Organizational Agent - Prova Knowledge Base:

The Prova knowledge base describes rules that help the Organizational Agent delegate queries to the appropriate Personal Agents in the community. This document describes the framework that Rule Responder uses and some of the general performatives.

- **<u>Location:</u>**
  Each Prova knowledge base file is pointed to by the Mule config file. Generally the KB location has the following format:

  `PragmaticAgentWeb\rules\use_case`*`Instance`*`\`*`NameOfKB.prova`*

- **<u>Assigning the OWL Ontology:</u>**
  A fact is used to point Prova to the OWL ontology used with the particular Rule Responder instance.
  e.g.:

```
%import external ontology of responsibility assignment matrix
import("http://ruleml.org/WellnessRules/files/WellnessRules.owl").
```

- **<u>Defining Interfaces:</u>**
  Interfaces describe what queries are expected to be issued to the Rule Responder. They have the following format:

```
interface(performative(Performative),performative("?"),"description").
```

  For example:

```
interface(getContact(Topic,Request,Contact),
      getContact("+","+","-"),"request personal contact information for
      a certain Topic and Request regarding RuleML-2008").
```

  The first argument is the performative. The second is the relation, with +'s and -'s to represent constants and variables, respectively. The last argument is the description of the performative.

  These interfaces must be written as facts for every expected query, otherwise a "no public interface" message will be returned.

- **<u>General processMessage Rule:</u>**
  A query is handled by the processMessage rule and its conclusion looks like the following:

```
processMessage(XID,From,UserName,performative(Performative)) :-
```

  XID = The name of the OA (e.g. WellnessRules).
  From = The name of the endpoint
  Primitive = Username (e.g. User)
  performative = The relation name surrounded by <Rel>
  Performative = The contents of the relation (<Ind>'s, <Vals>'s, <Expr>'s, <Plex>'s, etc.)

  Optionally, you then can check the free and bound variables to restrict the rule to certain <Ind>'s (constants) and <Var>'s (variables):

```
free(ProfileID),
bound(InOut,
```

free = A variable (<Var>)
bound = A constant (<Ind>)

A rule is then used to “look up” the responsible Agent:

```
assigned(XID,Agent,Responsibility,Role),
```

XID = The name of the OA (e.g. WellnessRules)
Agent = Will be the name of the found Agent
Responsibility = A responsibility name for the query (provided in the query somehow).
Role = What kind of role the responsibility will be matched to.

The query is then sent to the Agent that was found:

```
sendMsg(XID,esb,Agent,Type,performative(Performative)),
```

XID = The name of the OA (e.g. WellnessRules)
Agent = Will be the name of the found Agent
Type = “query”
performative = The relation name of the query
Performative = The contents of the relation

The answer is then received from the Agent:

```
rcvMult(XID,esb,Agent,Type,performative(Performative)),
```

XID = The name of the OA (e.g. WellnessRules)
Agent = Will be the name of the found Agent
Type = “answer”
performative = The relation name of the query
Performative = The contents of the relation

The answer is then sent back to the EA:

```
sendMsg(XID,esb,From,"answer",performative(Performative),
```

XID = The name of the OA (e.g. WellnessRules)
Agent = Will be the name of the EA endpoint
Type = “answer”
performative = The relation name of the query
Performative = The contents of the relation