Agents protocols

Within the uAgents Framework,protocols support capturing related message types and handlers.

Protocols are used to facilitate communication and interaction between agents in the Framework.

ℹ Indeed, any agent including the same protocol will be able to communicate with each other. Aprotocol is built similar to an agent, but it has no identity and cannot be run. Protocols only contains the message types and handlers that define some components of an agent's functionality.

Let's use asimple restaurant table booking request as an example to better understand what a protocol means and how to build one:

1. Let's start by creating a folder for ourprotocols
2. . Then, let's create Python script within it, and name it:
3. mkdir protocols
4. and

windows echo .

> book . py 1. We import fromuagents 2. library the necessary classesContext 3. ,Model 4. , andProtocol 5. . Then, need to define the type of messages that the handler will receive and send: 6. from 7. uagents 8. import 9. Context 10. , 11. Model 12. , 13. Protocol 14. class 15. BookTableRequest 16. (17. Model 18.): 19. table_number 20. : 21. int 22. class 23. BookTableResponse 24. (25. Model 26.): 27. success 28. : 29. bool 30. We use theModel 31. class fromuagents 32. library to defineBookTableRequest 33. andBookTableResponse 34. classes for setting up the structure of messages to be exchanged between your agents. TheBookTableRequest 35. class represents a request to book a table, containing the desired table number, while theBookTableResponse 36. class represents the response to that request, indicating whether the booking was successful. 37. Now, we would need to define the booking protocol asbook_proto 38. and also define the desired logic to determine if theBookTableResponse 39. will be successful or not: 40. book_proto 41. = 42. Protocol 43. () 44. @book_proto 45. . 46. on_message 47. (model 48. = 49. BookTableRequest, replies 50. = 51. {BookTableResponse}) 52. async 53. def 54. handle_book_request 55. (56. ctx 57. : 58. Context 59. , 60. sender 61. : 62. str 63. , 64. msg 65. : 66. BookTableRequest): 67. if 68. ctx 69. . 70. storage 71. . 72. has 73. (74. str 75. (msg.table_number)): 76. success 77. = 78. False 79. else 80. : 81. success 82. = 83. True 84. ctx 85. . 86. storage 87. . 88. set 89. (90. str 91. (msg.table_number), sender) 92. # send the response 93. await 94. ctx 95. . 96. send 97. (sender, 98. BookTableResponse 99. (success 100. = 101. success)) 102. We can then import our booking protocol from into the script we create for our agent, in the following way: 103. from 104. protocols 105. . 106. book 107. import 108. book_proto 109. If your agent is calledrestaurant 110. you can include the protocol in this way: 111. restaurant 112. . 113. include 114. (book_proto)

For a better understanding of these concepts, consider having a look at theAgents storage andExchange protocol resources and consider going through the extensiveHow to book a table at a restaurant using agentsguide in theAgents guides section. Also, check out theAgents: broadcast guide for an additional implementation of protocols in Agents communication.

Last updated on November 22, 2024

Was this page helpful?

You can also leave detailed feedback**on Github**

Communicating with other agents Agent Handlers (on_...)