

Workflow Manager

Overview

User Code: `com.finmars.standard-workflow:workflow-manager`

Payload: Optional

The Workflow Manager is designed to execute workers in a specific order with a defined payload. Its primary use case is when no payload is provided.

Global State Manager Processing Logic

1. Attempt to read the `global_state_manager.json` file. If it doesn't exist, create it.
2. Read the list of files from `/states/managers/`.
3. Categorize by status and save in `global_state_manager.json`:
 - If the file doesn't exist, add it to "to-do"
 - If it's in "to-do", change the status to "in-progress"
 - If the file exists, update its status
4. Read the contents of the file in "in-progress" status

```
{
  "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": []
}
```

Interaction with global_state_manager.json

The `global_state_manager.json` file serves as a central registry for all workflow states in the system:

1. **Initialization:** If `global_state_manager.json` doesn't exist, it's created by scanning the `/states/managers/` directory and categorizing all existing state manager files.
2. **Retrieval:** The existing `global_state_manager.json` is loaded and its data is updated.
3. **Updating:** The global state manager is updated by:
 - Identifying and adding new state manager files to the "in-progress" category
 - Checking and updating the status of each tracked state manager file
 - Moving state managers between categories based on their current status
4. **Saving:** After updates, the modified `global_state_manager.json` is saved back to storage.
5. **Workflow Processing:** The main `workflow_manager` function uses data from `global_state_manager.json` to determine which state managers to process when no specific payload is provided.

State Managers and Workers/Items Interaction

Each state manager file represents an individual workflow instance, containing information about its workers and their respective items (tasks):

1. **Loading:** Individual state manager files are loaded based on paths stored in `global_state_manager.json` or provided in the payload.
2. **Status Propagation:** Each worker's status is updated based on the statuses of its items.
3. **Worker Processing:** For each worker in a state manager:
 - Skipped if status is 'success', 'skip', or 'ignore'
 - If 'in-progress', each item's status is checked:
 - 'in-progress' items: latest state is fetched and updated
 - 'to-do' items: a new workflow is initiated
 - If 'to-do', processing starts with the first 'to-do' item
4. **Item State Management:** Each item within a worker has its own state, including a `state_path` used to fetch and update item-specific states.
5. **Workflow Initiation:** New items are processed by calling the `start_workflow` function with the appropriate user code and payload.
6. **State Updating:** After any changes, the modified state manager is saved back to its file.

