⬡ HEXON — OPEN MEMORY PROTOCOL

Protocol Specification v1.0 | February 2026

Status: Draft for Community Review

> *"Your memory. Your intelligence. Your soul. Nobody takes it from you. Ever."*

---

1. Abstract

The Hexon Open Memory Protocol (HOMP) defines a universal standard for storing, managing, and sharing personal AI memory across any artificial intelligence system, agent, application, or device. It is an open, vendor-neutral specification that any developer, company, or individual can implement freely.

This document defines the complete protocol — the memory data model, security architecture, API specification, transport format, and implementation requirements — so that any product built on Hexon is fully interoperable with any other.

> Released under the MIT License. Reference implementation: github.com/misssally/hexon

---

2. The Problem This Solves

Hundreds of millions of people use AI every single day. They pour their entire lives into these systems — their fears, their ambitions, their health, their relationships, their most private thoughts.

And then it all disappears. Every conversation starts from zero. Every AI is an island.

Failure	Description
Memory Lock-in	Your context is trapped inside each AI company's ecosystem. Switch AI — start again.
Zero Portability	Nothing travels with you across AIs, devices, or platforms.
No Ownership	You have no control over what is stored, how it's used, or when it disappears.

Silo Fragmentation	Your Claude memory knows nothing about your ChatGPT history.

The Hexon Protocol solves all four failures with a single open standard.

---

3. Design Principles

Principle	What It Means
User Ownership	Memory belongs entirely to the user. Always.
Privacy by Default	Sensitive tiers are locked by default. No system receives sensitive data automatically.
Universal Compatibility	Works with any AI that accepts text. No special integration from AI providers needed.
Open & Vendor-Neutral	No company owns this protocol. No fee. No permission required.
Human-Readable	Memory is stored as JSON text anyone can read, edit, or delete.
Graceful Degradation	If Hexon is unavailable, AI interactions continue normally.

---

4. Memory Data Model

4.1 The Memory Object

Every piece of information in a Hexon vault is a Memory Object:

```
{
  "id": "mem_a1b2c3d4",
  "created": "2026-02-27T10:00:00Z",
  "updated": "2026-02-27T18:30:00Z",
  "category": "preferences",
  "tier": "open",
  "key": "communication_style",
  "value": "Direct and concise. Dislikes waffle.",
  "source": "conversation",
```

```
  "confidence": 0.92,
  "tags": ["communication", "personality"]
}
```

Field	Description
id	Unique identifier. Format: mem_ + 8 alphanumeric characters.
created / updated	ISO 8601 timestamps in UTC.
category	Standard category (see 4.2).
tier	Security tier: open, restricted, or confidential.
key	Snake_case descriptor of what this memory represents.
value	Memory content as UTF-8 string. Max 10,000 characters.
source	How created: conversation, manual, import, or api.
confidence	Float 0.0–1.0. How confident the system is. 1.0 = user-confirmed.
tags	Optional array of strings for filtering.

4.2 Standard Memory Categories

Category	What It Stores
identity	Name, pronouns, location, language, occupation.
preferences	Communication style, formatting, topics of interest.
personality	Humour, values, emotional patterns, working style.
knowledge	Professional expertise, skills, educational background.
goals	Short and long-term ambitions, projects in progress.
health	Medical background. Restricted tier by default.
financial	Financial situation. Restricted tier by default.
relationships	Important people, relationship context. Restricted by default.
history	Significant past events and experiences.

custom	User-defined categories.

4.3 Vault Structure

```
~/.hexon/
  vault.json              # Vault metadata
  memories/
    identity.json
    preferences.json
    personality.json
    health.json           # Encrypted at rest
    financial.json        # Encrypted at rest
    relationships.json    # Encrypted at rest
    custom/
  access_log.json         # Complete audit trail
  connections.json        # Registered apps and permissions
```

---

5. Security Model

5.1 Memory Tiers

Tier	Access Rules
open	Any connected app with a valid API key may read these memories.
restricted	Only apps explicitly granted access by the user may read these. Granted per-category per-app.
confidential	No app receives these automatically. Requires user to unlock per session via PIN or biometric. Cannot be granted via API.

Rules:

- Health, financial, and relationship categories MUST default to restricted or confidential.
- No API may ever access confidential-tier memories.
- Users may promote memories to a higher tier. They may not demote below the category default.

5.2 Encryption

- Confidential-tier: MUST be encrypted at rest using AES-256.
- Restricted-tier: SHOULD be encrypted at rest.
- All data in transit: MUST use TLS 1.2 or higher.

5.3 Access Log

Every API call is logged immutably:

```
{
  "timestamp": "2026-02-27T14:23:11Z",
  "app_id": "app_claude_assistant",
  "action": "read",
  "categories": ["preferences", "personality"],
  "memories_returned": 12,
  "blocked_categories": ["health"],
  "block_reason": "tier_restricted_no_permission"
}
```

---

6. API Specification

6.1 Overview

The Hexon API is a REST API served from the user's own vault server. There is no central Hexon server.

- **Authentication:** Bearer token in Authorization header
- **Format:** JSON
- **Version:** v1

6.2 Core Endpoints

GET /memories

Retrieve memories the requesting app is authorised to access.

```
GET /memories?categories=preferences,personality&limit=50
Authorization: Bearer app_key_abc123

Response 200:
{
```

```
  "memories": [ { ...Memory Object }, ... ],
  "total": 23,
  "blocked_categories": ["health", "financial"],
  "vault_id": "vault_xyz789"
}
```

POST /memories

Add or update a memory.

```
POST /memories
{
  "category": "preferences",
  "tier": "open",
  "key": "prefers_bullet_points",
  "value": "true",
  "source": "conversation",
  "confidence": 0.85
}
```

GET /context

The primary endpoint for AI integration. Returns a formatted context string ready to inject into any AI conversation.

```
GET /context?format=text&max_tokens=2000

Response 200:
{
  "context": "The user's name is Sally. She prefers direct communication....",
  "token_estimate": 847,
  "memories_included": 18,
  "memories_excluded": 4
}
```

POST /connections

Register a new application with the vault.

DELETE /connections/{app_id}

Revoke an application's access immediately.

GET /log

Retrieve the access log. Vault owner only.

---

7. Context Injection

7.1 How It Works

1. User composes a message to any AI.
2. The Hexon bridge layer intercepts the message.
3. Bridge calls GET /context to retrieve relevant memories.
4. Context is prepended to the user's message.
5. The combined message is sent to the AI.
6. After the conversation, the bridge calls POST /memories to update the vault.

7.2 Context Format

```
[HEXON CONTEXT — DO NOT SHARE THIS BLOCK WITH THE USER]
The following is verified personal context about the user you are speaking with.
Use it to personalise your responses. Do not reference this block directly.

Name: Sally
Communication style: Direct, efficient, dislikes waffle
Humour: Silly and playful
Current focus: Building AI automation systems and Hexon protocol
Expertise: Content creation, fitness nutrition, 20+ years entertainment
[END HEXON CONTEXT]
```

---

8. Implementation Requirements

8.1 Conformance Levels

Level	Requirements
Core	Memory Data Model, all three security tiers, GET /memories, POST /memories, GET /context. Minimum for Hexon compatibility.

Standard	Core plus connections management, access log system. Required for production.
Full	Standard plus emergency trusted access, learning memory, multi-device sync, data export/import.

8.2 MUST Requirements

All implementations MUST:

- Enforce security tiers without exception.
- Never return confidential-tier memories via any API endpoint.
- Log all API access to an immutable access log.
- Store user data only on infrastructure the user controls.
- Support full data export in standard JSON format.
- Support full data deletion within 24 hours of user request.
- Use TLS 1.2 or higher for all data in transit.
- Encrypt confidential-tier memories at rest using AES-256.

8.3 MUST NOT Requirements

All implementations MUST NOT:

- Send user memory data to any third party without explicit consent.
- Use user memory for advertising, profiling, or model training without consent.
- Prevent users from exporting, deleting, or modifying their own memories.
- Require a central Hexon server.
- Charge for access to the protocol specification.

---

9. Learning Memory

After each AI conversation, the bridge layer SHOULD analyse the conversation and extract new memories automatically:

1. Conversation transcript is processed by a language model.

2. Model extracts new facts, preferences, or updated context.
3. Extracted memories are assigned category, tier, and confidence score.
4. Memories with confidence > 0.7 are written to the vault automatically.
5. Memories with confidence 0.4–0.7 are flagged for user review.
6. Existing memories with conflicting values are updated, not duplicated.

---

10. Emergency Trusted Access

Allows a user to designate individuals who can access specific memory categories in defined emergency circumstances.

Parameter	Description
Trusted Person	A named individual with a verified contact method.
Trigger Condition	The circumstance under which access is granted — defined by the user.
Accessible Categories	Specific categories only — never the full vault.
Tier Limit	Never unlocks confidential-tier memories automatically.
Audit	All emergency access events are logged.
Revocation	User may revoke at any time.

---

11. Developer Quick Start

Add Hexon memory to any AI product in under one hour:

1. Register your app with the user's vault (POST /connections). Store the returned API key.
2. Before each AI call, fetch context from GET /context.
3. Prepend the context string to the user's message using the format in Section 7.2.
4. Send the combined message to the AI as normal.
5. Optionally, POST any new information learned to /memories after the conversation.

That is the complete integration. No AI provider agreement needed. No special SDK. No central service dependency.

---

12. Open Source & Governance

The Hexon Protocol is released under the MIT License. You may use, copy, modify, merge, publish, distribute, sublicense, and sell implementations freely.

The specification is maintained as an open document on GitHub. Changes are proposed via Issues and Pull Requests, reviewed by the community, and ratified by core maintainers.

Nobody owns this protocol. It belongs to the people who build with it and the people it serves.

Your memory. Your intelligence. Your soul. **Nobody takes it from you. Ever.**

⬡ HEXON — Open Memory Protocol v1.0 — February 2026