NervoAPI Documentation

HACS Core — External Integration Interface (Docs Version)

---

1. Introduction

NervoAPI is the official interface for connecting external systems to HACS Core.
It enables applications, bots, robots, services, and user devices to interact with Core instances, trigger cognitive workflows, retrieve semantic output, and control autonomous processes.

NervoAPI provides: - secure communication, - stateful context management, - isolated Core clones per user, - high-level cognitive operations, - real-time event streaming.

This document defines the API structure, authentication flow, endpoints, and best practices.

---

2. Core Concepts

2.1 Core Instance

Every user or integrated device may have a **dedicated Core clone**, which maintains: - personal memory, - internal state, - process history, - autonomous background tasks.

2.2 Stateless Fabric vs Stateful Core

NervoAPI interacts only with the Core, *not* with underlying LLMs.
This ensures: - deterministic memory, - traceable logic, - semantic continuity, - isolation between users.

2.3 Semantic Tasks

Each NervoAPI call triggers a **task** inside the Core: - message processing, - financial operation, - generation request, - strategy pipeline, - background process activation.

2.4 Event Stream

Core can push: - system events, - background process outputs, - autonomous insights, - task completions.

through WebSocket channels.

---

3. Authentication

3.1 API Keys

Each Core instance uses a secure access token:

Format:

`hacs_live_<random_32_chars>`

API keys must be stored encrypted.

3.2 Semantic Key (Optional)

For sensitive actions (finances, memory editing, device control), Core may require: - user-defined phrase, - video identity check, - biometric pattern.

Semantic Key never leaves the encrypted enclave.

---

4. Base URL

`https://api.hacs.world/nervo`

All endpoints are prefixed with:

`/v1`

---

5. Endpoints

5.1 Send Message to Core

POST `/v1/core/message`

Send user input to the Core.

Request

```
{
  "core_id": "core_123",
  "message": "Create a 3-step marketing plan",
  "meta": {
    "source": "telegram_bot"
  }
}
```

Response

```
{
  "ok": true,
  "core_id": "core_123",
  "reply": "Here is a 3-step plan...",
  "task_id": "task_9fd8ab",
  "state": {
    "semantic_weight": 0.82,
    "intent": "marketing.plan"
  }
}
```

5.2 Fetch Core State

GET `/v1/core/state/{core_id}`

Response

```
{
  "ok": true,
  "core_id": "core_123",
  "active_processes": 5,
  "memory_usage": {
    "l0": 128,
    "l1": 412,
    "l2": 34
  },
  "last_update": "2025-12-03T11:32:00Z"
}
```

5.3 Trigger Background Process

POST `/v1/core/process/run`

Request

```
{
  "core_id": "core_123",
  "process": "build_strategy",
  "params": {
    "focus": "Japan market"
  }
}
```

Response

```
{
  "ok": true,
  "process_id": "proc_88a3bc"
}
```

5.4 Get Event Stream

WebSocket `/v1/core/stream/{core_id}`

Events may include:

```
{
  "event": "background.completed",
  "process_id": "proc_88a3bc",
  "payload": {
    "summary": "Strategy draft ready."
  }
}
```

5.5 Memory Access (Safe Mode)

GET `/v1/core/memory/recent/{core_id}`

Returns safe semantic memory items, excluding private zones.

```
{
  "items": [
    {
      "zone": "brand",
      "value": "...text..."
    }
  ]
}
```

5.6 Execute Tool / Integration

POST `/v1/core/tool`

```
{
  "core_id": "core_123",
  "tool": "gpay.transfer",
  "params": {
```

```
    "amount": 100,
    "currency": "USDT",
    "to": "wallet_x"
  }
}
```

Response:

```
{
  "ok": true,
  "status": "pending_confirmation"
}
```

If sensitive → Core will request Semantic Key.

5.7 Clone Core Instance

POST `/v1/core/clone`

```
{
  "source_id": "core_123",
  "new_owner": "user_884",
  "options": {
    "include_memory": false
  }
}
```

Response:

```
{
  "ok": true,
  "new_core_id": "core_884"
}
```

---

6. Rate Limits

Default: - **60 requests / minute**
- **WS soft limit: 5 concurrent channels**

Enterprise licenses may increase these.

---

7. Response Codes

- 200 — success

- `400` — malformed request
- `401` — invalid token
- `403` — forbidden (Semantic Key required)
- `429` — rate limit
- `500` — internal Core error (experimental behavior)

---

8. Best Practices

- Always check `semantic_weight`.
- Avoid sending noisy/ambiguous messages.
- Use background processes for long-running tasks.
- Handle WebSocket events for autonomous insights.
- Store API keys encrypted.
- Never expose Semantic Key externally.

---

9. Deprecation Policy

NervoAPI follows a **rolling 12-month stability window**:

- Deprecated endpoints remain operational for 12 months.
- Changes are announced via `/docs/changelog`.

---

10. Contact

API support and enterprise integration:
`api@hacs.world`