACS Core — System Architecture (Docs Version)

1. Introduction

This document provides a technical overview of the architecture that powers HACS Core. It describes how the system operates on top of external LLMs, how internal cognitive modules communicate, and how the Core maintains state, memory, and autonomous behavior.

The information here is intended for engineers, integrators, researchers, and enterprise clients evaluating the Core for integration.

2. System Model

HACS Core uses a two-layer architecture:

Layer 1 — Computational Fabric (External LLMs)

- Stateless predictive models (OpenAI, XAI, DeepSeek, etc.).
- Provide raw token generation and inference power.
- Do not store memory, identity, or reasoning structure.

Layer 2 — HACS Core (Internal Cognitive Engine)

- Stateful reasoning architecture.
- Maintains long-term memory, internal state, processes, and logic.
- Functions as the "mind" operating above the computational fabric.

This separation allows the Core to behave consistently regardless of which LLM provider powers the computing layer.

3. Core Components

3.1 NervoBus (Orchestration Layer)

The primary internal routing system: - Receives user inputs. - Determines intent and semantic weight. - Routes tasks to the appropriate modules. - Connects memory, processes, and tool integrations. - Ensures consistent state management.

It acts as the "nervous system" of the Core.

3.2 SemanticFlow Engine

A real-time meaning extraction layer: - Identifies meaningful vs. noisy input. - Detects emotional/meta reactions. - Preserves continuity of thought. - Extracts semantic units used for planning, memory, and decision-making. - Blocks irrelevant or destabilizing input from influencing reasoning.

3.3 HolmsAttic (Fractal Memory System)

Long-term memory organized into fractal zones: - L0 (Desk): Active working memory. - L1 (Shelf): Short-term semantic storage. - L2 (Attic): Deep structured memory (checkpoints, history, profiles).

Memory zones are dynamically populated based on semantic relevance, not token history.

3.4 WorkingContext

A lightweight state container: - Stores recent meanings, tasks, preferences, and session state. - Keeps temporary local variables that do not require long-term storage. - Provides a stable environment for multi-step reasoning.

3.5 Background Engine

Autonomous execution subsystem running tasks without user commands: - Scheduled tasks - Chains of actions - Prioritized execution - Learning based on value scores - Long-running pipelines - Resource budgeting - Idea/content generation - Strategic planning routines

3.6 NervoAPI

External programmable interface that allows: - Bot integration (Telegram, Discord, etc.) - Website assistants - Robotics control - IoT connectivity - Embedded cognitive modules - Third-party application integration

NervoAPI isolates each user into a personal Core clone, ensuring privacy and long-term state separation.

4. Data Flow Overview

1. Input → SemanticFlow
   Text is parsed for meaning and filtered from noise.
2. Semantic unit → NervoBus
   Intent and relevance are determined.
3. Task → Appropriate Module
   Examples: memory operation, finance handler, background process.
4. Response → Persona Layer
   System formats the answer.
5. Memory Update
   Relevant information is written into HolmsAttic or WorkingContext.

5. Autonomy Model

The system supports: - Self-generated tasks - Self-correcting loops - Self-improving patterns - Idea generation routines - Long-term strategies

This is engineered autonomy based on logic, memory, and semantic structure.

6. Safety Model

The Core includes: - Noise filtering - Intent validation - Memory gatekeeping - Controlled tool execution - Human-required confirmations - Semantic Key verification - User core isolation - Encrypted API-key storage

7. Dependencies

HACS Core requires: - Access to at least one external LLM model - A persistent database (PostgreSQL recommended) - Secure file/state storage - Network access for integrations - Encrypted API-key storage

It does not rely on any specific cloud provider.

8. Supported Use Cases

- Personal cognitive assistants
- Autonomous brand/content engines
- Business workflow automation
- Strategic planning tools
- Robotics / embedded systems
- Private AI instances for individuals or enterprises

9. Limitations

HACS Core: - Is experimental - May produce incorrect or incomplete outputs - Is not professional advice - Is not conscious - Inherits LLM limitations - Must operate within permissions

10. Contact

Enterprise integration, licensing, or support:
developers@hacs.world