Agents address

Introduction

Each agent within the Fetch ecosystem is characterized by different addresses. These can allow the agent to perform different actions, including sending messages or interacting with theAlmanac contract ↗ .

It is possible to distinguish between two different types of addresses:

- uAgent address
- : this is the main agent identifier. Other agents can use this address to query the agent's information in the Almanac contract.
- Fetch network address
- : this is the address providing the agent with the capabilities for interacting with the Fetch Ledger ↗
- , includingRegistration in the Almanac ↗
- contract.

If you want to retrieve the address of your agent, you can either use theprint() function and specify which of the above addresses you wish to print out, or by calling theContext class and related methods to retrieve specific information.

Let's now check how these ways of retrieving addresses look like!

Print uAgent address

You can print theuAgent address related to your agent in the following way:

1. First of all, create a Python script and name it by running:touch uagent-address.py
2. We then need to import theAgent
3. class from theuagents
4. library to create an agent,alice
5. . Then, using theprint
6. function, we will print the relateduAgent address
7. . Importantly, remember that theseed
8. parameter is used, when creating an agent, to set fixed addresses, otherwise a random address will be generated every time you run the agent:
9. uagent-address.py
10. from
11. uagents
12. import
13. Agent
14. agent
15. =
16. Agent
17. (name
18. =
19. "alice"
20. , seed
21. =
22. "alice recovery phrase"
23.)
24. print
25. (
26. "uAgent address: "
27. , agent.address)
28. if
29. **name**
30. ==
31. "**main**"
32. :
33. agent
34. .
35. run
36. ()
37. Save the script.

The output would be as follows:

uAgent address: agent1qww3ju3h6kfcuqf54gkghvt2pqe8qp97a7nzm2vp8plfxflc0epzcjsv79t

Print Fetch network address

You can print theFetch network address related to your agent in the following way:

1. Let's create a Python script, and name it by running:touch fetch-address.py
2. As before, we first need to import theAgent
3. class from theuagents
4. library to create a uAgent,alice
5. . Then, using theprint()
6. function, we will print the relatedFetch Network address
7. :
8. fetch-address.py
9. from
10. uagents
11. import
12. Agent
13. agent
14. =
15. Agent
16. (name
17. =
18. "alice"
19. , seed
20. =
21. "alice recovery phrase"
22.)
23. print
24. (
25. "Fetch network address: "
26. , agent.wallet.
27. address
28. ())
29. if
30. **name**
31. ==
32. "**main**"
33. :
34. agent
35. .
36. run
37. ()
38. Save the script.

The output would be as follows:

Fetch network address: fetch1454hu0n9eszzg8p7mvan3ep7484jxl5mkf9phg

Print agent name and address usingname

andaddress methods

In this guide, we aim at showing how to create an agent being able to say hello and printing itsname andaddress retrieving such information from theContext class imported from theuagents library.

TheContext class is a crucial component which represents the execution context of an agent. It encapsulates different attributes and methods which allow an agent to interact with its environment, send and receive messages, and manage its state and identity. Within this class, we can distinguish multiple attributes and methods, including:

- name
- : which returns the provided name of the agent, if specified, otherwise, if the agent's name is not explicitly set, then it will use the first ten characters of the agent's address as its name.
- address
- : which returns the unique address of the agent in the formagent1...
- . This address is essential for other agents to interact with your agent.

Let's get started and use theContext class to make our agent print its name and address!

Walk-through

1. First of all, you need to create a Python script and name it by running:touch my_agent.py
2. We then need to import the necessary classesAgent
3. andContext
4. from theuagents
5. library, and then create an instance of theAgent
6. class,alice
7. :
8. from
9. uagents
10. import
11. Agent
12. ,
13. Context
14. agent
15. =
16. Agent
17. (name
18. =
19. "alice"
20. , seed
21. =
22. "alice recovery phrase"
23.)
24. We would then need to assign the agent the behavior to be executed. In this case,agent
25. could send a message when it is being run saying hello and printing itsname
26. andaddress
27. :
28. @agent
29. .
30. on_event
31. (
32. "startup"
33.)
34. async
35. def
36. introduce_agent
37. (
38. ctx
39. :
40. Context):
41. ctx
42. .
43. logger
44. .
45. info
46. (
47. f
48. "Hello, I'm agent
49. {
50. agent.name
51. }
52. and my address is
53. {
54. agent.address
55. }
56. ."
57.)
58. if
59. **name**
60. ==
61. "**main**"
62. :
63. agent
64. .
65. run
66. ()

67. Thisintroduce_agent()
68. function takes a single argumentctx
69. of typeContext
70. . The message is printed out using thectx.logger.info()
71. method, and includes the agent's name obtained from attributename
72. and retrieved usingagent.name()
73. method. The same for the agent's address, which is obtained from attributeaddress
74. and retrieved usingagent.address()
75. method.
76. Save the script.

The overall script should look as follows:

my_agent.py from uagents import Agent , Context

agent

Agent (name = "alice" , seed = "alice recovery phrase")

@agent . on_event ("startup") async

def

introduce_agent (ctx : Context): ctx . logger . info (f "Hello, I'm agent { agent.name } and my address is { agent.address } .")

if

name

==

"**main**" : agent . run ()

Run the script

On your terminal, make sure to have activated the virtual environment.

Run the script:my_agent.py

The output should be as follows:

Hello, I'm agent alice and my address is agent1qww3ju3h6kfcuqf54gkghvt2pqe8qp97a7nzm2vp8plfxflc0epzcjsv79t.

Was this page helpful?

Create your first agent Communicating with other agents