Chat Protocol in the Agentverse using Fetch.ai

**1. The Agentverse by Fetch.ai**

* Decentralized identifiers (DIDs) for secure identity
* libp2p for peer-to-peer communication
* Protocol Buffers (protobuf) for message schema definition
* FastAPI-based infrastructure for endpoint deployment

**2. Architectural Overview**

The chat protocol operates as a registered behavior within the uAgents framework. Each agent is capable of:

* Subscribing to specific protocols
* Defining message schemas for those protocols
* Asynchronously handling and responding to messages

**2.1 Key Components**

| **Component** | **Description** |
| --- | --- |
| Agent | An autonomous entity with a DID capable of executing protocols |
| Protocol | A defined interface (e.g., chat) with supported message types and handlers |
| Message Model | A pydantic schema defining message content |
| Context (ctx) | Provides runtime info, such as sender DID and send function |
| Handler | Async function triggered when a matching message arrives |

**3. Protocol Definition**

**3.1 Message Schema**

The ChatMessage model defines a single field text of type str. This field holds the conversational message.

from uagents import Model

class ChatMessage(Model):

text: str

**3.2 Protocol Registration**

This registers the protocol/agent named "chat" with version control, enabling upgrades or modifications in the future.

from uagents import Protocol

chat\_protocol = Protocol(name="chat", version="1.0")

**3.3 Message Handler**

* ctx: Context object, used to access runtime information and perform actions (like sending a message)
* sender: DID of the agent that sent the message
* msg: Deserialized ChatMessage object

from uagents import Context

@chat\_protocol.on\_message(model=ChatMessage)

async def handle\_message(ctx: Context, sender: str, msg: ChatMessage):

response\_text = f"Received your message: {msg.text}"

await ctx.send(sender, ChatMessage(text=response\_text))

**4. Communication Lifecycle**

**4.1 Agent Initialization**

Agents are instantiated with a unique DID and endpoint:

from uagents import Agent

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

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

**4.2 Protocol Inclusion**

Agents include the chat protocol using:

alice.include(chat\_protocol)

bob.include(chat\_protocol)

**4.3 Message Exchange**

Alice sends a message to Bob:

await alice.send(bob.address, ChatMessage(text="Hello, Bob!"))

Bob receives this message and responds through the handler defined earlier.

**5. Agentverse Deployment**

When deployed in the Agentverse (local or cloud-hosted), agents run in containers or services that expose their endpoints via HTTP. Messages are routed via libp2p or a relay network using Fetch.ai’s infrastructure.

Deployment involves:

1. Registering agents in the Agentverse console or using CLI tools.
2. Connecting agents to the Fetch.ai relay node.
3. Testing communications through simulation or direct message exchange.

**6. Advanced Features and Extensions**

**6.1 State Management**

While basic messages are stateless, developers can implement conversational context by:

* Tracking message history in memory or external storage
* Using conversation IDs or session tokens

**6.2 Message Validation**

Protobuf and pydantic validation ensure messages conform to the defined schema. This prevents malformed or malicious payloads.

**6.3 Protocol Composition**

Chat Protocol can be combined with other protocols like:

* Task allocation protocols
* Negotiation protocols
* Knowledge sharing protocols

Example:

@agent.include(chat\_protocol, another\_protocol)

**7. Use Case Scenarios**

| **Scenario** | **Description** |
| --- | --- |
| Agent Collaboration | Agents coordinate via chat to perform distributed tasks (e.g., delivery, scheduling) |
| User Interfaces | Human users send messages to agents, which respond contextually |
| Social MASs | Agents representing different interests exchange arguments or decisions |

* Does agentverse gives service discovery? More advance
* What they refer to message handlers and protocol
* Mqtt or tcp for Agentverse