# AutoSwarm

The `AutoSwarm` class represents a swarm of agents that can be created and managed automatically. This class leverages the `AutoSwarmRouter` to route tasks to appropriate swarms and supports custom preprocessing, routing, and postprocessing of tasks. It is designed to handle complex workflows efficiently.

### Key Concepts

- **Swarm**: A group of agents working together to complete tasks.
- **Routing**: Directing tasks to the appropriate swarm based on specific criteria.
- **Preprocessing and Postprocessing**: Customizable functions to handle tasks before and after routing.
- **Event Loop**: Managing the execution of tasks in a loop.

## Attributes

### Arguments

| Argument | Type | Default | Description |
|---------------------|-------------------------------|-----------|-------------|
| `name` | `Optional[str]` | `None` | The name of the swarm. |
| `description` | `Optional[str]` | `None` | The description of the swarm. |
| `verbose` | `bool` | `False` | Whether to enable verbose mode. |
| `custom_params` | `Optional[Dict[str, Any]]` | `None` | Custom parameters for the swarm. |
| `custom_preprocess` | `Optional[Callable]` | `None` | Custom preprocessing function for

tasks. |

| | | | |
|---|---|---|---|
| `custom_postprocess` | `Optional[Callable]` | `None` | Custom postprocessing function for task results. |
| `custom_router` | `Optional[Callable]` | `None` | Custom routing function for tasks. |
| `max_loops` | `int` | `1` | The maximum number of loops to run the workflow. |

### Attributes

| Attribute | Type | Description |
|---|---|---|
| `name` | `Optional[str]` | The name of the swarm. |
| `description` | `Optional[str]` | The description of the swarm. |
| `verbose` | `bool` | Whether to enable verbose mode. |
| `custom_params` | `Optional[Dict[str, Any]]` | Custom parameters for the swarm. |
| `custom_preprocess` | `Optional[Callable]` | Custom preprocessing function for tasks. |
| `custom_postprocess` | `Optional[Callable]` | Custom postprocessing function for task results. |
| `custom_router` | `Optional[Callable]` | Custom routing function for tasks. |
| `max_loops` | `int` | The maximum number of loops to run the workflow. |
| `router` | `AutoSwarmRouter` | The router for managing task routing. |

## Methods

### init_logging

Initializes logging for the `AutoSwarm`.

**Examples:**

```python
swarm = AutoSwarm(name="example_swarm", verbose=True)
swarm.init_logging()
```

### run

Runs the swarm simulation.

**Arguments:**

| Parameter | Type | Default | Description |
|-----------|---------|---------|-------------|
| `task` | `str` | `None` | The task to be executed. |
| `*args` | | | Additional arguments. |
| `**kwargs` | | | Additional keyword arguments. |

**Returns:**

| Return Type | Description |
|-------------|-------------|
| `Any` | The result of the executed task. |

**Raises:**

- `Exception`: If any error occurs during task execution.

**Examples:**

```python
swarm = AutoSwarm(name="example_swarm", max_loops=3)
result = swarm.run(task="example_task")
print(result)
```

### list_all_swarms

Lists all available swarms and their descriptions.

**Examples:**

```python
swarm = AutoSwarm(name="example_swarm", max_loops=3)
swarm.list_all_swarms()
# Output:
# INFO: Swarm Name: swarm1 || Swarm Description: Description of swarm1
# INFO: Swarm Name: swarm2 || Swarm Description: Description of swarm2
```

### Additional Examples

#### Example 1: Custom Preprocessing and Postprocessing

```python
def custom_preprocess(task, *args, **kwargs):
    # Custom preprocessing logic
    task = task.upper()
    return task, args, kwargs

def custom_postprocess(result):
    # Custom postprocessing logic
    return result.lower()

swarm = AutoSwarm(
    name="example_swarm",
    custom_preprocess=custom_preprocess,
    custom_postprocess=custom_postprocess,
    max_loops=3
)

# Running a task with custom preprocessing and postprocessing
result = swarm.run(task="example_task")
print(result)  # Output will be the processed result
```

#### Example 2: Custom Router Function

```python
def custom_router(swarm, task, *args, **kwargs):
    # Custom routing logic
    if "specific" in task:
        return swarm.router.swarm_dict["specific_swarm"].run(task, *args, **kwargs)
    return swarm.router.swarm_dict["default_swarm"].run(task, *args, **kwargs)

swarm = AutoSwarm(
    name="example_swarm",
    custom_router=custom_router,
    max_loops=3
)

# Running a task with custom routing
result = swarm.run(task="specific_task")
print(result)  # Output will be the result of the routed task
```

#### Example 3: Verbose Mode

```python
swarm = AutoSwarm(
    name="example_swarm",
    verbose=True,
    max_loops=3
)

# Running a task with verbose mode enabled
result = swarm.run(task="example_task")
# Output will include detailed logs of the task execution process
```

#### Full Example 4:

First create a class with BaseSwarm -> Then wrap it in the router -> then pass that to the `AutoSwarm`

```python
from swarms import BaseSwarm, AutoSwarmRouter, AutoSwarm



class FinancialReportSummarization(BaseSwarm):
    def __init__(self, name: str = None, *args, **kwargs):
        super().__init__()

    def run(self, task, *args, **kwargs):
        return task
```

```
# Add swarm to router
router = AutoSwarmRouter(swarms=[FinancialReportSummarization])


# Create AutoSwarm Instance
autoswarm = AutoSwarm(
    name="kyegomez/FinancialReportSummarization",
    description="A swarm for financial document summarizing and generation",
    verbose=True,
    router=router,
)


# Run the AutoSwarm
autoswarm.run("Analyze these documents and give me a summary:")
```

## Summary

The `AutoSwarm` class provides a robust framework for managing and executing tasks using a swarm of agents. With customizable preprocessing, routing, and postprocessing functions, it is highly adaptable to various workflows and can handle complex task execution scenarios efficiently. The integration with `AutoSwarmRouter` enhances its flexibility, making it a powerful tool for dynamic task management.