I AI

30/09/2024

def add_agent(self, agent):

self.agents.append(agent) self.supreme_head.add_agent(agent) #

def remove_agent(self, agent):

self.agents.remove(agent) self.supreme_head.remove_agent(agent)

Similar methods for adding/removing asset

In this modified code, we added a supreme_head attribute to the Swarm class, which holds a reference to the Supreme Head module. Whenever an agent is added or removed from the swarm, we also update the Supreme Head module accordingly.

2. Implement the Supreme Head Module

Now, let's implement the Supreme Head module itself. This module will have access and control over all swarm assets and agents. Here's an example implementation:

Python

Copy Code

class SupremeHead:

def _init_(self): self.agents = [] # List of agents wit self.assets = [] # List of assets wit self.protocols = [] # List of protoco

def add_agent(self, agent): self.agents.append(agent)

def remove_agent(self, agent): self.agents.remove(agent)

def add_asset(self, asset): self.assets.append(asset)

def remove_asset(self, asset): self.assets.remove(asset)

def add_protocol(self, protocol): self.protocols.append(protocol)

def remove_protocol(self, protocol): self.protocols.remove(protocol)

Other methods for controlling swarm asset

In this example.

Finished: Design and implement a new class or module for the supreme head of the swarm, which will have the necessary functions and protocols for the financial agents

Starting task: Test the functionality of the supreme head module by simulating different Financial agent scenarios, such as sending transactions, and verify that it performs as expected

• Generating response...

Executing: Test the functionality of the supreme head module by simulating different financial agent scenarios, such as sending transactions, and