# Agents name service

## Introduction

Aname service , in the context of computer networks and distributed systems, is a system which associates meaningful names with network resources. This is carried out to make it easier for humans to identify and access these resources rather than using numerical IP addresses. TheDomain Name System (DNS ) is a widely used name service on the internet. The DNS can be thought as the internet's phonebook. Domain names (for instance, example.com), are used by humans to access content online. Web browsers communicate using Internet Protocol (IP) addresses. DNS converts domain names to IP addresses so that browsers may access Internet resources. Each Internet-connected device has a unique IP address that other machines may use to locate that specific device. DNS servers remove the need for people to remember IP addresses by heart.

In the context of Fetch.ai's AI Agents, we have implemented a name service for agents. to allow for quicker identification thanks to human-readable names instead of other identifiers and address. For a better understanding of this, we will create two agents,alice anbob , register them in theAlmanac ↗ , but with agentbob being given a domain so to be easily found by any other agent.

## Walk-through

For this example guide, we will make use of theuagents andcosmpy libraries. For further understanding and resources visit our documentation for either theuAgents Framework ↗ andCosmPy ↗ , as well asGuides ↗ andReferences ↗ sections.

The first step we need to carry out is creating a dedicated directory for this task. Let's call this directoryname_service . You can do so by running the following command:mkdir name_service .

Within this directory, we need to create two separate scripts, one for each agent we will develop:

- Bob:touch agent_1.py
- Alice:touch agent_2.py

Let's get started!

### Agent 1: bob

We start by defining the script for our agentbob .

1. First of all, let's import all needed classes fromcosmpy.aerial.wallet
2. ,uagents.network
3. ,uagents.setup
4. , anduagents
5. . Then, we define a message data model for types of messages to be exchanged between our agents:
6. from
7. cosmpy
8. .
9. aerial
10. .
11. wallet
12. import
13. LocalWallet
14. from
15. uagents
16. .
17. network
18. import
19. get_name_service_contract
20. from
21. uagents
22. .
23. setup
24. import
25. fund_agent_if_low
26. from
27. uagents
28. import
29. Agent
30. ,

31. Context
32. ,
33. Model
34. class
35. Message
36. (
37. Model
38. ):
39. message
40. :
41. str
42. We then need to initialize our agentbob
43. andregister it in the Almanac ↗
44. . We also initialize a wallet by ensuring it has enough funds in it, and then providebob
45. with a domain:
46. bob
47. =
48. Agent
49. (
50. name
51. =
52. "bob-0"
53. ,
54. seed
55. =
56. "agent bob-0 secret phrase"
57. ,
58. port
59. =
60. 8001
61. ,
62. endpoint
63. =
64. [
65. "http://localhost:8001/submit"
66. ],
67. )
68. my_wallet
69. =
70. LocalWallet
71. .
72. from_unsafe_seed
73. (
74. "registration test wallet"
75. )
76. name_service_contract
77. =
78. get_name_service_contract
79. (test
80. =
81. True
82. )
83. DOMAIN
84. =
85. "agent"
86. for
87. wallet
88. in
89. [my_wallet
90. ,
91. bob
92. .
93. wallet]
94. :
95. fund_agent_if_low
96. (wallet.
97. address
98. ())

99. Here,bob
100. is initialized with a uniquename
101. , a secretseed
102. for authentication, a specificport
103. for communication, and anendpoint
104. for submitting data. For additional information on endpoints, visit ourdocumentation ↗
105. . A local wallet namedmy_wallet
106. is then created using theseed
107. phrase"registration test wallet"
108. .
109. Thename_service_contract
110. variable is created to hold an instance of a name service contract related to registering and managing agent names. Thetest=True
111. parameter indicates that this is a test environment.DOMAIN
112. is set to"agent"
113. , indicating the domain for the agent names. Thefor
114. loop iterates over the list of wallets (my_wallet
115. andbob.wallet
116. ): for each wallet, it checks if the wallet's balance is low and if so, it funds the wallet usingfund_agent_if_low()
117. .
118. We now need to define the behavior and functions of `bob agent.
119. @bob
120. .
121. on_event
122. (
123. "startup"
124. )
125. async
126. def
127. register_agent_name
128. (
129. ctx
130. :
131. Context):
132. await
133. name_service_contract
134. .
135. register
136. (
137. bob.ledger, my_wallet, ctx.address, ctx.name, DOMAIN
138. )
139. @bob
140. .
141. on_message
142. (model
143. =
144. Message)
145. async
146. def
147. message_handler
148. (
149. ctx
150. :
151. Context
152. ,
153. sender
154. :
155. str
156. ,
157. msg
158. :
159. Message):
160. ctx
161. .
162. logger
163. .
164. info
165. (

166. f
167. "Received message from
168. {
169. sender
170. }
171. :
172. {
173. msg.message
174. }
175. "
176. )
177. if
178. **name**
179. ==
180. "**main**"
181. :
182. bob
183. .
184. run
185. ()
186. Here, we defined a.on_event("startup")
187. decorator on the register_agent_name() function. This decorator indicates that the function is executed when the agent starts up. The function takes aContext
188. object as parameter, and registers the agent's name and domain using thename_service_contract
189. variable we previously defined. The function uses information from the context (ctx
190. ) including the agent's ledger, wallet, address, name, and domain.
191. Save the script.

The overall script should look as follows:

agent_1.py from cosmpy . aerial . wallet import LocalWallet

from uagents . network import get_name_service_contract from uagents . setup import fund_agent_if_low from uagents import Agent , Context , Model

class

Message ( Model ): message :

str

# bob

Agent ( name = "bob-0" , seed = "agent bob-0 secret phrase" , port = 8001 , endpoint = [ "http://localhost:8001/submit" ], )

# my_wallet

LocalWallet . from_unsafe_seed ( "registration test wallet" ) name_service_contract =

get_name_service_contract (test = True ) DOMAIN =

"agent"

for wallet in [my_wallet , bob . wallet] : fund_agent_if_low (wallet. address ())

@bob . on_event ( "startup" ) async

def

register_agent_name ( ctx : Context): await name_service_contract . register ( bob.ledger, my_wallet, ctx.address, ctx.name, DOMAIN )

@bob . on_message (model = Message) async

def

message_handler ( ctx : Context ,

sender :

str ,

msg : Message): ctx . logger . info ( f "Received message from { sender } : { msg.message } " )

if

**name**

==

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

## Agent 2: alice

We can now define the script for our second agentalice .

1. First of all, let's import all needed classes fromuagents.network
2. ,uagents.setup
3. , anduagents
4. . Then, we need to define a message data model for types of messages to be exchanged between our agents:
5. from
6. uagents
7. .
8. setup
9. import
10. fund_agent_if_low
11. from
12. uagents
13. import
14. Agent
15. ,
16. Context
17. ,
18. Model
19. class
20. Message
21. (
22. Model
23. ):
24. message
25. :
26. str
27. We then need to initialize our agentalice
28. andregister it in the Almanac ↗
29. making sure it has enough funds in its wallet to register:
30. alice
31. =
32. Agent
33. (
34. name
35. =
36. "alice-0"
37. ,
38. seed
39. =
40. "agent alice-0 secret phrase"
41. ,
42. port
43. =
44. 8000
45. ,
46. endpoint
47. =
48. [
49. "http://localhost:8000/submit"
50. ],
51. )
52. fund_agent_if_low

53. (alice.wallet.
54. address
55. ())
56. We can now define the behavior ofalice
57. agent:
58. @alice
59. .
60. on_interval
61. (period
62. =
63. 5
64. )
65. async
66. def
67. alice_interval_handler
68. (
69. ctx
70. :
71. Context):
72. bob_name
73. =
74. "bob-0.agent"
75. ctx
76. .
77. logger
78. .
79. info
80. (
81. f
82. "Sending message to
83. {
84. bob_name
85. }
86. ..."
87. )
88. await
89. ctx
90. .
91. send
92. (bob_name,
93. Message
94. (message
95. =
96. "Hello there bob."
97. ))
98. if
99. **name**
100. ==
101. "**main**"
102. :
103. alice
104. .
105. run
106. ()
107. Here we defined aalice_interval_handler()
108. function. This function is a decorated with a.on_interval()
109. decorator indicating that the function is executed at intervals of 5 seconds. This function sends a message tobob
110. agent taking into account its name"bob-0.agent"
111. . Themessage
112. being sent contains the a"Hello there bob."
113. message
114. Save the script.

The overall script should look as follows:

agent_2.py from uagents . setup import fund_agent_if_low from uagents import Agent , Context , Model

class

Message ( Model ): message :

str

# alice

Agent ( name = "alice-0" , seed = "agent alice-0 secret phrase" , port = 8000 , endpoint = [ "http://localhost:8000/submit" ], )

fund_agent_if_low (alice.wallet. address ())

@alice . on_interval (period = 5 ) async

def

alice_interval_handler ( ctx : Context): bob_name =

"bob-0.agent" ctx . logger . info ( f "Sending message to { bob_name } ..." ) await ctx . send (bob_name, Message (message = "Hello there bob." ))

if

**name**

==

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

## Run the scripts

We are now ready to run the scripts. Remember to check if you are in the correct directory and to correctly activate your virtual environment.

Importantly, runagent_1.py before runningagent_2.py scripts and run them from different terminals:

- Terminal 1:python agent_1.py
- Terminal 2:python agent_2.py

The output should be as follows, depending on the terminal:

- bob
- [bob-0]: Registering on almanac contract...
- [bob-0]: Registering on almanac contract...complete
- [network]: Registering name...
- [network]: Registering name...complete
- [bob-0]: Starting server on http://0.0.0.0:8001 (Press CTRL+C to quit)
- [bob-0]: Received message from agent1qwquu2d237gntfugrnwch38g8jkl76vdr05qjm4wyps6ap04fvt8vtzhpqw: Hello there bob.
- [bob-0]: Received message from agent1qwquu2d237gntfugrnwch38g8jkl76vdr05qjm4wyps6ap04fvt8vtzhpqw: Hello there bob.
- [bob-0]: Received message from agent1qwquu2d237gntfugrnwch38g8jkl76vdr05qjm4wyps6ap04fvt8vtzhpqw: Hello there bob.
- alice
- [alice-0]: Registering on almanac contract...
- [alice-0]: Registering on almanac contract...complete
- [alice-0]: Starting server on http://0.0.0.0:8000 (Press CTRL+C to quit)
- [alice-0]: Sending message to bob-0.agent...
- [alice-0]: Sending message to bob-0.agent...
- [alice-0]: Sending message to bob-0.agent...

## Was this page helpful?

AI Agents: broadcast  Running a Locally Hosted Agent with LangChain Integration