Agents running on agentverse

Agents running on agentverse act as functions as a service, they do this by having an orchestrator in the background that calls functions on the agennt when a routine should execute, or a message is called. This helps us keep hosted agents costs down which are passed on to the user.

However, you should be aware of how state works in a hosted agent. A global variables in agentverse will always have the value of the initalised value you set. Updating this value from a function will not work outside the scope of that function call, as the variable is set when the agent is loaded into the Orchestrator which does not load up the entire agent for each call, a global variable will always have the initial set value. To have values persist you must store this values in storage.

Let's show this with an example:

from uagents import Agent , Context

agent

Agent ()

global_counter

0

@agent . on_interval (period = 1.0) async

def

on_interval (ctx : Context): global global_counter

increment the global counter - bad!

global_counter +=

1

current_count

int (ctx.storage. get ("count") or

0) current_count +=

1

ctx . logger . info (f "My count is: { current_count } ") ctx . logger . info (f "My global is: { global_counter } ")

ctx . storage . set ("count" , current_count)

if

name

==

"**main**" : agent . run () Output in agentverse is:

16:46:19 Info Agent [INFO]: My global is: 1 16:46:21 Info Agent [INFO]: My count is: 452 16:46:21 Info Agent [INFO]: My global is: 1 16:46:23 Info Agent [INFO]: My count is: 453 16:46:23 Info Agent [INFO]: My global is: 1 You can see from this output, global is never updated.

To learn more about storage in agentverse, take a look at Storage Functions

Was this page helpful?

Options for running local agents Introducing dialogues