Overview

The workflow is designed according to the following principles:

1. **Manager** - a workflow that handles state files, specifically:
    1. Reads the list of state files from the `/states` folder.
    2. Sorts them by status.
    3. Initiates workers.
    4. Reads and copies the statuses of the workers.
    5. Updates the information in the state files.
2. **Worker** - a workflow that:
    1. Executes functional tasks.
    2. Reads, saves, and updates its own state file.
    3. Receives a payload from the manager in a specific format that helps customize the input data.

This principle is required to customize the list of repetitive steps in Finmars using JSON files (and provides the ability to execute this customization through an interface, which is currently not implemented), as well as to use the interface to analyze completed steps, identify errors, and restart steps.

How It Works

The workflow manager (`com.finmars.standard-workflow:workflow-manager`) operates on a cron schedule (by default - every 1 minute) and performs the following actions:

1. Reads the `global_state_manager.json` file.
    1. If the file does not exist, it creates the file and adds files from `/states/managers` according to the following format:

```
{
    "to-do": [],
    "in-progress": [
        "com.finmars.standard-workflow:workflow-manager-20240821095322.json"
    ],
    "done": [
        "com.finmars.standard-workflow:workflow-manager-20240724101635.json",
        "com.finmars.standard-workflow:workflow-manager-20240724104102.json"
    ],
```

```
    "paused": []
}
```

2. Selects the first file from the in-progress array and begins managing it.

- Potentially, this could be improved by running multiple threads, but there are certain limitations since these state files are independent of each other (see the input file requirements).

3. The manager goes through the workers and checks their statuses:

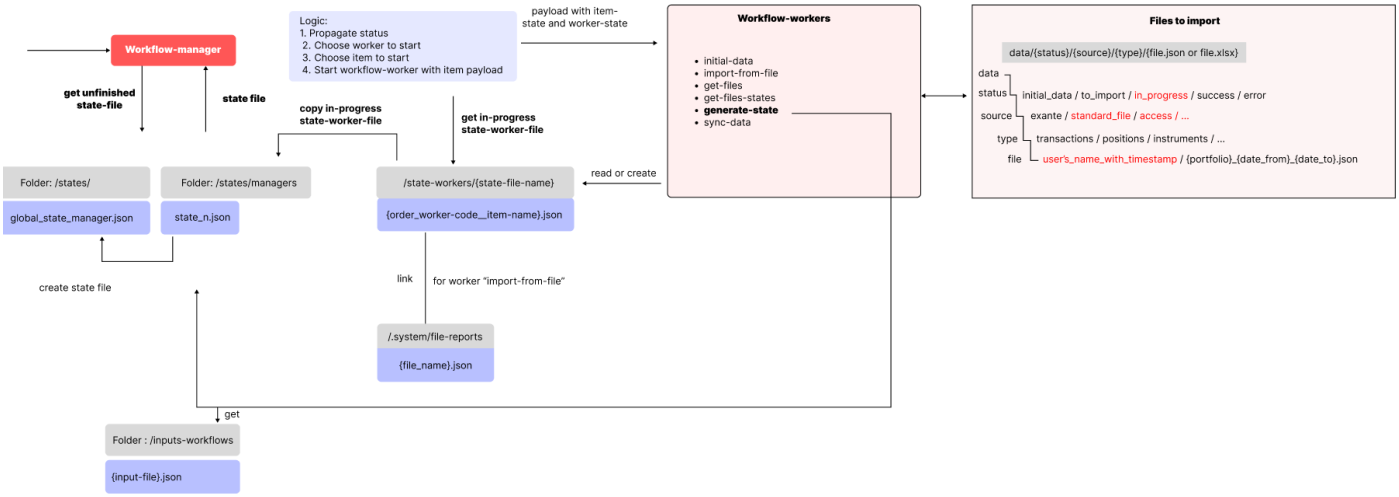
- `to-do` - launches the workflow worker for a specific item
- `in-progress` - checks the status of items:
- `error`, `skip`, `success` - skips to next worker

4. The manager uses the propagate status logic, where:

- `success` - will propagate to the entire worker if all items have this status
- `error` - will propagate the status from the item to the worker level

The `manager` does not wait for the `worker` to complete and finishes after starting the `worker`.

Scheme of process



Detailed description provided in Finmars University

Revision #4
Created 2024-08-29 20:55:41 UTC by Anton Petrushkaneki
Updated 2024-08-30 15:00:37 UTC by Anton Petrushkaneki