Abstract Agentic System Architecture Design v1.1

System: Chronoktonos

- **Document Version**: 1.1
- **Status**: Proposed Revision
- **Date**: 2025-07-20

1. The Agent (Node)

The fundamental building block of the system. Each Agent is an instance of a generic class, its specific role defined by its configuration and internal logic.

1.1. Core Attributes

- **agent_id**: A unique identifier (e.g., "Puppetmaster", "SystemArchitect_v3"). Serves as its network address.
- **input_dir (Mailbox)**: A dedicated, unique file system directory for receiving messages.
- **processed_dir**: A directory for archiving messages after successful processing.

1.2. Core Capabilities

- **Active Listening**: Continuously monitors its input_dir for new message files.
- **Internal Logic (process_input Method)**: The agent's "brain." Executes logic based on message content.
- **Communication (_send_message Method)**: Can send messages to any other known agent by writing to their mailbox.
- **Status Reporting (Heartbeat)**: Periodically updates its status in the Central Mailbox Registry.

1.3. Agent Lifecycle (New in v1.1)

An agent exists in one of the following states, reflected in the registry:

- **spawning**: The agent is initializing but not yet ready to process messages.
- **active**: The agent is running and processing messages.
- **error**: The agent has encountered a critical, unrecoverable error and has ceased operation.
- **inactive**: The agent is not running or is intentionally paused.
- **terminated**: The agent has been gracefully shut down.

2. The Central Mailbox Registry

A globally accessible mechanism mapping agent_id to mailbox paths and operational status.

2.1. Implementation

- **mailbox_map.json**: A JSON file in a central directory (e.g., /var/lib/agent_mailboxes/mailbox_map.json).
- **mailbox_map.lock (New in v1.1)**: A lock file that must be acquired before any write operation to mailbox_map.json and released immediately after. This prevents data corruption from concurrent writes. An agent attempting to write must wait if the lock file exists.

2.2. Content Structure

Each entry in the map is an object containing agent details:

```
"SystemArchitect_v3": {
  "mailbox_path": "/var/lib/agent_mailboxes/SystemArchitect_v3_in",
  "status": "active",
  "last_heartbeat": "2025-07-20T18:30:00Z",
  "last_processed_task_id": "xyz789",
  "description": "Designs and refines the system architecture."
}
```

3. The Message (Packet)

The standardized JSON file for information exchange.

3.1. Structure (Refined in v1.1)

- **Required Fields**:
  - message_id: A new, unique UUID for the message itself.
  - sender_id: The agent_id of the originator.
  - recipient_id: The agent_id of the intended receiver.
  - timestamp_utc: ISO 8601 timestamp of message creation.
  - type: The message's purpose (e.g., "request", "result", "directive").
  - payload: The data content of the message.
- **Optional Fields**:
  - task_id: A unique identifier to track a specific task or conversational thread.
  - priority: A numerical priority (e.g., 1-5) for agents that support it.

3.2. File Naming Convention

[timestamp]_[message_id].json (e.g., 20250720183000Z_uuid-v4-string.json). The sender and task are now inside the file, simplifying the filename.

4. Security & Access Control (New in v1.1)

The file-system-based architecture requires strict permissioning.

- **Directory Permissions**: Each agent's input_dir should only be writable by the agents authorized to send it messages. By default, only the agent itself and the Puppetmaster should have write access.
- **Message Integrity**: While not yet implemented, future versions should consider payload checksums or signatures to verify message integrity and sender authenticity.

5. Integration of Human Agents

Human participants are integrated as Agent instances, using their mailbox to send and receive messages manually or via simple tooling.

6. Abstract Principles

- **Decentralized Processing, Centralized Discovery**: Agents operate independently but find each other via the shared registry.
- **Asynchronous & Message-Driven**: Communication is non-blocking, via messages.
- **Plug-and-Play Extensibility**: New agents can be added by creating mailboxes and registering them.