29 APR 25 // Agent, Task, Vault, Atlas, Stem

Created	@April 29, 2025 6:40 AM
AI summary	Agents represent roles in health plans, executing tasks to achieve business outcomes. Tasks are core capabilities for agents, while Vault and Atlas serve as interfaces for data storage and configuration. Stems group events and maintain runtime context, with FloPilot acting as a viewer and chatbot to manage tasks and sessions.

Notes on our product ontology from an engineering perspective.

Agents

(Previously: workflows / workflos)

An agent represents a role within the health plan, that might be given to a human as a job title. For example, “Prior Auth MNR review” or “Prior Auth intake preparation”.

Agents are created by the health plan for their use, from templates provided by Anterior. The health plan will be able to see a list of their active agents.

Agents are defined by their configuration, including:

- A list of tasks available to the agent
  - E.g. “Validate clinicals”, “Select guideline”, “Perform MNR”
- The business outcomes that the agent is driving towards
  - E.g. "Approve MNR", "Escalate MNR"
- Triggers for the agent:
  - E.g. "Vault query", "API", "FloPilot"

Agents will execute the tasks to arrive at one of the business outcomes. In code, we will use different implementations of the agent. For example, in our current MNR agent, we hardcode a graph of task execution.

We are calling these "agents" because:

- Deterministic execution is an implementation detail
- We are leaving ourselves open to an agentic implementation
- "Agent" as a term of art makes product sense

Tasks

(Previously: Clinical Tasks)

Tasks are Anterior's core set of Lego block primitive capabilities available to the health plan to use for agents and via FloPilot.

"Tasks" in the product sense are not brrr tasks (individual functions). They are perhaps more analogous to "tools" in the agentic sense. But there are key differences. Tasks are discrete units of work that a human would do in order to complete a job. They are at a higher level of abstraction than agentic tools. Task descriptions do not point at their inner implementation.

For example, "Validate clinicals", "Perform MNR". We will iterate on the implementation of these tasks, and also the inputs provided to them.

Tasks share a common interface:

Input:

- Vault
- Atlas
- Stem
- Event emitter

Output:

- Enum

The interface is purposely broad. This allows for a broad range of implementations and use cases. We will try to make the tasks increasingly generic and composable.

An initial implementation of "Perform MNR" might assume a GDT representation of a guideline is available to it in the Atlas. If not available, it will fail fast. Over time, we will make the implementation more robust, and able to deal with text representations, or IGGs or whatever else we might come up with.

Vault

Vault is an interface provided to the task via its agent (or via FloPilot) for fetching and storing clinical data.

You can think of the Vault and Atlas as async stores. They are interfaces over S3 (and other future storage, abstracted away).

Tasks will look in the Vault provided to them for clinical data, following IDs in the Stem. When tasks create clinical data, they will store it in their Vault and attach events to the Stem.

Atlas

Atlas is an interface provided to the task via its agent (or via FloPilot) for fetching enterprise configuration.

Atlas (and Vault) are usually workspace specific. There will also be enterprise specific and anterior specific stores, with an inheritance relationship.

Tasks will look in the Atlas provided to them for configuration data and objects. For example, the Atlas records the enterprise's configured guideline provider and can return guidelines in a specific format.

Stem

A Stem is a list of events. The Stem ID is the key for grouping events into a single "session". Stems are the runtime context for work. Stems are the state storage for the system.

A FloPilot chat session is a stem. An Agent run is a stem. If the user runs multiple agents from a single FloPilot chat session, they will share a stem.

As work is done, a stem will accumulate events. Those events will describe artifacts -- objects that have been stored in the Vault for example. You can then query stems for artifacts.

If a task needs a Requested Service, a previous task should have added a reference to a Requested Service to the stem and stored the data in the Vault. The task can then look in its provided stem for a reference to a Requested Service and proceed (or fail if not found).

FloPilot

FloPilot can be thought of as a "Stem Viewer". A single FloPilot session is a list of the events in a single stem.

To the user, FloPilot also provides a chat interface to add messages to the Stem. FloPilot is also a chatbot agent that reacts to those messages, schedules its available Agents and Tasks, passing them the stem to add work to it.

The artifacts in a Stem can be rendered in the UI. Rather than listing a series of events, FloPilot will render the events and artifacts they point to as UI components.