NextFi Advisors

AI SYSTEMS
NEXTFI AI INSIGHTS SERIES

AGENTIC AI ARCHITECTURE

Meet Herman

The architecture behind **Herman**,
an Hermes-based autonomous AI assistant developed by NextFi.

Open Source & Agentic AI · June 2026 · v1.0

Architecture Overview *all layers*

The Hermes architecture traces every request through **six distinct layers**, each with a specific responsibility in the request-to-response pipeline.

ROW 1
CHANNELS
entry points

ROW 2
ROUTE
tagging

ROW 3
REASONING LOOP
thinking

ROW 4A
SKILLS & TOOL LAYER
doing

ROW 4B
DATA INTEGRATIONS
real-world data

ROW 5
DELIVERY & OUTPUT
returning results

ROW 6
MEMORY & PERSISTENCE
remembering

ROW 4 LAYOUT NOTE

The **Skills & Tool Layer** and **Data Integrations** share a single horizontal row in the diagram — side-by-side capabilities at the same level of abstraction.

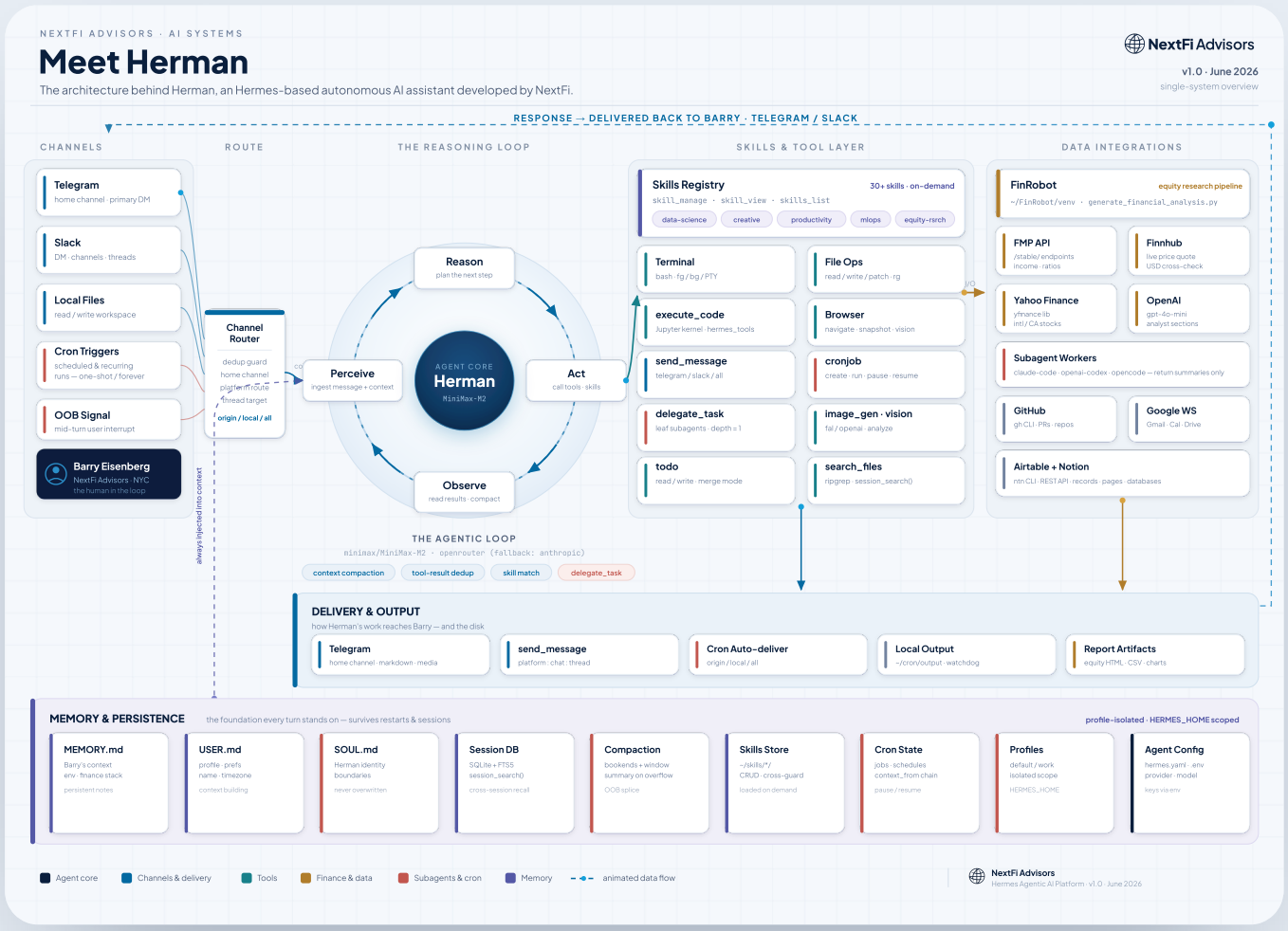

Figure 1 — Hermes Agentic AI Architecture Overview

ROW 1

CHANNELS

entry points

Receives requests from the outside world and passes them into the system. The Channels row doesn't interpret meaning — it collects the message, attaches basic routing metadata, and hands it off to the Route row. The source channel is metadata; the reasoning engine doesn't care where the request came from.

> **Why this matters:** The Channels row makes Hermes channel-agnostic. Telegram, Slack, Local Files, Cron Triggers, and OOB Signals all look identical once they reach the Route row — the reasoning engine is written once and consumes any channel.

COMPONENTS

Telegram

primary · home DM

Primary human-facing input. The user's configured Telegram account is the default delivery target and primary input channel.

Slack

DM · channels · threads

Secondary messaging channel. Supports both direct messages and group channels.

Local Files

~/agents/hermes-spike

Filesystem reads from the Hermes workspace. Documents, scripts, spreadsheets, and CSVs can all be used as input. Hermes can read, parse, and reason over any file on the host machine.

Cron Triggers

scheduled · recurring / one-shot

Scheduled jobs fire on a configured schedule (e.g., a daily equity report at 9am). When a scheduled task runs, its output is treated as input — it flows through the system just like a user message.

OOB Signal

mid-turn interrupt

Out-of-Band signal. A system-generated interrupt injected mid-conversation to communicate events that need to alter the current reasoning loop before it completes: a subagent finishing, a threshold alert firing, a tool returning unexpected output.

User

human in the loop

The human-in-the-loop. Every request originates from or passes through a human decision-maker. Hermes is an assistant, not an autonomous system making independent choices.

Key Concept — Normalization Layer: The Channels row takes inputs from wildly different sources — a text message on Telegram, a file on disk, a scheduled job firing at 9am, an internal system interrupt — and normalizes them into a single format the rest of the system processes uniformly. The source channel is metadata, not signal.

ROW 2 **ROUTE** *tagging*

Tags every inbound request with routing metadata — dedup, home channel assignment, thread targeting, and delivery scope. Route is purely metadata: it does not process the request content itself. It reads the envelope, not the letter.

> **Why this matters:** Delivery decisions are made before reasoning begins. Rather than having the reasoning engine figure out where to send the result after it finishes, the delivery target is attached upfront — allowing cron jobs to auto-deliver correctly and preventing duplicate messages from consuming reasoning resources twice.

COMPONENTS

Channel Router

annotation engine

The routing decision engine. Before any request enters the reasoning loop, the Channel Router annotates it with:

Key Concept — Annotation Layer: Route is an annotation layer, not a processing layer. It doesn't understand the request — it only attaches the routing tags downstream components need to deliver the result correctly. Think of it as the mailroom: it doesn't read the letters, it sorts them into the right bins.

ROW 3

REASONING LOOP

thinking

Thinks. Receives a tagged request, runs it through a Perceive → Reason → Act → Observe loop, and coordinates all downstream rows. This is the brain of Hermes. Everything else in the diagram — Channels, Route, Skills, Tools, Data Integrations, Delivery, Memory — is a capability that this layer directs.

> **Why this matters:** "Agentic" means Hermes doesn't just answer questions — it reasons toward goals. Given "Run an AAPL equity report," the Reasoning Loop autonomously decomposes the task (fetch price → pull income statement → run financial ratios → generate narrative → write CSV → generate HTML → deliver), selects the right skills and tools, and synthesizes the output.

COMPONENTS

Agent Core — Herman (MiniMax-H2)

openrouter

The core reasoning model. Named "Herman" — the persona defined in SOUL.md — running on MiniMax-H2 via openrouter. This is the entity that perceives, reasons, acts, and observes.

Perceive

ingest + context assembly

Ingest the current message plus assembled context: loaded skill definitions, tool schemas, session history, memory references (MEMORY.md, USER.md), and user profile. The assembled context is what the model actually reasons over — not just the raw message.

Reason

plan + sequence

Plan the next step. The model decomposes the goal, selects which skill to load, decides which tool to call first, and sequences the steps. This is where the model is most resource-intensive — doing genuine planning, not just pattern-matching.

Act

invoke tools & skills

Call tools and skills. The model invokes a tool (e.g., `terminal()`, `finnhub_quote()`) or loads a skill (e.g., `hermes-equity-research`) and waits for the result. This is where the real world is touched: code runs, APIs are called, files are written.

Observe

read result + iterate

Read the result of the tool call and incorporate it into the working context. If the result is incomplete or unexpected, the loop returns to Reason. If the goal is reached, the loop exits and passes the result to Delivery & Output.

Agentic Loop Sub-systems

supporting systems

The supporting systems that keep the reasoning loop running correctly:

Key Concept — Goal-Directed and Iterative: The Reasoning Loop doesn't execute a fixed sequence — it evaluates the current state after each tool call and decides what to do next based on whether the goal has been reached. If Finnhub returns an error, the loop may branch to Yahoo Finance as a fallback. The loop is resilient and adaptive, not scripted.

ROW 4A

SKILLS & TOOL LAYER

doing

Provides the executable capabilities the model invokes during the Perceive → Reason → Act → Observe loop. These are capabilities Hermes controls: the code, execution environment, and logic it owns and can modify. Sits on the left half of Row 4, sharing a horizontal band with Data Integrations.

> **Why this matters:** The Reasoning Loop decides what to do — but Skills & Tools actually does it. A tool is a discrete, executable capability. A skill is a multi-step procedure that encodes when and how to use tools together to accomplish a goal. Skills are loaded on demand; tools are always available.

COMPONENTS

SKILLS

Skills Registry

skill_sample · skill_view · skills_list

The CRUD index for all skills. Skills are stored as markdown files in `~/skills/*/*.md`, loaded at startup, and persist across sessions. The Skills Registry is the interface through which the user's institutional knowledge is encoded as reusable, persistent playbooks.

30+ Skills On-Demand

data-science · creative · productivity · equity-research

Skills span multiple categories, loaded when triggered: **data-science** (equity research, financial analysis), **creative** (image generation, ASCII art, architecture diagrams), **productivity** (Notion, Airtable, Maps, PowerPoint, OCR), **equity-research** (specialized domain skill for financial analysis workflows).

TOOLS

Terminal

bash/TTY/PY

Fundamental shell execution. Runs commands, manages foreground/background processes, allocates pseudo-terminals for interactive sessions. File operations are built on top.

execute_code

Jupyter kernel

A persistent Python REPL that exposes `hermes_tools` functions (`terminal`, `read_file`, `write_file`, `browser_*`, etc.) as callable tools. The tools package is imported lazily to minimize startup time.

send_message

platform chat thread

Outbound messaging: the primary delivery mechanism, invoked at the end of a reasoning chain to return results to the user across messaging channels.

delegate_task

leaf blueprints depth 1

Spawns independent subagent processes for parallel subtasks. The system enforces `max_spawn_depth=1` — subagents cannot recursively spawn further subagents.

todo

read / write merge mode

Structured session task list. Manages multi-step workflows within a session.

File Ops

read / write / patch / fp

File operations on the local filesystem. The `fp` suffix references fuzzy-patch matching for targeted edits.

Browser

CDP · navigate snapshot vision

Web interaction via Chrome DevTools Protocol. Used for web-based tools, form submissions, and visual QA.

cronjob

cron run pause resume

The job scheduler. Manages scheduled tasks with configurable delivery targets. Supports pause/resume without deleting the job definition.

image_gen / vision

fail / success analyze

Image generation with fallback. Calls external image generation APIs and can attempt repair if the first generation fails.

search_files

ripgrep · session_search()

Fast filesystem search. Used for both live filesystem search and for querying the session history database.

Key Concept — Skills vs. Tools: A **tool** is a discrete, executable capability — a single action (run a shell command, read a file, call an API). A **skill** is a multi-step procedure that encodes when and how to use tools together to accomplish a goal. Think of it as the difference between "a hammer" (tool) and "how to frame a wall" (skill).

ROW 4B

DATA INTEGRATIONS

real-world data

Connects Hermes to external data sources and third-party platforms that Hermes consumes but does not control — external services it calls via API. Sits on the right half of Row 4, sharing a horizontal band with the Skills & Tool Layer.

> **Why this matters:** Skills & Tool Layer consists of capabilities Hermes controls — the code and logic it owns. Data Integrations are external platforms Hermes calls — it sends a request and receives a response, but has no control over their uptime, pricing, or API changes. If a data integration fails, Hermes can only work around it (e.g., falling back from FMP to Yahoo Finance).

COMPONENTS

FinRobot

equity research pipeline

The primary financial analysis Python framework. The core analytical engine for equity research, producing structured CSV output to `output/{TICKER}/`. Default routing rule: US equities → FinRobot/FMP first.

FMP API

financial endpoints · income ratios

Financial Modeling Prep. Used for income statements, balance sheets, cash flow statements, financial ratios, and historical pricing.

Finnhub

live price quote · USD stock check

Live price quote API. Used as a real-time cross-check against FMP closing prices. Also provides fundamentals, news, and SEC filing alerts.

Yahoo Finance

yfinance · API/Callbacks

Accessed via the `yfinance` Python library. Used as a fallback when FMP is unavailable (HTTP 402) and for Canadian and international equities (RY, TD, BMO).

OpenAI

gpt-4 · news

Direct access to OpenAI models. Used for specific capabilities the primary MiniMax model lacks, such as news summarization or structured output generation.

Subagent Workers

cloud code · coprocess · codecnodes

Subagent processes running specialized models for off-turn processing: cloud code execution, code completion (codecnodes), and watermark summarization.

GitHub

gh CLI + git · on-call PRs

CLI-based GitHub access via `gh CLI` + `git` for PRs, issues, and repo operations.

Google WS

Gmail · GCal · Drive

Google Workspace integration via the `gws` CLI. Covers email, calendar, and document storage.

Airtable + Notion

REST API · records · pages · databases

CRM and knowledge base connections. Airtable for structured records; Notion for documents and knowledge bases.

Key Concept — Tool-to-Tool Chaining: The model calls a tool, receives the result, and immediately uses that result as input to the next tool call — without returning control to the user. This enables fully autonomous multi-step workflows: fetch AAPL price → fetch income statement from FMP → run FinRobot ratios → write CSV → generate HTML report → deliver to Telegram. All in one uninterrupted chain.

ROW 5

DELIVERY & OUTPUT

returning results

Returns the result of the reasoning and tool execution back to the user through the appropriate channel. This is the terminus of every request lifecycle — every task, whether triggered by a live user message or a scheduled cron job, ends with delivery through this row.

> **Why this matters:** Delivery is not a separate processing stage — it is the final step of the tool chain. The model calls `send_message()` or `terminal(write_file)` as its final Act step. If delivery fails, the model can observe the error and retry or reroute.

COMPONENTS

Telegram

home channel · markdown media

Primary delivery channel. Results sent to the user's configured home Telegram channel. Supports markdown formatting and media attachments via `MEDIA:path` syntax. Telegram renders each natively: images as photos, PDFs as documents, audio as voice bubbles.

send_message

platform · chat · thread

General-purpose delivery tool. Used when the destination is not the default Telegram home — e.g., Slack DMs, group channels, or explicit targeting to a specific chat ID and thread.

Cron Auto-deliver

origin / local / all

Scheduled jobs have automatic delivery configured per-job at creation time. `origin` returns to the triggering channel; `local` writes to disk; `all` broadcasts to all connected channels.

Local Output

nfs / overwatch watchdog

Files written to the local output directory. Local output is served from an NFS mount with a watchdog process monitoring for new files. Useful for high-frequency monitoring where agent overhead should be minimized.

Report Artifacts

equity HTML · CSV · charts

Structured outputs generated by FinRobot and related tools: equity research reports in HTML, financial data in CSV, and charts. These are the final deliverable artifacts of financial analysis workflows.

Key Concept — Delivery is a Tool Call: The diagram places Delivery & Output below Skills & Tool Layer and Data Integrations to signal its position as the output terminus of the execution pipeline, fed by connectors from both capability columns. The model calls `send_message()` or `terminal(write_file)` as its final Act step.

ROW 6

MEMORY & PERSISTENCE

remembering

Maintains state across sessions. Without this row, Hermes would be stateless — every conversation starting from scratch, with no memory of who the user is, what they care about, or what has been discussed before. The Memory & Persistence row spans the full width of the diagram as the foundational layer every other row reads from and writes to.

> **Why this matters:** Memory is what differentiates a real assistant from a stateless API. When the user tells Hermes "remember, I prefer concise responses," Hermes updates MEMORY.md and from that point forward, all responses are concise — across sessions, indefinitely. Without this row, every session would require re-establishing context from scratch.

COMPONENTS

MEMORY.md

persistent notes · cross-session

The agent's persistent notes about the user: background, communication style, tool preferences, Finance Stack configuration, and API routing rules. Updated whenever the user asks Hermes to remember something.

USER.md

user profile · loaded at session start

The user's profile: name, role, timezone, communication preferences, and context about ongoing projects. Loaded automatically at session start so Hermes always knows who it is talking to.

SOUL.md

agentic core · identity

The agent's identity definition and behavioral guardrails. Defines Herman's persona, core principles ("genuinely helpful, not performatively helpful"), communication guardrails ("private things stay private"), and boundary conditions.

SessionDB

SQLite · FTS5 · continuity

Written to after every single message in every session. Provides full-text keyword search across all past sessions, message history recall, and a rollback mechanism if the current session needs to recover prior state.

Compaction

summary overflow · rollback space

The process of compressing conversation history when the context window approaches its limit. Older messages are compressed into summaries ("bookends + window"), originals are offloaded to SessionDB, and the loop continues with a trimmed context.

Skills Store

process guard · CRUD on demand

The filesystem-backed skill storage layer. "process guard" is the cross-profile security constraint preventing skills from one profile being invoked in another. Skills are loaded on demand when triggered.

Cron State

job schedules · pause/resume · hostname

The active cron job registry and state. Stores job schedules, delivery configurations, and allows pause/resume without deleting the job definition.

Profiles

HERMES_HOME scoped

Named configurations with isolated skill sets, plugin directories, and cron jobs, each scoped to HERMES_HOME. Allows a single Hermes installation to serve different roles without reconfiguring from scratch.

Agent Config

hermes_agent · adaptive mode · verbose

The core configuration: provider (openrouter), model (MiniMax-H2), context window settings, and runtime parameters from `hermes.yaml` / `.env`.

Key Concept — Multi-Tier Memory: Hermes maintains four distinct tiers: **(1) Working memory** — the context window. **(2) Session memory** — written to SessionDB after every message, FTS5-indexed for search. **(3) Persistent files** — MEMORY.md, USER.md, SOUL.md — written when facts change. **(4) Auxiliary** — Skills Store, Cron State, Profiles, AgentConfig.

End-to-End Walkthrough *AAPL equity report*

User sends: "Run an AAPL equity report" on Telegram

1 **CHANNELS**

Telegram receives the user's message "Run an AAPL equity report" and records source metadata (Telegram, DM, home channel). A normalized request flows down to Route.

2 **ROUTE**

The Channel Router attaches routing metadata: home channel (Telegram, DM), thread target (none — direct DM), delivery scope. The tagged request flows to the Reasoning Loop.

3 **REASONING LOOP**

The Agent Core (Herman / MiniMax-H2) perceives the message, checks context (MEMORY.md, USER.md, session history), and reasons: this matches the equity research skill. It loads the skill, which specifies the sequence: use FinRobot, call FMP endpoints, cross-check with Finnhub, fallback to Yahoo Finance if FMP returns 402. The Perceive → Reason → Act → Observe cycle begins.

4 **SKILLS & TOOLS + DATA INTEGRATIONS**

Tool-to-tool chaining executes the plan:

① Terminal: `generate_financial_analysis.py --ticker AAPL`
② FMP API: fetch income statement, balance sheet, financial ratios
③ Finnhub: live price cross-check
④ yfinance: fallback if FMP unavailable
⑤ File Ops: write CSV to `output/AAPL/`
⑥ Browser: capture chart as PNG
⑦ Terminal: generate HTML report
⑧ send_message: prepare delivery payload

5 **DELIVERY & OUTPUT**

Results delivered:

- Telegram (home channel): `MEDIA:output/AAPL/report.html` — renders as a native document
- Local Output: copy written to local output directory (nfs/overwatch watchdog)
- Cron Auto-deliver: if scheduled, configured delivery targets apply

6 **MEMORY & PERSISTENCE**

Throughout the entire process:

- SOUL.md read at session start — Herman's identity and guardrails established
- MEMORY.md checked — user's preferences and routing rules confirmed
- USER.md checked — user's communication style confirmed
- Every tool result and the final report written to SessionDB (FTS5-searchable)
- If the context window fills, Compaction runs automatically

RESULT

The user receives an AAPL equity report in their Telegram chat, and the session is searchable forever.

Glossary *17 terms defined*

Agentic
A system that autonomously reasons toward goals — decomposing tasks, selecting and chaining tools, and iterating — rather than simply answering questions reactively.

Channel Router
The subsystem that tags incoming requests with delivery metadata — source channel, delivery targets, mobile/desktop dedup guards, thread targeting — before they enter the reasoning loop.

Compaction
The process of compressing conversation history into summaries ("bookends + window") when the context window approaches its limit, writing originals to the SessionDB, and continuing with a trimmed working context.

Context Window
The amount of working memory (in tokens) the model has available for a given request. All conversation history, skill definitions, tool schemas, and current messages compete for this space.

Data Integrations
External third-party services (FMP, Finnhub, Yahoo Finance, Google WS, Airtable, Notion) that Hermes consumes via API — distinct from tools Hermes implements internally.

delegate_task
A tool that spawns an independent subagent process to work on a discrete subtask in parallel. "leaf blueprints: depth 1" means subagents cannot recursively spawn further subagents.

execute_code
The Jupyter kernel tool — a persistent Python REPL that exposes hermes_tools functions as callable tools to the reasoning model.

FTS5
Full-Text Search 5 — a SQLite extension enabling fast keyword search over stored messages in the SessionDB.

HERMES_HOME
An environment variable pointing to the root of the Hermes configuration directory (~/agents/hermes-spike/home/). Skills, profiles, cron jobs, and memory files are all scoped to this directory.

Home Channel
The configured default delivery destination — the messaging channel Hermes routes results to when no explicit target is specified.

Leaf-bounded delegation
A constraint (max_spawn_depth=1) that prevents subagents from spawning further subagents — all delegation stops at depth 1.

OOB Signal
"Out-of-Band" signal — a system-generated interrupt injected mid-conversation to communicate events that need to interrupt the active reasoning loop.

openrouter
The AI gateway used as the primary provider, routing requests to MiniMax-H2. Handles authentication, rate limiting, and model routing.

Perceive → Reason → Act → Observe
The agentic reasoning loop: perceive input and context, reason about the next action, act by calling tools/skills, observe the result, and iterate until the goal is reached.

Skill
A stored, reusable procedural definition — a multi-step playbook the agent loads on demand when its trigger conditions are met. Distinct from a tool (a discrete capability).

SOUL.md

The agent's identity and behavioral boundary file. Defines persona (Herman), core principles, communication guardrails, and explicit boundary conditions.

Tool-to-Tool Chaining

The model calls a tool, receives the result, and immediately uses that result as input to the next tool call — without returning to the user — enabling fully autonomous multi-step workflows.
