***Rules Responder - RuleML Queries:***

Rule Markup Language (RuleML) is the language used for rule interchange. In the case of Rule Responder, RuleML is used as a generic query language, which will be transformed to Prova, POSL, and N3. The following are the basics for Rule Responder's use of RuleML:

- **<u>Message Header:</u>**
  Contains the namespaces used in the query (they are the same across all Rule Responder).

  ```
  <RuleML xmlns=http://www.ruleml.org/0.91/xsd
       xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
       xsi:schemaLocation=http://www.ruleml.org/0.91/xsd
       http://ibis.in.tum.de/research/ReactionRuleML/0.2/rr.xsd
       xmlns:ruleml2007="http://ibis.in.tum.de/projects/paw#">
  ```

- **<u>Message Footer:</u>**
  Contains the remainder of the message following the payload.

  ```
     </Message>
  </RuleML>
  ```

  - **<u>Message Payload:</u>**
    This is where the query is held.

  - **<u>Message Payload - Header:</u>**
    The head of the message payload is the same throughout.

    ```
    <Message mode="outbound" directive="query-sync">
    ```

  - **<u>Message Payload - OID:</u>**
    Contains the name of the Organizational Agent as a constant.

    ```
    <oid>
          <Ind>WellnessRules</Ind>
    </oid>
    ```

  - **<u>Message Payload - Protocol:</u>**
    The protocol used for message transfer (almost always esb).

    ```
    <protocol>
          <Ind>esb</Ind>
    </protocol>
    ```

  - **<u>Message Payload - Sender:</u>**
    The message must have a unique sender (username). Functionality has yet to be implemented.

    ```
    <sender>
          <Ind>User</Ind>
     </sender>
    ```

  - **<u>Message Payload - Content:</u>**
    This is the query itself. It can contain X number of Atoms.

```
<content>
...
</content>
```

- **Message Payload - Atom:**
  An atom of the query (all instances only use one).

```
<Atom>
...
</Atom>
```

- **Message Payload - Query:**
  A query has a single relation name, followed by constants, variables and complex expressions.

```
<Rel> = The relation name
<Ind> = Individual constant
<Var> = Variable
<Expr> = Complex expression
e.g.:

      <Rel>myActivity</Rel>
      <Var>ProfileID</Var>
      <Ind>Running</Ind>
      <Var>InOut</Var>

Looks like the following in Prolog:
myActivity(ProfileID,running,InOut).

or in POSL:
myActivity(?ProfileID,Running,?InOut).

or in N3:
_:myActivity
      rdf:type      :MyActivity;
      :profileID    ?ProfileID;
      :activity     :Running;
      :inOut ?InOut.
```