Academy 1.0: Resilience and Other Features

Alok Kamatar

What We've Been Focused On

Demo's and Proof of Concepts to Long-running Campaigns

- Heartbeats for detecting agent liveness
- Responding to shutdown/terminated agents
- Managing token expiration

Compatibility and Interfaces for Future Versions

- Customizable non-pickle serialization
- (Hidden) Versions and compatibility checks in message protocol

Agent Heartbeats

"How do I know if an agent is alive?"

- Introduced new heartbeat loop, configurable with `ExchangeClientConfig`
- Status
  - Missing -> Mailbox with ID had not been created
  - Active -> Recent activity (< n missed heartbeats)
  - Inactive -> No listening agent (> n missed heartbeats)
  - Terminated -> Mailbox closed, not accepting requests

```
exchange.status(<agent_id>)
```

Configuring Heartbeats

- Heartbeats are only a push mechanism
- `status` is calculated on the client side — can configure the number of missing heartbeats before agent is marked inactive

```
from academy.exchange.client_config \
    import ExchangeClientConfig

config = ExchangeClientConfig(
    heartbeat_interval: float
    stale_heartbeat_interval: int
)

factory = HttpExchangeFactory(
    config=config
)
```

Cancelling Actions

- Requests wait for agents to be online by design
- Dead/disconnected agents can cause workflows to hang
- Now this is configurable:

```
handle = await manager.launch(...)
handle.reject_on_inactive = True
await handle.shutdown()

# Agent is inactive
# Raises `AgentInactiveError`
await handle.action(...)
```

Token Refresh Simplified

Authentication tokens used by Academy expire every 2 days

- Token's are refreshed when a manager is launched, but tokens are statically sent to agents
- This would manifest as the entire workflow dying
- Now, a forbidden error triggers a refresh
- Caveat: you must be authenticated on every machine

Customizing Serialization

- A bad combination:
  - Pickle is the default serialization method
  - Pickle is dangerous
- But “it just works”
- Configuring the serialization of agents and managers

```
rtc = RuntimeConfig(
   default_serializer=SerializationStrategy.JSON,
   allowed_deserializers={SerializationStrategy.JSON,}
)
hdl = await manager.launch(<agent>, config=rtc)
hdl.request_serializer = SerializationStrategy.JSON
```

Rapid Fire: Other Features You May Have Missed

- Custom Executor for Launching Agents within the same Event Loop
- HttpExchangeConsole - sharing agents with Globus groups
- Systemd Agents – configure, launch and resume agents from a TOML File
- GlobusExchangeClient - a different way of managing tokens with more identity possibilities

Please Contribute!

What other things should we be thinking about?

What problems have you run into with Academy?

What things would you like to do with agents that you can't?

Open PRs, join Slack, send emails, and please talk to us!

The GlobusExchangeClient

- Each agent is minted as its own Identity
- The agent gets a specific (delegated) token from the user created for that identity
- That delegated token is exchanged for an access token and refresh token for the exchange
- The refresh token is used to keep the the access token fresh