

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA RDF Content Language Specification

Document title	FIPA RDF Content Language Specification		
Document number	XC00011B	Document source	FIPA TC C
Document status	Experimental	Date of this status	2001/08/10
Supersedes	FIPA00003		
Contact	fab@fipa.org		
Change history			
2000/08/18	Approved for Experimental		
2001/08/10	Line numbering added		

© 2000 Foundation for Intelligent Physical Agents - <http://www.fipa.org/>

Geneva, Switzerland

Notice

Use of the technologies described in this specification may infringe patents, copyrights or other intellectual property rights of FIPA Members and non-members. Nothing in this specification should be construed as granting permission to use any of the technologies described. Anyone planning to make use of technology covered by the intellectual property rights of others should first obtain permission from the holder(s) of the rights. FIPA strongly encourages anyone implementing any part of this specification to determine first whether part(s) sought to be implemented are covered by the intellectual property of others, and, if so, to obtain appropriate licenses or other permission from the holder(s) of such intellectual property prior to implementation. This specification is subject to change without notice. Neither FIPA nor any of its Members accept any responsibility whatsoever for damages or liability, direct or consequential, which may result from the use of this specification.

19 **Foreword**

20 The Foundation for Intelligent Physical Agents (FIPA) is an international organization that is dedicated to promoting the
21 industry of intelligent agents by openly developing specifications supporting interoperability among agents and agent-
22 based applications. This occurs through open collaboration among its member organizations, which are companies and
23 universities that are active in the field of agents. FIPA makes the results of its activities available to all interested parties
24 and intends to contribute its results to the appropriate formal standards bodies.

25 The members of FIPA are individually and collectively committed to open competition in the development of agent-
26 based applications, services and equipment. Membership in FIPA is open to any corporation and individual firm,
27 partnership, governmental body or international organization without restriction. In particular, members are not bound to
28 implement or use specific agent-based standards, recommendations and FIPA specifications by virtue of their
29 participation in FIPA.

30 The FIPA specifications are developed through direct involvement of the FIPA membership. The status of a
31 specification can be either Preliminary, Experimental, Standard, Deprecated or Obsolete. More detail about the process
32 of specification may be found in the FIPA Procedures for Technical Work. A complete overview of the FIPA
33 specifications and their current status may be found in the FIPA List of Specifications. A list of terms and abbreviations
34 used in the FIPA specifications may be found in the FIPA Glossary.

35 FIPA is a non-profit association registered in Geneva, Switzerland. As of January 2000, the 56 members of FIPA
36 represented 17 countries worldwide. Further information about FIPA as an organization, membership information, FIPA
37 specifications and upcoming meetings may be found at <http://www.fipa.org/>.

38 **Contents**

39	1	Introduction	1
40	1.1	A Summary of RDF	1
41	2	RDF as a FIPA Content Language	2
42	2.1	Objects.....	2
43	2.2	Propositions	2
44	2.3	Actions	4
45	2.4	Action Implementations.....	6
46	3	Exchange of Rules Extensions	10
47	3.1	Introduction	10
48	3.2	Rules in XML/RDF	10
49	3.3	Exchanging Rules as Programming Code	11
50	3.4	Using Rules with FIPA Communicative Acts	12
51	3.5	Further Remarks.....	12
52	4	Examples of Use.....	13
53	4.1	RDF Schemas for FIPA RDF 0.....	13
54	4.2	RDF Schemas for FIPA RDF 1	14
55	5	References.....	16
56			

56 1 Introduction

57 This specification describes how the Resource Description Framework (RDF - see [W3crdf]) can be used as content
58 language in a FIPA message. Although FIPA does not require that a content language is able to represent actions¹, a lot
59 of communicative acts require actions in their content. Therefore, we will show how RDF schemas can be defined
60 extending its model to express:

61
62 **Objects** which are constructs that represent an identifiable entity (be it abstract or concrete) in the domain of
63 discourse,

64
65 **Propositions** which are statements expressing that some sentence in a language is true or false, and,

66
67 **Actions** which try to express an activity that can be carried out by an object.
68

69 By means of existing mechanisms in RDF (schema definitions), modular RDF extensions will be proposed, based on
70 the *fipa-rdf0* content language. Those extensions will be able to tackle for example rules, logic algebra constructs,
71 and others. These extensions can then be labelled as *fipa-rdf1*, *fipa-rdf2*, etc.
72

73 The general motivation behind this approach is to enable and ease the use of RDF Schemas emerging on the Web
74 such as CC/PP, and to define one common standard approach here to increase the level of interoperability. The main
75 strengths of the RDF language are in its extensibility, reusability and simplicity. Another advantage of RDF is that data
76 and schemas are exchanged in a similar way.
77

78 The RDF model proposes the eXtensible Markup Language (XML - see [W3Cxml]) as an encoding syntax, but does not
79 prevent anyone from using alternative encoding schemes. All *fipa-rdf* examples will therefore use an XML encoding,
80 although, in principle, other encoding schemes could be used.
81

82 1.1 A Summary of RDF

83 The RDF framework is based on an entity-relationship model. The RDF Data Model is described by means of
84 resources, properties and their values. A specific resource together with one or more named properties plus the values
85 of these properties is an RDF description (a collection of RDF statements).
86

87 In addition to the RDF Data Model, the RDF Schemas (see [W3Crdfs]) specification provides a typing system for the
88 resources and properties used in the RDF data. It defines concepts such as classes, subclasses, properties or sub-
89 properties. It also allows expressing constraints. Both the RDF Data Model and RDF Schema propose XML as a
90 serialization syntax.
91

92 RDF is a "foundation for processing meta-data in the way that it provides interoperability between applications that
93 exchange machine-understandable information." This suggests that RDF could be most useful to facilitate knowledge
94 sharing and exchange between agents.
95
96

¹ A content language must be able to express at least any of propositions, objects or actions.

96 2 RDF as a FIPA Content Language

97 To be able to use RDF as a content language for FIPA ACL messages, we have to explore how objects, propositions
98 and functions can be expressed in RDF, without endangering key extensibility inherent to the language. On the other
99 hand, we will try to preserve RDF's simplicity, which is crucial for the success of languages on the Internet.

100 We suggest to use the name `fipa-rdf0`, for the combined use of RDF and the basic schemas which define the
101 extensions needed for FIPA.
102
103

104 2.1 Objects

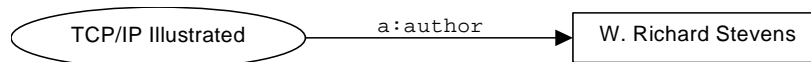
105 Taking the above into account, it is obvious to see an analogy between an ACL object and an RDF resource, since both
106 are defined as descriptions of a certain identifiable entity. This enables us to use RDF resource identifiers and
107 references as ACL object identifiers and references. This means that to resolve an RDF reference, we can use a the
108 FIPA communicative act `query-ref` (see [FIPA00054]), which will then be followed by an 'inform' message, describing
109 this object.
110

111 2.2 Propositions

112 In the same context it seems logical to model ACL propositions using RDF statements. An RDF statement is composed
113 out of three parts: subject (resource), predicate (property) and object (literal/value). As an example, consider the
114 sentence "W. Richard Stevens is the author of TCP/IP Illustrated". The RDF components of this proposition are the
115 subject (TCP/IP Illustrated), the predicate (Author) and the object (W. Richard Stevens). This sentence/statement can
116 then be described in RDF in the following manner:

```
117 <?xml version="1.0"?>
118 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
119     xmlns:s="http://description.org/schema/">
120     <rdf:Description ID="TCP/IP Illustrated">
121         <s:author>W. Richard Stevens</s:author>
122     </rdf:Description>
123 </rdf:RDF>
```

124 *Figure 1* represents this in RDF graph form. This way we have a starting point to state logical expressions in our
125 content. Taking this one step further, we can say that by expressing this statement, we indicate our belief in this
126 statement. In this way we can say that we always assume that an RDF statement expresses a belief. This approach
127 would be sufficient in any context where the level of logic involved is limited.
128
129
130
131



132
133
134 **Figure 1: A Proposition as an RDF Statement**
135

136 To overcome this shortcoming however, we will explain how logical belief or disbelief of a certain statement could be
137 expressed explicitly using RDF. To express that we believe a statement to be true or false, we have to model the
138 original statement as a reified statement, that is, a resource with four predefined properties:
139

140 The **subject** property identifies the resource being described by the modelled statement; that is, the value of this
141 property is the resource about which the original statement was made.
142

143 The **predicate** property identifies the property of the original statement; that is, the value is the specific property in
144 the original statement.
145

146 The **object** property identifies the property value in the original statement; that is, the value is the object in the
 147 original statement.

148

149 The value of the **type** property describes the type of the new resource. All reified statements are instances of
 150 `rdf:Statement`.

151

152 A new resource with the above four properties represents the original statement and can both be used as the object of
 153 another statement and have additional statements made about it. The resource with these four properties is not a
 154 replacement for the original statement, but it is a model of the statement.

155

156 By extending the RDF syntax model with the following elements, a means to express belief or disbelief of a statement is
 157 allowed (the complete schema of the RDF extensions can be found in *Section 4.1, RDF Schemas for FIPA RDF 1*):

158

```
159 <?xml version="1.0"?>
160 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
161     xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#">
162
163   <rdfs:Class rdf:ID="http://www.fipa.org/schemas#Proposition">
164     <rdfs:label xml:lang="en">proposition</rdfs:label>
165     <rdfs:label xml:lang="fr">proposition</rdfs:label>
166     <rdfs:comment>This describes the set of propositions</rdfs:comment>
167     <rdfs:subClassOf rdf:resource=
168       "http://www.w3.org/1999/02/22-rdf-syntax-ns#Statement"/>
169   </rdfs:Class>
170
171   <rdfs:ConstraintProperty rdf:ID="http://www.fipa.org/schemas#belief">
172     <rdfs:label xml:lang="en">belief</rdfs:label>
173     <rdfs:label xml:lang="fr">acte</rdfs:label>
174     <rdfs:domain rdf:resource="#Proposition"/>
175     <rdfs:range rdf:resource=
176       "http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
177   </rdfs:ConstraintProperty>
178 </rdf:RDF>
```

179

180 Using this method we can easily describe ACL propositions in RDF. As an example, the following proposition will be
 181 modelled: "The statement 'W. Richard Stevens is the author of TCP/IP Illustrated' is true". One way to do this is as
 182 follows:

183

```
184 <?xml version="1.0"?>
185 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
186     xmlns:fipa="http://www.fipa.org/schemas/fipa-rdf0#">
187
188   <fipa:Proposition>
189     <rdf:subject>TCP/IP Illustrated</rdf:subject>
190     <rdf:predicate rdf:resource="http://description.org/ schema#author"/>
191     <rdf:object>W. Richard Stevens</rdf:object/>
192     <fipa:belief>true</fipa:belief>
193   </fipa:Proposition>
194 </rdf:RDF>
```

195

196 Expressing that the same statement is false, is equally easy by replacing the value 'true' with 'false'. The RDF graph
 197 representation of the 'false' statement is presented in *Figure 2*.

198

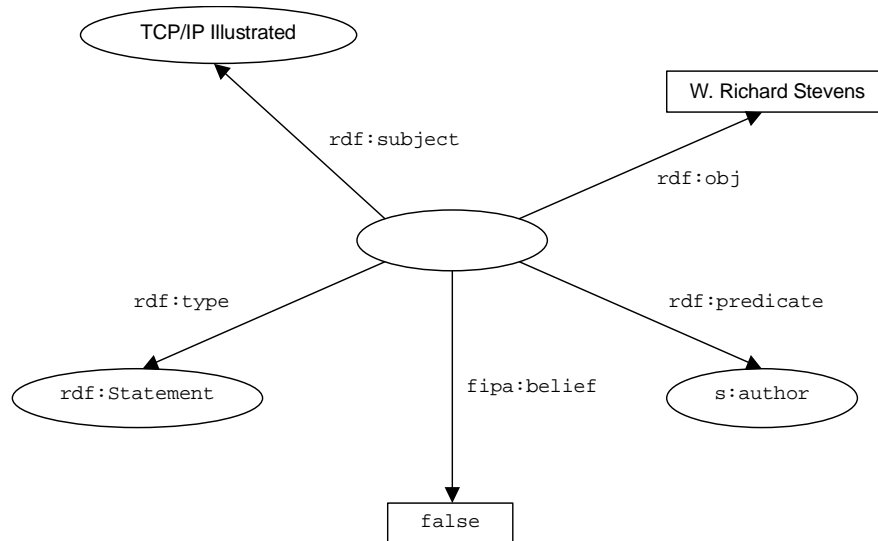


Figure 2: Explicit Logical Proposition in RDF

199
200
201
202

2.3 Actions

An **action** expresses an activity, carried out by an object. There are three different properties related to an 'action':

203

204

205

206

207

208

An **act** identifies the operative part of the action; it can serve to identify the type of act or merely to describe the act. In the latter case specific types of action classes can be derived from the Action class.

209

210

211

An **actor** identifies the entity responsible for the execution of the action, that is, the value is the specific entity which will/can/should perform the act (often the receiver, but possibly another agent/entity under "control" of the receiver).

212

213

214

An **argument** identifies an (optional) entity which can be used for the execution of the action; that is, the value is entity which is used by the actor to perform the act. An action can have multiple arguments.

When looking at an action this way, there is a structural analogy with a RDF statement.

215

216

To model an action, the RDF syntax model can be extended with a new RDF type `fipa:Action` which has these properties. As an example, the following action will be modelled: "John opens door1 and door2". In this small example, the properties are the act (Open), the actor (John) and the arguments (door1 and door2). In RDF, this action can then be described as:

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:fipa="http://www.fipa.org/schemas/fipa-rdf0#">
  <fipa:Action rdf:ID="JohnAction1">
    <fipa:actor>John</fipa:actor>
    <fipa:act>open</fipa:act>
    <fipa:argument>
      <rdf:bag>
        <rdf:li>door1</rdf:li>
        <rdf:li>door2</rdf:li>
      </rdf:bag>
    </fipa:argument>
  </fipa:Action>
</rdf:RDF>
```

238 According to the RDF specification, the resource type defined in the schema corresponding to the type property can be
 239 used directly as an element name when the `Description` element contains a type property. So, a shorter version of
 240 the above example could be written as follows:
 241

```
242 <?xml version="1.0"?>
243 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
244     xmlns:fipa="http://www.fipa.org/schemas#">
245   <fipa:Action rdf:ID="JohnAction1">
246     <fipa:actor>John</fipa:actor>
247     <fipa:act>open</fipa:act>
248     <fipa:argument>
249       <rdf:bag>
250         <rdf:li>door1</rdf:li>
251         <rdf:li>door2</rdf:li>
252       </rdf:bag>
253     </fipa:argument>
254   </fipa:Action>
255 </rdf:RDF>
256
257
```

258 The model above still lacks the ability to state whether some action has finished or what the result is of the action. this
 259 can be solved by simply adding extra properties to the description of the action.

260

261 As an example, suppose Mary requests John to open door 1 and door 2 and then wants John to inform her if he
 262 performed the action and what the result is. This little scenario exists of two messages:

263

264 Request from Mary to John containing the description of the action, and,

265

266 Inform from John to Mary, referring to the action and stating the completion of the action.

267

268 Using FIPA ACL combined with RDF content, the first messages could be expressed as:

269

```
270 (request
271   :sender Mary
272   :receiver John
273   :content (
274     <?xml version="1.0"?>
275     <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
276       xmlns:fipa="http://www.fipa.org/schemas#">
277       <fipa:Action rdf:ID="JohnAction1">
278         <fipa:actor>John</rdf:actor>
279         <fipa:act>open</rdf:act>
280         <fipa:argument>
281           <rdf:bag>
282             <rdf:li>door1</rdf:li>
283             <rdf:li>door2</rdf:li>
284           </rdf:bag>
285         </fipa:argument>
286       </fipa:Action>
287     </rdf:RDF>)
288   :language fipa-rdf0)
289
290
```

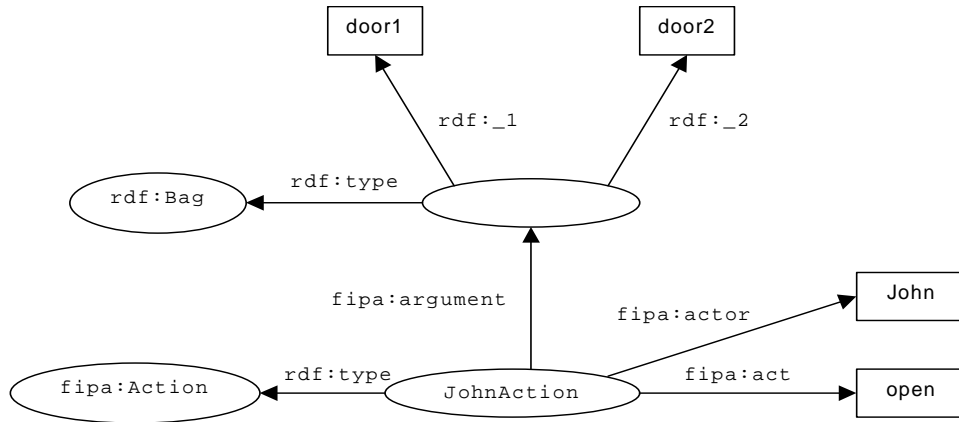



Figure 3: Example of an Open Action

And the subsequent reply message could be:

```

291 (inform
292   :sender John
293   :receiver Mary
294   :content (
295     <?xml version="1.0"?>
296     <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
297       xmlns:fipa="http://www.fipa.org/schemas#">
298       <rdf:Description about="#JohnAction1">
299         <fipa:done>true</fipa:done>
300         <fipa:result>doors closed</fipa:result>
301       </rdf:Description>
302     </rdf:RDF> )
303   :language fipa-rdf0)

```

Note the ability offered by RDF to include previous actions by means of a reference instead of repeating the whole action. The RDF graph representation of the complete action description is presented in Figure 3.

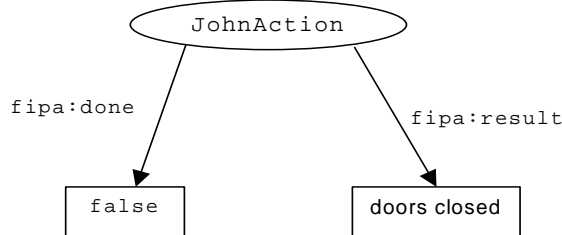


Figure 4: Result of an Open Action

2.4 Action Implementations

Different possible scenarios can be distinguished between when using the RDF actions. One possible usage is when a software designer describes in documentation (that is, in the RDF schemas in `rdfs:comment`) what is meant by a particular action; it is left to the implementer to decide which functions will be called. In another scenario, a more explicit description of the semantics might be needed by linking the action with some programming language. This section deals with the latter case.

When an agent does not know how to perform an action and needs a more explicit representation of this action, the sender agent can specify the code which implements the action. For this purpose a new property for actions is introduced, called `implementedBy`, which has a resource of the type `Code` as property its value.

329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371

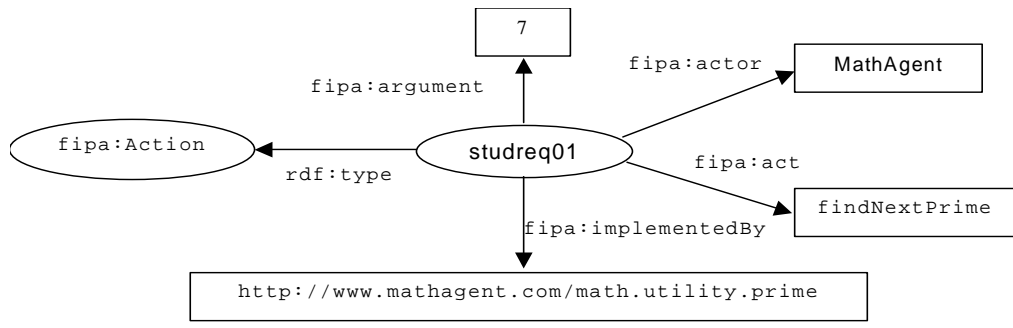
A first possibility is that the property `implementedBy` contains a reference (a URI) to an external software module written in a specific programming language. For this purposes the `Code` resource therefore has a property `language` and a property `code-uri`. For reasons of simplicity, it is assumed that the language used is either Java or a scripting language such as JScript or ECMAScript. So, the property `code-uri` is a reference to the location of code where the method or function can be found (for Java a code base followed by a class name).

When a Java class is referenced, `code-uri` can contain the Java code-base. The receiving agent can then download this class, instantiate it (if needed), and perform the required action (or not). When a non-static class is being referred, we assume that there is always a zero-argument constructor (cfr. the requirement for JavaBeans).

In addition, we assume that there always exists a one-to-one correspondence between the FIPA arguments and `fipa` result property, into the method's arguments resp. return value. When multiple arguments are used, and the sequence of those is important, one should use the `rdf:Seq` container to separate them.

As an example, suppose agent 'Student' requests agent 'Mathematician' to find the next prime following after '7'. The request message is as follows (see *Figure 5*):

```
(request
  :sender Student
  :receiver Mathematician
  :content (
    <?xml version="1.0"?>
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:fipa="http://www.fipa.org/schemas/fipa-rdf0#">
      <fipa:Action rdf:ID="studreq01">
        <fipa:actor>Mathematician</fipa:actor>
        <fipa:act>findNextPrime</fipa:act>
        <fipa:argument>7</fipa:argument>
        <fipa:implementedBy>
          <fipa:Code>
            <fipa:language>Java</fipa:language>
            <fipa:code-uri>
              http://www.mathagent.com/math.utility.prime
            </fipa:code-uri>
          </fipa:Code>
        </fipa:implementedBy>
      </fipa:Action>
    </rdf:RDF> )
  :language fipa-rdf0)
```



372
373
374
375
376
377

Figure 5: Actions and Implementation References

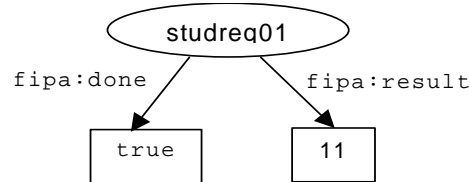
In the previous example, it is assumed that there exists a function or method in the static class `math.utility.prime.class` with the same name of the FIPA act (`findNextPrime`). If the name of the method is

378 different from the FIPA act's name, then the method name should be included after the hash sign (#) of the property
 379 value `code-uri`. For example:

```
380
381 <fipa:implementedBy>
382   <fipa:Code>
383     <fipa:language>Java</fipa:language>
384     <fipa:code-uri >
385       http://www.mathagent.com/math.utility.prime#nextPrime
386     </fipa:code-uri>
387   </fipa:Code>
388 </fipa:implementedBy>
389
```

390 The Mathematician agent could reply with:

```
391 (inform
392   :sender Mathematician
393   :receiver Student
394   :content (
395     <?xml version="1.0"?>
396     <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
397       xmlns:fipa="http://www.fipa.org/schemas/fipa-rdf0#">
398
399       <fipa:Action rdf:about="#studreq01">
400         <fipa:done>true</fipa:done>
401         <fipa:result>11</fipa:result>
402       </fipa:Action>
403     </rdf:RDF>)
404 :language fipa-rdf0)
405
406
```



407
408
409
410

Figure 6: Result of the `findNextPrime` Action

411 Sometimes, multiple implementations can be associated with one specific action so the `implementedBy` property can
 412 contain an `rdf:Alt` container of `Code` classes. In some cases, the method implementation of the code may need to
 413 refer to values of the RDF data model and conventions are needed to establish a mapping between the RDF data and
 414 (Java) object model. Although no real standards already exist, several initiatives are taking off to define such a binding.
 415 Examples include:

- 416 DATA: the Java interface (see [DATAX]),
- 417 GINF: the interfaces specified in the Generic Interoperability Framework (see [Melnik99]),
- 418 3AP: the RDF-Java mapping as used in Alcatel's 3AP platform, and,
- 419 Other Java API's have been suggested on the RDF-DEV mailing lists.

424 The following example shows the use of the binding property:

```
425 (request
426   :sender agent-dealer
427   :receiver agent-carshop
428   :content (
429     <?xml version="1.0"?>
430     <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
431
```

```

433         xmlns:fipa="http://www.fipa.org/schemas/fipa-rdf0#">
434
435     <fipa:Action rdf:ID="price-updatel">
436         <fipa:actor>agent-carshop</rdf:actor>
437         <fipa:act>addNewPrices</rdf:act>
438         <fipa:implementedBy>
439             <fipa:Code>
440                 <fipa:language>Java</fipa:language>
441                 <fipa:binding>DATA</fipa:binding>
442                 <fipa:codeURI>
443                     http://www.carshop.com/bin/CarStock
444                 </fipa:codeURI>
445             </fipa:Code>
446         </fipa:implementedBy>
447     </fipa:Action>
448 </rdf:RDF> )
449 :language fipa-rdf0)
450

```

451 The file CarStock.java could look as follows:

```

452
453 import com.muze.datax.*;
454 import com.muze.datax.rdf.*;
455
456 public class CarStock {
457
458     public CarStock() { }
459
460     void addNewPrices() {
461         EntitySet entities = new RDFReader().read("carstock.rdf");
462         DATAFactory f = new DefaultDATAFactory();
463         Iterator it = entities.iterator();
464
465         while (it.hasNext()) {
466             Entity e = (Entity)it.next();
467             Property p = e.getProperty("http://www.carshop.com/schemas#price");
468             Float price = Float.valueOf(p.getValue());
469             p-new = f.createProperty(Property.ATTRIBUTE,
470                 "newprice", 1.05*price.floatValue());
471             e.add(p-new);
472         }
473     }
474

```

475 In this example, the car dealer requests the car shop to attach new prices to their car stock: the new prices should
476 become 5% higher than the old ones. In the Java file, the DATA model is used to map the RDF data model into Java
477 objects.

478
479 A second possibility is that the `fipa:implementedBy` property includes code which is directly embedded as a (Java)
480 script. The property `fipa:script` of the resource `fipa:Code` can be used these for purposes. Once again,
481 conventions are needed to map the RDF data and the Java (script) model. For an example, see Section 3.3, .
482

483

483 3 Exchange of Rules Extensions

484 This module allows the expression and exchange of rules, based on the FIPA-RDF0 model.
485

486 3.1 Introduction

487 Using the `fipa-rdf1` language, agents can exchange knowledge about rules. An agent can inform another agent
488 about one of its own "house" rules, but may also request to fire a particular rule on (a subset of) their knowledge base.
489 In general, we leave it up to the implementer of the agent how to use the exchanged rules. The `fipa-rdf1` builds on
490 top of the `fipa-rdf0` schemas, and provides extra schema information for expressing rules.

491
492 We will distinguish between two different approaches for dealing with rules:

- 493 1. Rules exchanged as XML/RDF encoded expressions.
 - 494 2. Rules exchanged as pieces of programming code (scripts or Java classes).
- 495
496
497

498 3.2 Rules in XML/RDF

499 An RDF rule consists of two basic components: a *selection* part and a *manipulation* part, which applies to all RDF
500 resources contained in the selection. To express the selection, an RDF notation for this purpose is chosen. To express
501 the manipulation part, which allows to change property values of the selected resources, we will simply use the RDF
502 data model itself.

503
504 In order to select parts of the RDF data resources, one can use an RDF query language. No real standards do exist at
505 the moment, but various specifications are available which define how to query/select particular RDF resources
506 including:

- 507 RDF Query Specification (see [W3Crdfquery]), and,
 - 508 A Query and Inference Service for RDF (see [Decker98]).
- 509
510

511
512 The selection results will be put in an RDF container, identified by the property `fipa:selection-result` of the rule.
513 The manipulation part will then give an RDF description for all resources contained in the container of the selection
514 results. The following is an example of an RDF encoded rule:

```
515 <?xml version="1.0"?>
516 <rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#"
517         xmlns:fipa="http://www.fipa.org/schemas/fipa-rdf1#"
518         xmlns:car="http://www.cars.org/schemas#"
519         xmlns:rdfq="http://www.w3.org/TandS/QL/QL98/pp/rdfqquery.html">
520
521   <fipa:Rule rdf:ID="categorizeCars1">
522     <fipa:selection-result rdf:ID="speedycars"/>
523     <fipa:selection>
524       <rdfq:rdfqquery>
525         <rdfq:From eachResource="http://www.carshop.com/res/">
526           <rdfq:Select>
527             <rdfq:Condition>
528               <rdfq>equals>
529                 <rdfq:Property name="rdf:type"/>
530                 <rdfq:String>
531                   http://www.cars.org/schemas#Car
532                 </rdfq:String>
533               </rdfq>equals>
534             <rdfq:greaterThan>
535               <rdfq:Property name="http://www.cars.org/schemas#speed"/>
```

```

537         <rdfq:Integer>200</rdfq:Integer>
538         </rdfq:greaterThan>
539     </rdfq:Condition>
540 </rdfq>Select>
541 </rdfq:From>
542 </rdfq:rdquery>
543 </fipa:selection>
544 <fipa:manipulation>
545     <rdf:Description rdf:aboutEach="speedycars">
546         <car:category>speed-car</car:category>
547     </rdf:Description>
548 </fipa:manipulation>
549 </fipa:selection-result>
550 </fipa:Rule>
551 </rdf:RDF>
552

```

553 In the above example, first all cars are selected from all resources contained in `http://www.carshop.com/res/` for
554 which the maximum speed exceeds 200 (km/h). In the manipulation part, for all resources contained in the resulting
555 collection, the value of the property `car:category` is set to `speed-car`.
556

557 3.3 Exchanging Rules as Programming Code

558 A rule is directly expressed as some piece of code (which presumably also selects nodes, and subsequently
559 manipulates the RDF data). For this purpose, the property `fipa:implementedAs` is attached to the `fipa:Rule`
560 class, as the property `implementedBy` was attached to a `fipa:Action` class.

561
562 The following example states that "for all cars for which the property speed exceeds 200 (km/h), the property category
563 should be set to `race-car`:"

```

564
565 <?xml version="1.0"?>
566 <rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#"
567     xmlns:car="http://www.cars.org/schemas#"
568     xmlns="http://www.fipa.org/schemas/fipa-rdf1#">
569
570     <fipa:Rule rdf:ID="categorizeCars2">
571         <fipa:implementedAs>
572             <fipa:Code>
573                 <fipa:language>ECMAScript</fipa:language>
574                 <fipa:binding>3AP</fipa:binding>
575                 <fipa:script>
576                     NodeSelection selection = new NodeSelection("");
577                     Iterator it = selection.iterator();
578
579                     while (it.hasNext()) {
580                         Node n = (Node)it.next();
581
582                         if ((n.getProperty("speed").getValue() > 200) &
583                             (n.getProperty("type").getValue().
584                             equals("http://www.cars.org/schemas#Car"))) {
585                             n.getProperty("category").setValue("race-car");
586                         }
587                     }
588                 </fipa:script>
589             </fipa:Code>
590         </fipa:implementedAs>
591     </fipa:Rule>
592 </rdf:RDF>
593

```

594 This script uses the 3AP APIs to map the RDF data with the Java object model.
595

596 3.4 Using Rules with FIPA Communicative Acts

597 An agent may request another agent to fire a specific rule to his knowledge base.

```
598
599 (request
600   :sender i
601   :receiver j
602   :content (
603     <?xml version="1.0"?>
604     <rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#"
605             xmlns="http://www.fipa.org/schemas/fipa-rdf1#">
606
607       <FireRule>
608         <rdf:type rdf:resource="http://www.fipa.org/schemas#Action"/>
609         <argument rdf:resource="#categorizeCars2">
610       </FireRule>
611     </rdf:RDF> )
612   :language fipa-rdf1 )
613
```

614 The rules engine will then have an impact on the properties of all car instances.

615
616 Another use is that an agent informs another agent about its (implicit) belief in the correctness of a rule:

```
617 (inform
618   :sender i
619   :receiver j
620   :content (
621     <?xml version="1.0"?>
622     <rdf:RDF xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#"
623             xmlns="http://www.fipa.org/schemas/fipa-rdf1#">
624
625       <fipa:Rule about="#categorizeCars2"/>
626     </rdf:RDF> )
627   :language fipa-rdf0)
628
629
```

630 The receiving agent may then decide to apply the rule (or not).

631

632 3.5 Further Remarks

633 In practice, the RDF content in a FIPA message may look quite verbose. However, this problem can be tackled in
634 different ways:

635

636 The RDF specification itself has been foreseen in a number of alternative 'abbreviated forms'.

637

638 Binary encodings can be used instead, as defined by the XML Token specification (see [W3Cxml]).

639

640 Some parts of the content can be defined in advance by unique XML identifiers (URIs) and then used in subsequent
641 messages. This may be especially useful when the negotiation focuses only on one specific service parameter.

642

643 To support the latter mechanism of cross-referencing parts of the RDF content, we suggest the usage of the `query-`
644 `ref` and `inform` (see [FIPA00046]) FIPA communicative acts.

645

645 4 Examples of Use

646 A number of companies and organisations in the FACTS project (see [FACTS]) have used FIPA RDF as content
647 language for agent-based provisioning of virtual private networks.
648

649 4.1 RDF Schemas for FIPA RDF 0

650 The RDF schema needed for using `fipa-rdf0` (for expressing actions and propositions) is as follows:

```
651 <?xml version="1.0"?>
652 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
653     xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#">
654
655     <rdfs:Class rdf:ID="Proposition">
656         <rdfs:label xml:lang="en">proposition</rdfs:label>
657         <rdfs:label xml:lang="fr">proposition</rdfs:label>
658         <rdfs:subClassOf rdf:resource=
659             "http://www.w3c.org/1999/02/22-rdf-syntax-ns#Statement"/>
660         <rdfs:comment>This describes the set of propositions</rdfs:comment>
661     </rdfs:Class>
662
663     <rdfs:ConstraintProperty rdf:ID="belief">
664         <rdfs:label xml:lang="en">belief</rdfs:label>
665         <rdfs:label xml:lang="fr">acte</rdfs:label>
666         <rdfs:domain rdf:resource="#Proposition"/>
667         <rdfs:range rdf:resource=
668             "http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
669     </rdfs:ConstraintProperty>
670
671     <rdfs:Class rdf:ID="Action">
672         <rdfs:label xml:lang="en">action</rdfs:label>
673         <rdfs:label xml:lang="fr">action</rdfs:label>
674         <rdfs:subClassOf rdf:resource=
675             "http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Resource"/>
676         <rdfs:comment>This describes the set of actions</rdfs:comment>
677     </rdfs:Class>
678
679     <rdfs:ConstraintProperty rdf:ID="act">
680         <rdfs:label xml:lang="en">act</rdfs:label>
681         <rdfs:label xml:lang="fr">acte</rdfs:label>
682         <rdfs:domain rdf:resource="#Action"/>
683     </rdfs:ConstraintProperty>
684
685     <rdfs:ConstraintProperty rdf:ID="actor">
686         <rdfs:label xml:lang="en">actor</rdfs:label>
687         <rdfs:label xml:lang="fr">acteur</rdfs:label>
688         <rdfs:domain rdf:resource="#Action"/>
689     </rdfs:ConstraintProperty>
690
691     <rdfs:ConstraintProperty rdf:ID="argument">
692         <rdfs:label xml:lang="en">argument</rdfs:label>
693         <rdfs:label xml:lang="fr">argument</rdfs:label>
694         <rdfs:domain rdf:resource="#Action"/>
695     </rdfs:ConstraintProperty>
696
697     <rdfs:ConstraintProperty rdf:ID="done">
698         <rdfs:label xml:lang="en">done</rdfs:label>
699         <rdfs:label xml:lang="fr">fini</rdfs:label>
700         <rdfs:domain rdf:resource="#Action"/>
701     </rdfs:ConstraintProperty>
702 </rdf:RDF>
703
```



```

704 <rdfs:ConstraintProperty rdf:ID="result">
705   <rdfs:label xml:lang="en">result</rdfs:label>
706   <rdfs:label xml:lang="fr">resultat</rdfs:label>
707   <rdfs:domain rdf:resource="#Action"/>
708 </rdfs:ConstraintProperty>
709
710 <rdfs:ConstraintProperty rdf:ID="implementedBy">
711   <rdfs:label xml:lang="en">implementedBy</rdfs:label>
712   <rdfs:label xml:lang="fr">implemente par</rdfs:label>
713   <rdfs:domain rdf:resource="#Action"/>
714 </rdfs:ConstraintProperty>
715
716 <rdfs:Class rdf:ID="Code">
717   <rdfs:label xml:lang="en">code</rdfs:label>
718   <rdfs:label xml:lang="fr">code</rdfs:label>
719   <rdfs:comment>This describes the code implementation</rdfs:comment>
720 </rdfs:Class>
721
722 <rdfs:ConstraintProperty rdf:ID="language">
723   <rdfs:label xml:lang="en">language</rdfs:label>
724   <rdfs:label xml:lang="fr">langue</rdfs:label>
725   <rdfs:domain rdf:resource="#Code"/>
726   <rdfs:range rdf:resource=
727     "http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
728 </rdfs:ConstraintProperty>
729
730 <rdfs:ConstraintProperty rdf:ID="binding">
731   <rdfs:label xml:lang="en">binding</rdfs:label>
732   <rdfs:label xml:lang="fr">binding</rdfs:label>
733   <rdfs:domain rdf:resource="#Code"/>
734   <rdfs:range rdf:resource=
735     "http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
736 </rdfs:ConstraintProperty>
737
738 <rdfs:ConstraintProperty rdf:ID="code-uri">
739   <rdfs:label xml:lang="en">code-uri</rdfs:label>
740   <rdfs:label xml:lang="fr">code-uri</rdfs:label>
741   <rdfs:domain rdf:resource="#Code"/>
742   <rdfs:range rdf:resource=
743     "http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
744 </rdfs:ConstraintProperty>
745
746 <rdfs:ConstraintProperty rdf:ID="script">
747   <rdfs:label xml:lang="en">script</rdfs:label>
748   <rdfs:label xml:lang="fr">script</rdfs:label>
749   <rdfs:domain rdf:resource="#Code"/>
750   <rdfs:range rdf:resource=
751     "http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Literal"/>
752 </rdfs:ConstraintProperty>
753 </rdf:RDF>
754

```

755 4.2 RDF Schemas for FIPA RDF 1

756 The RDF schemas corresponding to fipa-rdf1 are specified as follows (extending the above schemas):

```

757
758 <?xml version="1.0"?>
759 <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
760   xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
761   xmlns:fipa="http://www.fipa.org/schemas/fipa-rdf0#">
762
763   <rdfs:Class rdf:ID="Rule">
764     <rdfs:label xml:lang="en">rule</rdfs:label>
765     <rdfs:label xml:lang="fr">regle</rdfs:label>

```

```
766 </rdfs:Class>
767
768 <rdfs:ConstraintProperty rdf:ID="selection">
769   <rdfs:comment>The selection part </rdfs:comment>
770   <rdfs:domain rdf:resource="Rule"/>
771 </rdfs:ConstraintProperty>
772
773 <rdfs:ConstraintProperty rdf:ID="manipulation">
774   <rdfs:comment>The manipulation part</rdfs:comment>
775   <rdfs:domain rdf:resource="Rule"/>
776 </rdfs:ConstraintProperty>
777
778 <rdfs:ConstraintProperty rdf:ID="selection-result">
779   <rdfs:comment>
780     Identifies the container filled with selection results
781   </rdfs:comment>
782   <rdfs:domain rdf:resource="Rule"/>
783   <rdfs:range rdf:resource=
784     "http://www.w3c.org/TR/1999/PR-rdf-schema-19990303#Bag"/>
785 </rdfs:ConstraintProperty>
786
787 <rdfs:ConstraintProperty rdf:ID="implementedAs">
788   <rdfs:label xml:lang="en">implemented as</rdfs:label>
789   <rdfs:label xml:lang="fr">implemente comme</rdfs:label>
790   <rdfs:domain rdf:resource="Rule"/>
791 </rdfs:ConstraintProperty>
792 </rdf:RDF>
793
```

793 **5 References**

- 794 [DATAX] DATAX: Data Exchange in XML. 1999.
795 <http://www.megginson.com/DATAX/>
- 796 [Decker98] A Query and Inference Service for RDF, Decker, S, Brickley, D, Saarela, J and Angele, J. 1998.
797 <http://www.ilrt.bris.ac.uk/discovery/rdf-dev/purls/papers/QL98->
798 [queryservice/](http://www.ilrt.bris.ac.uk/discovery/rdf-dev/purls/papers/QL98-queryservice/)
- 799 [FACTS] FIPA Agent Communication Technologies and Services (FACTS).
800 <http://www.labs.bt.com/profsoc/facts/>
- 801 [FIPA00046] FIPA Inform Communicative Act Specification. Foundation for Intelligent Physical Agents, 2000.
802 <http://www.fipa.org/specs/fipa00046/>
- 803 [FIPA00054] FIPA Query Ref Communicative Act Specification. Foundation for Intelligent Physical Agents, 2000.
804 <http://www.fipa.org/specs/fipa00054/>
- 805 [Melnik99] Generic Interoperability Framework (GINF) Working Paper, Melnik, S. Stanford University, 1999.
- 806 [W3Crdf] Status for Resource Description Framework (RDF) Model and Syntax Specification (Proposed
807 Recommendation). World Wide Web Consortium, 1999.
808 <http://www.w3.org/TR/REC-rdf-syntax/>
- 809 [W3Crdfquery] RDF Query Specification (Technical Contribution). World Wide Web Consortium, 1998.
810 <http://www.w3.org/TandS/QL/QL98/pp/rdfquery.html>
- 811 [W3Crdfs] Resource Description Framework (RDF) Schema Specification 1.0 (Candidate Recommendation).
812 World Wide Web Consortium, 2000.
813 <http://www.w3.org/TR/rdf-schema/>
- 814 [W3Cxml] Extensible Markup Language (XML) 1.0 Specification (Recommendation). World Wide Web
815 Consortium, 1998.
816 <http://www.w3c.org/TR/REC-xml/>