TaskQueueSwarm Documentation

The `TaskQueueSwarm` class is designed to manage and execute tasks using multiple agents concurrently. This class allows for the orchestration of multiple agents processing tasks from a shared queue, facilitating complex workflows where tasks can be distributed and processed in parallel by different agents.

Attributes

Attribute	Type	Description
`agents`	`List[Agent]`	The list of agents in the swarm.
`task_queue`	`queue.Queue`	A queue to store tasks for processing.
`lock`	`threading.Lock`	A lock for thread synchronization.
`autosave_on`	`bool`	Whether to automatically save the swarm metadata.
`save_file_path`	`str`	The file path for saving swarm metadata.
`workspace_dir`	`str`	The directory path of the workspace.
`return_metadata_on`	`bool`	Whether to return the swarm metadata after running.
`max_loops`	`int`	The maximum number of loops to run the swarm.
`metadata`	`SwarmRunMetadata`	Metadata about the swarm run.

Methods

`__init__(self, agents: List[Agent], name: str = "Task-Queue-Swarm", description: str = "A swarm that processes tasks from a queue using multiple agents on different threads.", autosave_on: bool = True, save_file_path: str = "swarm_run_metadata.json", workspace_dir: str =

os.getenv("WORKSPACE_DIR"), return_metadata_on: bool = False, max_loops: int = 1, *args, **kwargs)`

The constructor initializes the `TaskQueueSwarm` object.

- **Parameters:**
  - `agents` (`List[Agent]`): The list of agents in the swarm.
  - `name` (`str`, optional): The name of the swarm. Defaults to "Task-Queue-Swarm".
  - `description` (`str`, optional): The description of the swarm. Defaults to "A swarm that processes tasks from a queue using multiple agents on different threads.".
  - `autosave_on` (`bool`, optional): Whether to automatically save the swarm metadata. Defaults to True.
    - `save_file_path` (`str`, optional): The file path to save the swarm metadata. Defaults to "swarm_run_metadata.json".
    - `workspace_dir` (`str`, optional): The directory path of the workspace. Defaults to os.getenv("WORKSPACE_DIR").
  - `return_metadata_on` (`bool`, optional): Whether to return the swarm metadata after running. Defaults to False.
  - `max_loops` (`int`, optional): The maximum number of loops to run the swarm. Defaults to 1.
  - `*args`: Variable length argument list.
  - `**kwargs`: Arbitrary keyword arguments.

`add_task(self, task: str)`

Adds a task to the queue.

- **Parameters:**
  - `task` (`str`): The task to be added to the queue.

`run(self)`

Runs the swarm by having agents pick up tasks from the queue.

- **Returns:**
  - `str`: JSON string of the swarm run metadata if `return_metadata_on` is True.

- **Usage Example:**
  ```python
  from swarms import Agent, TaskQueueSwarm
  from swarms_models import OpenAIChat

  # Initialize the language model
  llm = OpenAIChat()

  # Initialize agents
  agent1 = Agent(agent_name="Agent1", llm=llm)
  agent2 = Agent(agent_name="Agent2", llm=llm)

  # Create the TaskQueueSwarm
  swarm = TaskQueueSwarm(agents=[agent1, agent2], max_loops=5)

  # Add tasks to the swarm
  ```

```
swarm.add_task("Analyze the latest market trends")
swarm.add_task("Generate a summary report")

# Run the swarm
result = swarm.run()
print(result)  # Prints the swarm run metadata
```

This example initializes a `TaskQueueSwarm` with two agents, adds tasks to the queue, and runs the swarm.

`save_json_to_file(self)`

Saves the swarm run metadata to a JSON file.

`export_metadata(self)`

Exports the swarm run metadata as a JSON string.

- **Returns:**
  - `str`: JSON string of the swarm run metadata.

Additional Notes

- The `TaskQueueSwarm` uses threading to process tasks concurrently, which can significantly improve performance for I/O-bound tasks.

- The `reliability_checks` method ensures that the swarm is properly configured before running.
- The swarm automatically handles task distribution among agents and provides detailed metadata about the run.
- Error handling and logging are implemented to track the execution flow and capture any issues during task processing.