Application Profile (with scope)

Requirements (MVP)

1. producer can only send messages that match a filter on type/property
2. can subscribe to a container other than inbox
3. filter subscription by type
4. filter subscription results by property (rule driven)
5. frame subscriptions results by property (data consumer driven)

Examples

E1: filter producer messages allowed by type

Use case: User wants to restrict producer content for security reasons

```
type: Authorization
agent: <producer>
accessTo: <user>/inbox/
mode: Write
scope:
 matchObject:
    type: TestAction
```

E1.test

1. set ACL without scope
2. send message with producer key and type=TestAction
3. confirm test subscriber receives message
4. set ACL with scope
5. send message with producer key and type=TestAction
6. message should be rejected (requires middleware spec)
7. confirm subscriber does not receive message

E2: Subscribe to a container

Use case: subscriber only wants changes for a container/collection

```
type: Authorization
agent: <subscriber>
accessTo: <user>/path/to/container/**
mode: Read
```

E2.test

1. set ACL
2. post a message to the container as subscriber
3. confirm: message should be rejected because ACL is read only
4. post a message to the container from an admin producer
5. confirm message is received by subscriber

E3: filter subscription by type

Use case: Subscriber is only interested in certain Actions

```
type: Authorization
agent: <subscriber>
accessTo: <user>/inbox/
mode: Read
scope:
- type: AskAction
```

E3.test

1. set ACL
2. send a message of type=AskAction
3. confirm message for subscriber is delivered to the pod/outbox
4. send a message of type=Foo
5. confirm no message for subscriber is in the outbox

E4: filter subscription results by property

Use case: user wants to protect personal information by removing personal data from subscription results

```
type: Authorization
agentClass: foaf:Agent # matches everyone
accessTo: <user> # matches the entire pod
scope:
    policy:
    - type: Transform
      name: RedactEmailAddresses
      value:
        email: $redact
```

E4.test

1. set ACL (on root I guess)
2. send a message that includes an email property
3. confirm message delivered to the outbox has email value of 'REDACTED'

E5: frame results

Use case: Subscriber wants to normalize their messages

```
type: Authorization
agent: <subscriber>
accessTo: <resource>
scope:
  frame:
    Person:
      name: .
      email: [.]
```

E5.test

1. set ACL
2. send a message with a Person object containing more than name and email
3. confirm subscription receives only name and email properties
4. confirm subscription email is an array with only one value