

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Agent Management Specification

Document title	FIPA Agent Management Specification					
Document number	SC000XX	Document source	FIPA TC Agent Management			
Document status	Standard	Date of this status	2004/18/03			
Supersedes	FIPA00002, FIPA00017, FIPA00019					
Contact	fab@fipa.org					
Change history	<p>See <i>Informative Annex B — Control Flow Example</i></p> <p>To further clarify the reference model for discovery in ad hoc networks described in section 5.1.1, an example will be given to explain how the DF and its wrapped DMs work together. Figure 2 outlines the control flow of a DM based discovery process of the DF. (Figure 4 abbreviates a df-agent-description as 'DAD'.)</p>					
<pre>graph LR; Agent((Agent)) -- "Search DAD Template" --> DF[DF]; DF -- "Search DAD Template" --> DM1[DM 1]; DF -- "Search DAD Template" --> DMn[DM n]; DM1 -- "Results mapped DAD" --> DF; DMn -- "Results mapped DAD" --> DF; DF -- "Results DAD" --> Agent;</pre>						
<p>Figure 4: Control Flow of a DM based Discovery within the DF</p> <p>For searching a certain agent service, first the agent must compose a df-agent-description search template describing the service to search.</p> <p>To find the service in one or more ad hoc networks, the agent uses the search functionality of the DF by providing the composed df-agent-description search template. Eventually the agent also specifies certain DM(s) (via the <code>dms</code> slot in search constraints object), which should be used for this search. This can be done after the agent has queried the DF for all currently available DMs. Otherwise, the DF will use all available DMs by default.</p> <p>The DF now uses the search functionality of each available DM, in turn by providing the df-agent-description search template.</p> <p>Each invoked DM maps the df-agent-description search template to the appropriate representation of the corresponding technology and performs the</p>						

search in the corresponding ad hoc network.

All available df-agent-descriptions, which have previously been registered with the DF, are represented in the DM's appropriate form and are discoverable by agents on remote devices.

The results of such a search are matching df-agent-descriptions in the appropriate representation of the DM. The DM is now mapping the results back to real df-agent-descriptions and returns them to the DF.

The DF is collecting the resulting df-agent-descriptions of all invoked DMs and is returning them to the calling agent.

There are also cases where the DM is not able to provide a template based search. In that cases the DF has to take over the missing functionality.

1 Informative Annex C — Discovery Middleware

Various technologies exist or will appear, which provide discovery in ad hoc networks. Examples of such DM technologies are JXTA and Bluetooth.

Each DM maintained by the DF must be described in a FIPA specification of its own. In detail, such a DM specification must describe how the DM functionality, which is imposed by the offered DF functions described in section 7.2, can be realised on the basis of the respective technology.

This specification does not specify how the actual interface between the DF and its DMs must look like. However, functionality details of a DM, which are important to consider during DM specification, are summarized next. There are also cases where the DM is not able to provide that functionality completely. In that cases the DF has to take over the missing functionality.

- **DM ID:** The DF must be able to retrieve a DM's ID string. This enables the DF to differentiate between several DMs and to provide this information to the agents.
- **Startup and Shutdown:** The DF must be able to start up and shut down a DM at runtime.
- **Register and Deregister:** The DF must be able to register df-agent-descriptions within a DM and to deregister df-agent-descriptions from the DM.

The lease-time parameter of the df-agent-description should be used by a DM to determine the lifetime of the df-agent-description. If no lease-time parameter exists in the df-agent-description, the lifetime should be assumed unlimited. In this case, it is important to deregister the df-agent-description later on, in order to save system resources.

If the df-agent-description has already been registered, its lease-time should be renewed according to the value of the lease-time parameter.

- **Search:** The DF must be able to search for df-agent-descriptions within a DM on the basis of a df-agent-description search template. The matching criterion to determine the set of objects that satisfy the search criteria must exactly be the same as specified for the search function in section 7.2.4.1.

	<p>As a means of both limiting the network load as well as the processing load of a device within an ad hoc network, the max-results parameter of the search-constraints frame (see section 7.1.4) should be used to constrain the number of returned results per agent platform.</p> <ul style="list-style-type: none"> • Subscribe and Unsubscribe: The DF must be able to subscribe for and unsubscribe from <code>df-agent-descriptions</code>, which match a <code>df-agent-description</code> search template, within a DM. The matching criterion to determine the set of objects that satisfy the search criteria must exactly be the same as specified for the search function in section 7.2.4.1.
Informative Annex D — ChangeLog	

7

8

9

10

11

12

13

14

15

16

17

18 © 1996-2002 Foundation for Intelligent Physical Agents

19 <http://www.fipa.org/>

20 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.

21 **Foreword**

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

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

32 The FIPA specifications are developed through direct involvement of the FIPA membership. The status of a
33 specification can be either Preliminary, Experimental, Standard, Deprecated or Obsolete. More detail about the
34 process of specification may be found in the FIPA Document Policy [f-out-00000] and the FIPA Specifications Policy [f-
35 out-00003]. A complete overview of the FIPA specifications and their current status may be found on the FIPA Web
36 site.

37 FIPA is a non-profit association registered in Geneva, Switzerland. As of June 2002, the 56 members of FIPA
38 represented many countries worldwide. Further information about FIPA as an organization, membership information,
39 FIPA specifications and upcoming meetings may be found on the FIPA Web site at <http://www.fipa.org/>.

40 Contents

41	1 Scope	1
42	2 Agent Management Reference Model	2
43	3 Agent Naming.....	4
44	3.1 Transport Addresses.....	4
45	3.2 Name Resolution.....	4
46	4 Agent Management Services	6
47	4.1 Directory Facilitator	6
48	4.1.1 Overview	6
49	4.1.2 Management Functions Supported by the Directory Facilitator	7
50	4.1.3 Federated Directory Facilitators.....	8
51	4.1.4 Subscribing and Unsubscribing with the DF	8
52	4.2 Agent Management System.....	10
53	4.2.1 Overview	10
54	4.2.2 Management Functions Supported by the Agent Management System.....	10
55	4.3 Message Transport Service	11
56	5 Agent Platform.....	12
57	5.1 Agent Life Cycle	12
58	5.2 Agent Registration.....	13
59	5.2.1 Registration Lease Times	14
60	6 Agent Management Ontology.....	16
61	6.1 Object Descriptions	16
62	6.1.1 Agent Identifier Description.....	16
63	6.1.2 Directory Facilitator Agent Description	17
64	6.1.3 Service Description	18
65	6.1.4 Search Constraints	18
66	6.1.5 Agent Management System Agent Description	19
67	6.1.6 Agent Platform Description	19
68	6.1.7 Agent Service Description	19
69	6.1.8 Property Template	19
70	6.2 Function Descriptions.....	20
71	6.2.1 Registration of an Object with an Agent	20
72	6.2.2 Deregistration of an Object with an Agent	20
73	6.2.3 Modification of an Object Registration with an Agent.....	21
74	6.2.4 Search for an Object Registration with an Agent.....	21
75	6.2.5 Retrieve an Agent Platform Description.....	23
76	6.3 Exceptions.....	23
77	6.3.1 Exception Selection	23
78	6.3.2 Exception Classes	24
79	6.3.3 Not Understood Exception Predicates.....	24
80	6.3.4 Refusal Exception Propositions	24
81	6.3.5 Failure Exception Propositions	25
82	7 Agent Management Content Language	26
83	8 References	27
84	9 Informative Annex A — Dialogue Examples	28
85	10 Informative Annex B — Control Flow Example	35
86	11 Informative Annex C — Discovery Middleware	36
87	12 Informative Annex D — ChangeLog	37
88	12.1 2001/10/03 - version H by FIPA Architecture Board	37
89	12.2 2002/11/01 - version I by TC X2S	37
90	12.3 2002/12/03 - version J by FIPA Architecture Board	38
91	12.4 2004/03/18 - version K by TC Ad hoc	38
92		

92 2 Scope

93 This document is part of the FIPA specifications covering agent management for inter-operable agents. This
94 specification incorporates and further enhances [FIPA00002] and [FIPA00067] represents a companion specification.

95
96 This document contains specifications for agent management including agent management services, agent
97 management ontology and agent platform message transport, including the discovery of agents and their offered
98 services in ad hoc¹ networks. This document is primarily concerned with defining open standard interfaces for
99 accessing agent management services. The internal design and implementation of intelligent agents and agent
100 management infrastructure is not mandated by FIPA and is outside the scope of this specification.

101
102 The document provides a series of examples to illustrate the agent management functions defined.
103

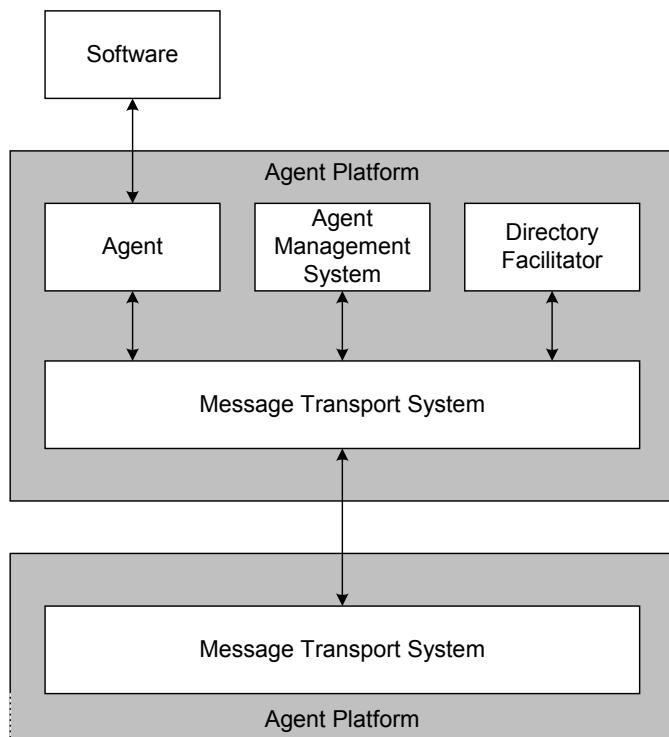
104

¹ Ad hoc networks may emerge in a fixed or mobile network infrastructure. They are formed for the duration of a communication session or, in the case of mobile devices, while in close proximity.

104 3 Agent Management Reference Model

105 Agent management provides the normative framework within which FIPA agents exist and operate. It establishes the
 106 logical reference model for the creation, registration, location, communication, migration and retirement of agents.
 107

108 The entities contained in the reference model (see *Figure 1*) are logical capability sets (that is, services) and do not
 109 imply any physical configuration. Additionally, the implementation details of individual APs and agents are the design
 110 choices of the individual agent system developers.
 111



112
 113
114 Figure 1: Agent Management Reference Model
 115

116 The agent management reference model consists of the following logical components², each representing a capability
 117 set (these can be combined in physical implementations of APs):
 118

- 119 • An **agent** is a computational process that implements the autonomous, communicating functionality of an
 120 application. Agents communicate using an Agent Communication Language. An Agent is the fundamental actor on
 121 an AP which combines one or more service capabilities, as published in a service description, into a unified and
 122 integrated execution model. An agent must have at least one owner, for example, based on organisational
 123 affiliation or human user ownership, and an agent must support at least one notion of identity. This notion of
 124 identity is the Agent Identifier (AID) that labels an agent so that it may be distinguished unambiguously within the
 125 Agent Universe. An agent may be registered at a number of transport addresses at which it can be contacted.
 126
- 127 • A **Directory Facilitator (DF)** is an optional component of the AP, but if it is present, it must be implemented as a
 128 DF service (see Section 5.1). The DF provides yellow pages services to other agents. Agents may register their
 129 services with the DF or query the DF to find out what services are offered by other agents, including the discovery
 130 of agents and their offered services in ad hoc networks. Multiple DFs may exist within an AP and may be
 131 federated. The DF is a reification of the Agent Directory Service in [FIPA00001].
 132

² The functionalities of these components are a specialization of the AA notion of Service [see FIPA00001].

133 • An **Agent Management System (AMS)** is a mandatory component of the AP. The AMS exerts supervisory control
134 over access to and use of the AP. Only one AMS will exist in a single AP. The AMS maintains a directory of AIDs
135 which contain transport addresses (amongst other things) for agents registered with the AP. The AMS offers white
136 pages services to other agents. Each agent must register with an AMS in order to get a valid AID. The AMS is a
137 reification of the Agent Directory Service in [FIPA00001].
138

139 • An **Message Transport Service (MTS)** is the default communication method between agents on different APs
140 (see [FIPA00067]).
141

142 • An **Agent Platform (AP)** provides the physical infrastructure in which agents can be deployed. The AP consists of
143 the machine(s), operating system, agent support software, FIPA agent management components (DF, AMS and
144 MTS) and agents.
145

146 The internal design of an AP is an issue for agent system developers and is not a subject of standardisation within
147 FIPA. AP's and the agents which are native to those APs, either by creation directly within or migration to the AP,
148 may use any proprietary method of inter-communication.
149

150 It should be noted that the concept of an AP does not mean that all agents resident on an AP have to be co-
151 located on the same host computer. FIPA envisages a variety of different APs from single processes containing
152 lightweight agent threads, to fully distributed APs built around proprietary or open middleware standards.
153

154 FIPA is concerned only with how communication is carried out between agents who are native to the AP and
155 agents outside the AP. Agents are free to exchange messages directly by any means that they can support.
156

157 • **Software** describes all non-agent, executable collections of instructions accessible through an agent. Agents may
158 access software, for example, to add new services, acquire new communications protocols, acquire new security
159 protocols/algorithms, acquire new negotiation protocols, access tools which support migration, etc.
160

161 4 Agent Naming

162 The FIPA agent naming reference model identifies an agent through an extensible collection of parameter-value pairs³,
 163 called an Agent Identifier (AID). The extensible nature of an AID allows it to be augmented to accommodate other
 164 requirements, such as social names, nick names, roles, etc. which can then be attached to services within the AP. An
 165 AID comprises⁴ (see Section 7.1.1):
 166

- 167 • The name parameter, which is a globally unique identifier that can be used as a unique referring expression of the
 168 agent. One of the simplest mechanisms is to construct it from the actual name of the agent and its home agent
 169 platform address⁵ (HAP), separated by the @ character. This is a reification of the notion of an Agent Name from
 170 [FIPA00001].
 171
- 172 • The addresses parameter, which is a list of transport addresses where a message can be delivered (see Section
 173 4.1). This is a reification of the notion of a Locator from [FIPA00001].
 174
- 175 • The resolvers parameter, which is a list of name resolution service addresses (see Section 4.2).
 176

177 The parameter values of an AID can be edited or modified by an agent, for example, to update the sequence of name
 178 resolution servers or transport addresses in an AID. However, the mandatory parameters can only be changed by the
 179 agent to whom the AID belongs. AIDs are primarily intended to be used to identify agents inside the envelope of a
 180 transport message, specifically within the to and from parameters (see [FIPA00067]).
 181

182 Two AIDs are considered to be equivalent if their name parameters are the same.
 183

184 4.1 Transport Addresses

185 A transport address is a physical address at which an agent can be contacted and is usually specific to a Message
 186 Transport Protocol. A given agent may support many methods of communication and can put multiple transport
 187 address values in the addresses parameter of an AID.
 188

189 The EBNF syntax of a transport addresses is the same as for a URL given in [RFC2396]. [FIPA00067] describes the
 190 semantics of message delivery with regard to transport addresses.
 191

192 4.2 Name Resolution

193 Name resolution is a service that is provided by the AMS through the search function. The resolvers parameter of
 194 the AID contains a sequence of AIDs at which the AID of the agent can ultimately be resolved into a transport address
 195 or set of transport address.
 196

197 An example name resolution pattern might be:
 198

- 199 1. agent-a wishes to send a message to agent-b, whose AID is:

```
200
201 (agent-identifier
202   :name agent-b@bar.com
203   :resolvers (sequence
204     (agent-identifier
205       :name ams@foo.com
206       :addresses (sequence iiop://foo.com/acc))))
```

³ The name of additional parameters added to an AID and not defined by FIPA, must be prefixed with "x-" to avoid name conflict with any future extension of the standard.

⁴ The name of an agent is immutable and cannot be changed during the lifetime of the agent; the other parameters in the AID of an agent can be changed.

⁵ The HAP of an agent is the AP on which the agent was created.

208 and *agent-a* wishes to know additional transport addresses that have been given for *agent-b*.
209

- 210 2. Therefore, *agent-a* can send a search request to the first agent specified in the `resolvers` parameter which is
211 typically an AMS. In this example, the AMS at `foo.com`.
- 212 3. If the AMS at `foo.com` has *agent-b* registered with it, then it returns a result message containing the AMS agent
214 description of *agent-b*; if not, then a failed message is returned.
- 215 4. Upon receipt of the result message, *agent-a* can extract the `agent-identifier` parameter of the `ams-`
217 *agent-description* and then extract the `addresses` parameter of this to determine the transport address(es)
218 of *agent-b*.
- 219 5. *agent-a* can now send a message to *agent-b* by inserting the `addresses` parameter into the AID of *agent-b*.
- 221

222

222 5 Agent Management Services

223 5.1 Directory Facilitator

224 5.1.1 Overview

225 A DF is a component of an AP that provides a yellow pages directory service to agents. It is the trusted, benign
226 custodian of the agent directory. It is trusted in the sense that it must strive to maintain an accurate, complete and
227 timely list of agents. It is benign in the sense that it must provide the most current information about agents in its
228 directory on a non-discriminatory basis to all authorised agents. The DF is an optional component of an AP. However,
229 an AP may support any number of DFs and DFs may register with each other to form federations.

230
231 Every agent that wishes to publicise its services to other agents, should find an appropriate DF and request the
232 **registration** of its agent description. There is no intended future commitment or obligation on the part of the registering
233 agent implied in the act of registering. For example, an agent can refuse a request for a service which is advertised
234 through a DF. Additionally, the DF cannot guarantee the validity or accuracy of the information that has been
235 registered with it, neither can it control the life cycle of any agent. An object description must be supplied containing
236 values for all of the mandatory parameters of the description. It may also supply optional and private parameters,
237 containing non-FIPA standardised information that an agent developer might want included in the directory. The
238 **deregistration** function has the consequence that there is no longer a commitment on behalf of the DF to broker
239 information relating to that agent. At any time, and for any reason, the agent may request the DF to **modify** its agent
240 description.

241
242 An agent may **search** in order to request information from a DF. The DF does not guarantee the validity of the
243 information provided in response to a search request, since the DF does not place any restrictions on the information
244 that can be registered with it. However, the DF may restrict access to information in its directory and will verify all
245 access permissions for agents which attempt to inform it of agent state changes.

246
247 The default DF on an AP, if present, has a reserved AID of:

248
249 (agent-identifier
250 :name df@hap_name⁶
251 :addresses (sequence hap_transport_address))

252
253 The DF also provides discovery functionality in ad hoc networks, in which network nodes may frequently join and leave
254 and DF federations as described in section 5.1.3 may not be feasible. It provides a high-level interface for agents,
255 which want to become discoverable for agents on remote devices, thereby hiding the possible usage of various
256 discovery middleware (DM) depending on the underlying ad hoc technology. Figure 2 outlines a possible model of a
257 DF wrapping various DMs. [FIPA00095] specifies a DM based on JXTA peer-to-peer technology..

⁶ The *hap_name* should be replaced with the name of the HAP that is published in the *ap-description*.

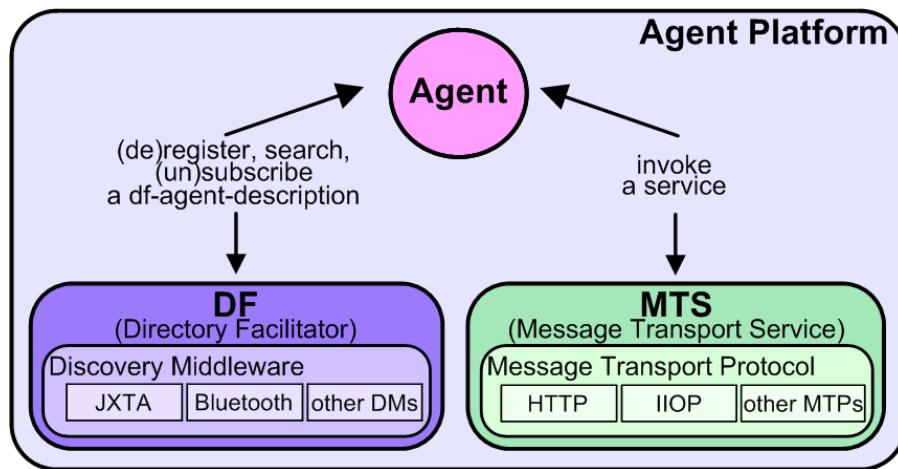


Figure 2: Reference Model of the Discovery Process in Ad Hoc Networks

5.1.2 Management Functions Supported by the Directory Facilitator

In order to access the directory of agent descriptions managed by the DF, each DF must be able to perform the following functions, when defined on the domain of objects of type `df-agent-description` in compliance with the semantics described in Section 7.1.2:

- register
- deregister
- modify
- search

The `fipa-request` interaction protocol [FIPA00026] must be used by agents wishing to request a DF to perform these actions.

A DF may support the following extended directory mechanism:

- subscribe mechanism

For the subscribe mechanism a DF must implement the `fipa-subscribe` interaction protocol [FIPA00035] in order to allow agents to subscribe for being notified about registration, deregistration and modifications of certain agent descriptions. If implemented, the implementation of this protocol must comply with the semantics and syntax specified in section 4.1.4.

To allow agents to get the configuration information about the DF, a DF may support an introspection functionality. In order to get that information an agent may ask the DF for its `df-agent-description` (the agent description of the DF). The `df-agent-description` contains beside other information a services slot which can contain a set of

292 service descriptions. Each of the service descriptions may contain information such as the ability of the DF to publish
 293 and search offered agent services in underlaying DMs (see type slot in section 7.1.3)⁷

294 5.1.3 Federated Directory Facilitators

295 The DF encompasses a search mechanism that searches first locally and then extends the search to other DFs, if
 296 allowed.⁹ The default search mechanism is assumed to be a depth-first search across DFs. For specific purposes,
 297 optional constraints can be used as described in Section 0 such as the number of answers (`max-results`). The
 298 federation of DFs for extending searches can be achieved by DFs registering with each other with `fipa-df` as the
 299 value of the `type` parameter in the `service-description`.

300
 301 When a DF receives a search action, it may determine whether it needs to propagate this search to other DFs that are
 302 registered with it¹⁰. It should only forward searches where the value of the `max-depth` parameter is greater than 1 and
 303 where it has not received a prior search with the same `search-id` parameter. If it does forward the search action,
 304 then it must use the following rules:

- 305
 306 1. It must not change the value of the `search-id` parameter when it propagates the search and the value of all
 307 `search-id` parameters should be globally unique.
- 308
 309 2. Before propagation, it should decrement the value of the `max-depth` parameter by 1.

310 5.1.4 Subscribing and Unsubscribing with the DF

311 Some DFs may implement the `fipa-subscribe` interaction protocol [FIPA00035] in order to allow agents to
 312 subscribe for being notified about registration, deregistration and modifications of certain agent descriptions.

313 A DF that does not support such a functionality is simply required to respond with a `not-understood` communicative
 314 act with `unsupported-act` as content of the communicative act (see also 7.3.3).

315 Further to what specified in [FIPA00035], the usage of the `fipa-subscribe` interaction protocol must obey to the
 316 following rules:

- 317 - the content of the `subscribe` communicative act must be a referential expression denoting a persistent
 318 search action. For simplicity, the following can be used
 319 " (

```
320   (action
321     (agent-identifier
322       :name df@foo.com
323       :addresses (sequence iiop://foo.com/acc)
324     )
325   (search
326     (df-agent-description
327       :ontologies (set meeting-scheduler)
328       :languages (set fipa-s10 kif)
329       :services (set
330         (service-description
331           :name profiling
332           :type meeting-scheduler-service)
333       )
334       (search-constraints :max-depth 2)))
335   )"
```

336 to denote (and be understood as):

```
337  "((iota ?x
338    (result
339      (action
340        (agent-identifier
341          :name df@foo.com
342          :addresses (sequence iiop://foo.com/acc))
```

⁷ In the case that a DF wraps some DMs, an agent should be able to apply all described mechanisms either to all available DMs or to a subset. (by usage of the `dms` slot of the `search-constraints` object (section 7.1.4) or the `scope` slot of the `df-agent-description` object (section 7.1.2)).

⁸

9 DF federations may not be efficient enough for ad hoc networks. In this case appropriate DF DMs should be introduced.

¹⁰ Some DFs may not support federated search, in which case the `max-result`, `max-depth` and `search-id` parameters have no effect.

```

343      )
344      (search
345        (df-agent-description
346          :ontologies (set meeting-scheduler)
347          :languages (set fipa-s10 kif)
348          :services (set
349            (service-description
350              :name profiling
351              :type meeting-scheduler-service))
352          )
353          (search-constraints :max-depth 2)))
354        ?x)))"
355 - the DF must continue to send an inform communicative act as the objects denoted by the referring expression
356 change, i.e. as the result of the persistent search changes. For simplicity, the following can be used:
357 "
358   ((result
359     (action
360       (agent-identifier
361         :name df@foo.com
362         :addresses (sequence iiop://foo.com/acc)))
363     (search
364       (df-agent-description
365         :ontologies (set meeting-scheduler)
366         :languages (set fipa-s10 kif)
367         :services (set
368           (service-description
369             :name profiling
370             :type meeting-scheduler-service))
371         (search-constraints :max-depth 2)))
372     (set
373       (df-agent-description
374         :name
375           (agent-identifier
376             :name scheduler-agent@foo.com
377             :addresses (sequence iiop://foo.com/acc))
378           :ontologies (set meeting-scheduler fipa-agent-management)
379           :languages (set fipa-s10 fipa-s11 kif)
380           :services (set
381             (service-description
382               :name profiling
383               :type meeting-scheduler-service)
384             (service-description
385               :name profiling
386               :type user-profiling-service)))))))))

387 to denote (and be understood as):
388 "
389   (= (iota ?x
390     (result
391       (action
392         (agent-identifier
393           :name df@foo.com
394           :addresses (sequence iiop://foo.com/acc)
395         )
396       (search
397         (df-agent-description
398           :ontologies (set meeting-scheduler)
399           :languages (set fipa-s10 kif)
400           :services (set
401             (service-description
402               :name profiling
403               :type meeting-scheduler-service))
404             )
405             (search-constraints :max-depth 2)))
406           ?x))
407       (set
408         (df-agent-description
409           :name
410             (agent-identifier

```

```

410           :name scheduler-agent@foo.com
411           :addresses (sequence iiop://foo.com/acc))
412           :ontologies (set meeting-scheduler fipa-agent-management)
413           :languages (set fipa-s10 fipa-s11 kif)
414           :services (set
415             (service-description
416               :name profiling
417               :type meeting-scheduler-service)
418             (service-description
419               :name profiling
420               :type user-profiling-service))))"
421
422
423 A subscription is terminated by a cancel act as specified in [FIPA00035].
424

```

425 5.2 Agent Management System

426 5.2.1 Overview

427 An AMS is a mandatory component of the AP and only one AMS will exist in a single AP. The AMS is responsible for
 428 managing the operation of an AP, such as the creation of agents, the deletion of agents and overseeing the migration
 429 of agents to and from the AP (if agent mobility is supported by the AP). Since different APs have different capabilities,
 430 the AMS can be queried to obtain a description of its AP. A life cycle is associated with each agent on the AP (see
 431 Section 6.1) which is maintained by the AMS.

432
 433 The AMS represents the managing authority of an AP and if the AP spans multiple machines, then the AMS represents
 434 the authority across all machines. An AMS can request that an agent performs a specific management function, such
 435 as quit (that is, terminate all execution on its AP) and has the authority to forcibly enforce the function if such a
 436 request is ignored.

437
 438 The AMS maintains an index of all the agents that are currently resident on an AP, which includes the AID of agents.
 439 Residency of an agent on the AP implies that the agent has been registered with the AMS. Each agent, in order to
 440 comply with the FIPA reference model, must **register** with the AMS of its HAP.

441
 442 Agent descriptions can be later **modified** at any time and for any reason. Modification is restricted by authorisation of
 443 the AMS. The life of an agent with an AP terminates with its **deregistration** from the AMS. After deregistration, the AID
 444 of that agent can be removed by the directory and can be made available to other agents who should request it.

445
 446 Agent description can be **searched** with the AMS and access to the directory of **ams-agent-descriptions** is
 447 further controlled by the AMS; no default policy is specified by this specification. The AMS is also the custodian of the
 448 AP description that can be retrieved by requesting the action **get-description**.

449
 450 The AMS on an AP has a reserved AID of:

```

451
452 (agent-identifier
453   :name ams@hap_name11
454   :addresses (sequence hap_transport_address))
455

```

456 The name parameter of the AMS (**ams@hap_name**) is considered to be the Service Root of the AP (see [FIPA00001]).

458 5.2.2 Management Functions Supported by the Agent Management System

459 An AMS must be able to perform the following functions, in compliance with the semantics described in Section 0 (the
 460 first four functions are defined within the scope of the AMS, only on the domain of objects of type **ams-agent-**
 461 **description** and the last on the domain of objects of type **ap-description**):

¹¹ The **hap_name** should be replaced with the name of the HAP that is published in the **ap-description**.

- 463 • register
- 464 • deregister
- 465 • modify
- 466 • search
- 467 • get-description

472
473 In addition to the management functions exchanged between the AMS and agents on the AP, the AMS can instruct the
474 underlying AP to perform the following operations:

- 475 • Suspend agent,
- 476 • Terminate agent,
- 477 • Create agent,
- 478 • Resume agent execution,
- 479 • Invoke agent,
- 480 • Execute agent, and,
- 481 • Resource management.

490 **5.3 Message Transport Service**

491 The Message Transport Service (MTS) delivers messages between agents within an AP and to agents that are
492 resident on other APs. All FIPA agents have access to at least one MTS and only messages addressed to an agent
493 can be sent to the MTS. See [FIPA00067] for more information on the MTS.

494

495

495 6 Agent Platform

496 6.1 Agent Life Cycle

497 FIPA agents exist physically on an AP and utilise the facilities offered by the AP for realising their functionalities. In this
 498 context, an agent, as a physical software process, has a physical life cycle that has to be managed by the AP. This
 499 section describes a possible life cycle that can be used to describe the states which it is believed are necessary and
 500 the responsibilities of the AMS in these states.

501
 502 The life cycle of a FIPA agent is (see *Figure 3*):
 503

- 504 • **AP Bounded**

505 An agent is physically managed within an AP and the life cycle of a static agent is therefore always bounded to a
 506 specific AP.

- 507 • **Application Independent**

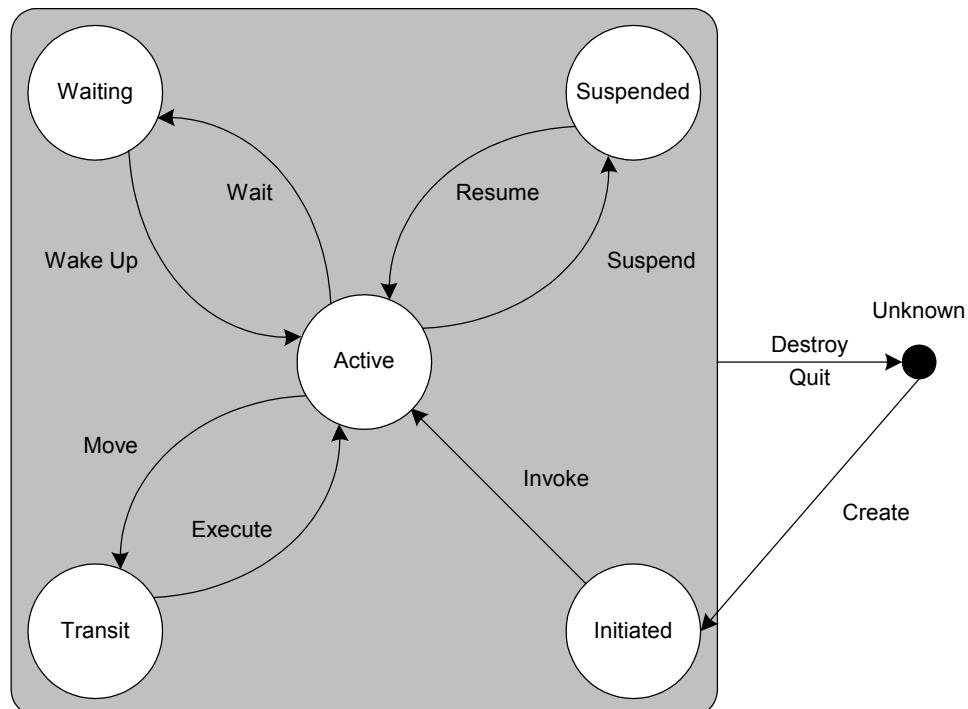
508 The life cycle model is independent from any application system and it defines only the states and the transitions of
 509 the agent service in its life cycle.

- 510 • **Instance-Oriented**

511 The agent described in the life cycle model is assumed to be an instance (that is, an agent which has unique name
 512 and is executed independently).

- 513 • **Unique**

514 Each agent has only one AP life cycle state at any time and within only one AP.
 515
 516



519
 520
 521 **Figure 3: Agent Life Cycle**
 522

523 The followings are the responsibility that an AMS, on behalf of the AP, has with regard to message delivery in each
 524 state of the life cycle of an agent:

- 525 • **Active**

527 The MTS delivers messages to the agent as normal.

528 • **Initiated/Waiting/Suspended**

529 The MTS either buffers messages until the agent returns to the active state or forwards messages to a new
530 location (if a forward is set for the agent).

531 • **Transit**

532 The MTS either buffers messages until the agent becomes active (that is, the move function failed on the original
533 AP or the agent was successfully started on the destination AP) or forwards messages to a new location (if a
534 forward is set for the agent). Notice that only mobile agents can enter the **Transit** state. This ensures that a
535 stationary agent executes all of its instructions on the node where it was invoked.

536 • **Unknown**

537 The MTS either buffers messages or rejects them, depending upon the policy of the MTS and the transport
538 requirements of the message.

539 The state transitions of agents can be described as:

540 • **Create**

541 The creation or installation of a new agent.

542 • **Invoke**

543 The invocation of a new agent.

544 • **Destroy**

545 The forceful termination of an agent. This can only be initiated by the AMS and cannot be ignored by the agent.

546 • **Quit**

547 The graceful termination of an agent. This can be ignored by the agent.

548 • **Suspend**

549 Puts an agent in a suspended state. This can be initiated by the agent or the AMS.

550 • **Resume**

551 Brings the agent from a suspended state. This can only be initiated by the AMS.

552 • **Wait**

553 Puts an agent in a waiting state. This can only be initiated by an agent.

554 • **Wake Up**

555 Brings the agent from a waiting state. This can only be initiated by the AMS.

556 The following two transitions are only used by mobile agents:

557 • **Move**

558 Puts the agent in a transitory state. This can only be initiated by the agent.

559 • **Execute**

560 Brings the agent from a transitory state. This can only be initiated by the AMS.

561 **6.2 Agent Registration**

562 There are three ways in which an agent can be registered with an AMS:

- 563 • The agent was created on the AP.

- 581
 582 • The agent migrated to the AP, for those APs which support agent mobility.
 583
 584 • The agent explicitly registered with the AP.

585
 586 Agent registration involves registering an AID with the AMS. When an agent is either created or registers with an AP,
 587 the agent is registered with the AMS, for example by using the `register` function. In the following example, an agent
 588 called *discovery-agent* is registering with an AP located at `foo.com`. The agent *discovery-agent* was created on the
 589 AP (that is, *discovery-agent's* HAP) at `bar.com` and requests that the AMS registers it.

590
 591 For example:

```
592
593 (request
594   :sender
595     (agent-identifier
596       :name discovery-agent@bar.com
597       :addresses (sequence iiop://bar.com/acc))
598   :receiver (set
599     (agent-identifier
600       :name ams@foo.com
601       :addresses (sequence iiop://foo.com/acc)))
602   :ontology fipa-agent-management
603   :language fipa-s10
604   :protocol fipa-request
605   :content
606     "((action
607       (agent-identifier
608         :name ams@foo.com
609         :addresses (sequence iiop://foo.com/acc))
610       register
611         (:ams-description
612           :name
613             (agent-identifier
614               :name discovery-agent@bar.com
615               :addresses (sequence iiop://bar.com/acc))
616             ...)))"
```

617
 618 It should be noted that the `addresses` parameter of the AID represents the transport address(es) that the agent would
 619 like any messages directed to (see [FIPA00067] for information on how the MTS deals with this). In the above
 620 example, the agent *discovery-agent* registers itself with the `foo.com` AP but by virtue of specifying a different
 621 transport address in the `addresses` parameter of its AID, messages that arrive at `foo.com` will be forwarded to
 622 `bar.com`.

623

624 6.2.1 Registration Lease Times

625 To enable the DF to manage a maintainable number of registrations over a long period of time, the DF may implement
 626 lease times using the `lease-time` parameter of a `df-agent-description`. A lease time is either a duration of
 627 time, such as 3 hours, or an absolute time, such as 08:00 26-Jul-2002, at which point a registration made by an agent
 628 can be removed from the DF registration database.

629

630 When an agent wishes to register with a DF, it can specify a lease time which is how long it would like the registration
 631 to be kept. If this lease time is okay for the DF, then it will accept the registration as usual and the value of the `lease-`
 632 `time` parameter in the content of the `inform` reply will be the same. Consequently, when the lease time expires, the
 633 registration will be silently removed by the DF. On the other hand, if the lease time is not acceptable to the DF, then
 634 the DF can include a *new* lease time as the value of the `lease-time` parameter in the content of the `inform` reply.
 635 This is the case when an agent does not specify a lease time in its registration.

636

637 If the DF does not support lease times, it will notify to the requesting agent that its registration is valid for an unlimited
 638 time by removing this parameter in the content of the `inform` reply, in fact the default lease-time is defined to be
 639 unlimited.

```

640
641 For example, an agent may register the following df-agent-description:
642
643 (request
644 ...
645 :content
646 "((action
647   (agent-identifier
648     :name df@foo.com
649     :addresses (sequence iiop://foo.com/acc))
650   (register
651     (df-agent-description
652       :name
653         (agent-identifier
654           :name dummy@foo.com
655           :addresses (sequence iiop://foo.com/acc))
656       :protocols fipa-request
657       :ontologies (set fipa-agent-management)
658       :languages (set fipa-s10)
659       :lease-time +00000000T600000000T
660       ...
661     )
662 Then if the DF agrees to this lease time, it will reply with an inform which contains the same value for the lease-
663 time parameter:
664
665 (inform
666 ...
667 :content
668 "((done
669   (action
670     (agent-identifier
671       :name df@foo.com
672       :addresses (sequence iiop://foo.com/acc))
673   (register
674     (df-agent-description
675       :name
676         (agent-identifier
677           :name dummy@foo.com
678           :addresses (sequence iiop://foo.com/acc))
679       :protocols (set fipa-request application-protocol)
680       :ontologies (set meeting-scheduler)
681       :languages (set fipa-s10 kif)
682       :lease-time +00000000T600000000T
683       ...
684
685 If an agent wishes to renew a lease time, then it can use the modify action to specify a new value for the lease-
686 time parameter. The verification of this lease time goes through the same procedure mentioned in the last paragraph:
687 if it is okay, then the value of the lease-time parameter in the content of the inform reply will be the same, if it is
688 not okay, the value of the lease-time parameter in the content of the inform reply will be a new value which is
689 acceptable to the DF.
690
691

```

691 7 Agent Management Ontology

692 7.1 Object Descriptions

693 This section describes a set of frames that represent the classes of objects in the domain of discourse within the
 694 framework of the fipa-agent-management ontology. The closure of symbols of this ontology can be obtained from
 695 [FIPA00067] that specifies additional set of frames of this ontology.

696
 697 This ontology does not specify any specific positional order to encode the parameters of the objects. Therefore, it is
 698 required to encode objects in SL by specifying both the parameter name and the parameter value (see Section 3.6 of
 699 [FIPA00008]).

700 The following terms are used to describe the objects of the domain:

- 703 • **Frame**. This is the mandatory name of this entity that must be used to represent each instance of this class.
- 704 • **Ontology**. This is the name of the ontology, whose domain of discourse includes the parameters described in the
 705 table.
- 707 • **Parameter**. This is the mandatory name of a parameter of this frame.
- 709 • **Description**. This is a natural language description of the semantics of each parameter.
- 711 • **Presence**. This indicates whether each parameter is mandatory or optional.
- 713 • **Type**. This is the type of the values of the parameter: Integer, Word, String, URL, Term, Set or Sequence.
- 715 • **Reserved Values**. This is a list of FIPA-defined constants that can assume values for this parameter.

718 7.1.1 Agent Identifier Description

719 This type of object represents the identification of the agent. The addresses parameter and the name resolution
 720 mechanism (see Section 4.2), is a reification of the notion of Locator from [FIPA00001]. See also Section 3.3.7 in FIPA
 721 Agent Message Transport Service [FIPA00067] specifications.

Frame Ontology	agent-identifier fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
name	The symbolic name of the agent.	Mandatory	word	df@hap_name ams@hap_name
addresses	A sequence of ordered transport addresses where the agent can be contacted. The order implies a preference relation of the agent to receive messages over that address.	Optional	Sequence of url	
resolvers	A sequence of ordered AIDs where name resolution services for the agent can be contacted. The order in the sequence implies a preference in the list of resolvers.	Optional	Sequence of agent-identifier	

724 **7.1.2 Directory Facilitator Agent Description**

725 This type of object represents the description that can be registered with the DF service. This is a reification of the
 726 Agent Directory Entry from [FIPA00001].
 727

Frame Ontology	df-agent-description fipa-agent-management	Parameter	Description	Presence	Type	Reserved Values
name	The identifier of the agent.	Optional	agent-identifier ¹²			
services	A list of services supported by this agent.	Optional	Set of service-description			
protocols	A list of interaction protocols supported by the agent.	Optional	Set of string		See [FIPA00025]	
ontologies	A list of ontologies known by the agent.	Optional	Set of string		fipa-agent-management	
languages	A list of content languages known by the agent.	Optional	Set of string		fipa-sl fipa-sl0 fipa-sl1 fipa-sl2	
lease-time	The duration or time at which the lease for this registration will expire ¹³ .	Optional	datetime ¹⁴			
scope	This parameter defines the visibility of this df-agent-description. The default value is global, meaning that the agent does not wish to put any restriction to the visibility of the registered df-agent-description. The value 'local' means that the registered df-agent-description must not be visible and returned as a result of a search propagated by a federated DF or a DM. All the other values restrict the visibility of the description of the specified DM. E.g. the reserved value df-federation means to advertise the description to all federated DFs (i.e. all cases except fipa-dm-jxta and fipa-dm-bluetooth).	Optional	Set of string		global local fipa-dm-jxta ¹⁵ fipa-dm-bluetooth df-federation	

728 Note: The scope slot is an extension to the first version [SC000023] of this standard. Existing DFs, that are compatible
 729 with the previous version of the specs, are also compatible with this version of the spec with the only exception that
 730 they will not be able to enforce a scope different from 'global'.
 731

¹² A valid df-agent-description must contain at least one AID to comply with the minimum constraints of an Agent Directory Entry from [FIPA00001], except when searching, when no AID need be present.

¹³ The default value for a lease time is assumed to be unlimited.

¹⁴ It is recommended that the value of the lease-time parameter is specified as time duration rather than in absolute time, unless it can be guaranteed that the clocks between the sender and the DF are synchronised.

¹⁵ As an example see [FIPA00095] defining the DM for JXTA.

¹⁶ As default value for dms all available DMs within the DF are assumed, i.e. if the parameter is missing all DMs are referred to.

¹⁷ As an example see [FIPA00095] defining the DM for JXTA.

732

7.1.3 Service Description

733

This type of object represents the description of each service registered with the DF.

734

Frame Ontology	service-description fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
name	The name of the service.	Optional	string	
type	The type of the service.	Optional	string	fipa-df ¹⁸ fipa-ams fipa-dm-jxta fipa-dm-bluetooth
protocols	A list of interaction protocols supported by the service.	Optional	Set of string	
ontologies	A list of ontologies supported by the service.	Optional	Set of string	fipa-agent-management
languages	A list of content languages supported by the service.	Optional	Set of string	
ownership	The owner of the service	Optional	string	
properties	A list of properties that discriminate the service.	Optional	Set of property	

735

737

738

7.1.4 Search Constraints

739

This type of object represents a set of constraints to limit the function of searching within a directory.

740

Frame Ontology	search-constraints fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
max-depth	The maximum depth of propagation of the search either to federated directories, or to DM-nodes ²⁰ . A negative value indicates that the sender agent is willing to allow the search to propagate across all DFs, or as many DM-nodes as possible.	Optional	integer	
max-results	The maximum number ²¹ of results to return for the search ²² . A negative value indicates that the sender agent is willing to receive all available results.	Optional	integer	
search-id	A globally unique identifier for a search.	Optional	string	
dms	A list of DMs ²⁵ that must be used to perform the DF function search (described in section 7.2).	Optional	Set of string	fipa-dm-jxta ²⁶ Fipa-dm-bluetooth

¹⁸ These reserved values denote agents that provide the DF or AMS services as defined Section 5.

¹⁹ See [FIPA00095].

²⁰ The default value for max-depth is 0. In the case of DMs a max-depth of 0 corresponds to one hop, 1 for two hops, etc..

²¹ It is up to the DF to return the most appropriate max-results search results. According to a herein not specified policy, the DF may choose to select all results from only one DM, from several DMs or from all ones in arbitrary quantity (up to max-results) and distribution.

²² The default value for max-results is 1.

²³ The default value for timeout is assumed to be unlimited, i.e. if the parameter is missing, an unlimited timeout is assumed.

741

7.1.5 Agent Management System Agent Description

This type of object represents the description of each service registered with the AMS. This is a reification of the Agent Directory Entry from [FIPA00001].

744

Frame Ontology	ams-agent-description fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
name	The identifier of the agent.	Optional	agent-identifier ²⁷	
ownership	The owner of the agent.	Optional	string	
state	The life cycle state of the agent.	Optional	string	initiated active suspended waiting transit

746

7.1.6 Agent Platform Description

Frame Ontology	ap-description fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
Name	The name of the AP.	Mandatory	string	
Ap-services	The set of services provided by this AP to the resident agents.	Optional	Set of ap-service	

748

7.1.7 Agent Service Description

Frame Ontology	ap-service fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
Name	The name of the AP Service.	Mandatory	string	
Type	The type of the AP Service.	Mandatory	string	fipa.mtp.*
addresses	A list of the addresses of the service.	Mandatory	Sequence of url	

750

7.1.8 Property Template

This is a special object that is useful for specifying parameter/value pairs.

753

Frame Ontology	property fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
name	The name of the property.	Mandatory	string	
value	The value of the property	Mandatory	term	

754

²⁴ The timeout parameter is especially useful when the DF wraps DMs based on ad hoc technology where responses within a reasonable duration often can not be guaranteed. The same may apply for DF federations.

²⁵ As default value for dms all available DMs within the DF are assumed, i.e. if the parameter is missing all DMs are referred to.

²⁶ See [FIPA00095].

²⁷ A valid ams-agent-description must contain at least one AID to comply with the minimum constraints of an Agent Directory Entry from [FIPA00001], except when searching, when no AID need be present.

755 7.2 Function Descriptions

756 The following tables define usage and semantics of the functions that are part of the `fipa-agent-management`
 757 ontology and that are supported by the agent management services and agents on the AP.

758
 759 This ontology does not specify any specific positional order to encode the parameters of the objects. Therefore, it is
 760 required to encode objects in SL by specifying both the parameter name and the parameter value (see Section 3.6 of
 761 [FIPA00008]).

762
 763 The following terms are used to describe the functions of the `fipa-agent-management` domain:
 764

- 765 • **Function**. This is the symbol that identifies the function in the ontology.
- 766
- 767 • **Ontology**. This is the name of the ontology, whose domain of discourse includes the function described in the table.
- 768
- 769 • **Supported by**. This is the type of agent that supports this function.
- 770
- 771 • **Description**. This is a natural language description of the semantics of the function.
- 772
- 773 • **Domain**. This indicates the domain over which the function is defined. The arguments passed to the function must belong to the set identified by the domain.
- 774
- 775 • **Range**. This indicates the range to which the function maps the symbols of the domain. The result of the function is a symbol belonging to the set identified by the range.
- 776
- 777 • **Arity**. This indicates the number of arguments that a function takes. If a function can take an arbitrary number of arguments, then its arity is undefined.

783 7.2.1 Registration of an Object with an Agent

Function	<code>register</code>
Ontology	<code>fipa-agent-management</code>
Supported by	DF and AMS
Description	The execution of this function has the effect of registering a new object into the knowledge base of the executing agent. The DF or AMS description supplied must include a valid AID.
Domain	<code>df-agent-description / ams-agent-description</code>
Range	The execution of this function results in a change of the state, but it has no explicit result. Therefore there is no range set.
Arity	1

784 785 7.2.2 Deregistration of an Object with an Agent

Function	<code>deregister</code>
Ontology	<code>fipa-agent-management</code>
Supported by	DF and AMS
Description	An agent may deregister an object in order to remove all of its parameters from a directory. The DF or AMS description supplied must include a valid AID.
Domain	<code>df-agent-description / ams-agent-description</code>
Range	The execution of this function results in a change of the state, but it has no explicit result. Therefore there is no range set.
Arity	1

787 7.2.3 Modification of an Object Registration with an Agent

Function	modify
Ontology	fipa-agent-management
Supported by	DF and AMS
Description	An agent may make a modification in order to change its object registration with another agent. The argument of a <code>modify</code> function will replace the existing object description stored within the executing agent. The DF or AMS description supplied must include a valid AID.
Domain	<code>df-agent-description / ams-agent-description</code>
Range	The execution of this function results in a change of the state, but it has no explicit result. Therefore there is no range set.
Arity	1

788

789 7.2.4 Search for an Object Registration with an Agent

Function	search
Ontology	fipa-agent-management
Supported by	DF and AMS
Description	An agent may search for an object template in order to request information from an agent, in particular from a DF or an AMS. A successful search can return one or more agent descriptions that satisfy the search criteria and a null set is returned where no agent entries satisfy the search criteria. The DF or AMS description supplied must include a valid AID.
Domain	<code>df-agent-description / ams-agent-description × search-constraints</code>
Range	Set of objects. In particular, a set of <code>df-agent-descriptions</code> (for the DF) and a set of <code>ams-agent-descriptions</code> (for the AMS).
Arity	2

790

791 7.2.4.1 Matching Criterion

792 The `search` action defined in this ontology mandates the implementation of the following matching criterion in order to
 793 determine the set of objects that satisfy the search criteria.

794

795 The first thing to note about the matching operation is that the `search` action receives, as its first argument, an object
 796 description that evaluates to a structured object that will be used as an object template during the execution of the
 797 search action. In the following explanation, the expressions *parameter template* and *value template* are used to
 798 denote a parameter of the object template, and the value of the parameter of the object template, respectively.

799

800 A registered object matches an object template if:

801

- 802 1. The class name of the object (that is, the object type) is the same as the class name of the object description
 803 template, and,
- 804 2. Each parameter of the object template is matched by a parameter of the object description.

805

806 A parameter matches a parameter template if the parameter name is the same as the template parameter name, and
 807 its value matches the value template.

808

809 Since the value of a parameter is a term, the rules for a term to match another term template must be given. Before, it
 810 must be acknowledged that the values of the parameters of descriptions kept by the AMS or by the DF can only be
 811 either a constant, set, sequence (see [FIPA00008]) or other object descriptions (for example, a service-
 812 description).

813

814 The `search` action evaluates functional expressions before the object template is matched against the descriptions
 815 kept by the AMS or by the DF. This means that if the value of a parameter of an object description is a functional term

²⁸ Where × is Cartesian product.

817 (for example, `(plus 2 3)`), then what is seen by the matching process is the result of evaluating the functional term
 818 within the context of the receiving agent. A constant matches a constant template if they are equal.
 819

820 Informally, a sequence matches a sequence template if the elements of the sequence template are matched by
 821 elements of the sequence appearing in the same order. Formally, the following recursive rules apply:
 822

- 823 1. An empty sequence matches an empty sequence, and,
 824
 825 2. The sequence `(cons x sequence1)`²⁹ matches the sequence template `(cons y sequence2)` if:
 826
 827 • `x` matches `y` and `sequence1` matches `sequence2`, or,
 828
 829 • `sequence1` matches `(cons y sequence2)`.
 830

831 Finally, a set matches a set template if each element of the set template is matched by an element of the set template.
 832 Notice that it is possible that the same element of the set matches more than one element of the set template.
 833

834 7.2.4.2 Matching Example

835 The following DF agent description:

```
836
837 (df-agent-description
838   :name
839   (agent-identifier
840     :name cameraproxy1@foo.com
841     :addresses (sequence iiop://foo.com/acc))
842   :services (set
843     (service-description
844       :name description-delivery-1
845       :type description-delivery
846       :ontologies (set traffic-surveillance-domain)
847       :properties (set
848         (property
849           :name camera-id
850           :value cameral)
851         (property
852           :name baud-rate
853           :value 1)))
854     (service-description
855       :name agent-feedback-information-1
856       :type agent-feedback-information
857       :ontologies (set traffic-surveillance-domain)
858       :properties (set
859         (property
860           :name camera-id
861           :value cameral))))
862   :protocols (set fipa-request fipa-query)
863   :ontologies (set traffic-surveillance-domain fipa-agent-management)
864   :languages (set fipa-sl))
```

865 will match the following DF agent description template:
 866

867

²⁹ `cons` is the usual LISP function that it is here used to describe the semantics of the process. The function (which must not be considered part of the `fipa-agent-management` ontology) takes two arguments, the second of which must be a list. It returns a list where the first argument has been inserted as the first element of its second argument. Example: `(cons x (sequence y z))` evaluates to `(sequence x y z)`.

```

868 (df-agent-description
869   :services (set
870     (service-description
871       :type description-delivery
872       :ontologies (set traffic-surveillance-domain)
873       :properties (set
874         (property
875           :name camera-id
876           :value camera1))
877       :languages (set fipa-sl fipa-sl1)))
878

```

Notice that several parameters of the df-agent-description were omitted in the df-agent-description template. Furthermore, not all elements of set-valued parameters of the df-agent-description were specified and, when the elements of a set were themselves descriptions, the corresponding object description templates are also partial descriptions.

883

884
885

886 7.2.5 Retrieve an Agent Platform Description

Function	get-description
Ontology	fipa-agent-management
Supported by	AMS
Description	An agent can make a query in order to request the platform profile of an AP from an AMS.
Domain	None
Range	ap-description
Arity	0

887

888 7.3 Exceptions

889 The normal pattern of interactions between application agents and management agents follow the form of the fipa-request interaction protocol (see [FIPA00026]). Under some circumstances, an exception can be generated, for 890 example, when an AID that has been already registered is re-registered. These exceptions are represented as 891 propositions that evaluate to true under the exceptional circumstances. This section describes the standard set of 892 predicates (defined over a set of arguments) and propositional symbols in the domain of discourse of the fipa- 893 agent-management ontology.

894

895

896 7.3.1 Exception Selection

897 The following rules are adopted to select the appropriate communicative act that will be returned in when a 898 management action causes an exception:

899

- 900 • If the communicative act is not understood by the receiving agent, then the replied communicative act is not-understood.
- 901
- 902 • If the requested action is not supported by the receiving agent, then the communicative act is refuse.
- 903
- 904 • If the requested action is supported by the receiving agent but the sending agent is not authorised to request the function, then the communicative act is refuse.
- 905
- 906 • If the requested function is supported by the receiving agent and the client agent is authorised to request the function but the function is syntactically or semantically ill-specified, then the communicative act is refuse.
- 907
- 908 • In all the other cases the receiving agent sends to the sending agent a communicative act of type agree. Subsequently if any condition arises that prevents the receiving agent from successfully completing the requested function, then the communicative act is failure.
- 909
- 910
- 911
- 912
- 913

914

915 7.3.2 Exception Classes

916 There are four main classes or exceptions that can be generated in response to a management action request:

- 918 • **unsupported**: The communicative act and the content has been understood by the receiving agent, but it is not supported.
- 919 • **unrecognised**: The content has not been understood by the receiving agent.
- 920 • **unexpected**: The content has been understood by the receiving agent, but it includes something that was unexpected.
- 921 • **missing**: The content has been understood by the receiving agent, but something that was expected is missing.

928 7.3.3 Not Understood Exception Predicates

Communicative Act Ontology	not-understood fipa-agent-management	
Predicate Symbol	Arguments	Description
unsupported-act	string	The receiving agent does not support the specific communicative act; the string identifies the unsupported communicative act.
unexpected-act	string	The receiving agent supports the specified communicative act, but it is out of context; the string identifies the unexpected communicative act.
unsupported-value	string	The receiving agent does not support the value of a message parameter; the string identifies the message parameter name.
unrecognised-value	string	The receiving agent cannot recognise the value of a message parameter; the string identifies the message parameter name.

929

930 7.3.4 Refusal Exception Propositions

Communicative Act Ontology	refuse fipa-agent-management	
Predicate symbol	Arguments	Description
Unauthorised		The sending agent is not authorised to perform the function.
unsupported-function	string	The receiving agent does not support the function; the string identifies the unsupported function name.
missing-argument	string	A mandatory function argument is missing; the string identifies the missing function argument name.
unexpected-argument	string	A mandatory function argument is present which is not required; the string identifies the function argument that is unexpected.
unexpected-argument-count		The number of function arguments is incorrect.

missing-parameter	string string	A mandatory parameter is missing; the first string represents the object name and the second string represents the missing parameter name.
unexpected-parameter	string string	The receiving agent does not support the parameter; the first string represents the function name and the second string represents the unsupported parameter name.
unrecognised-parameter-value	string string	The receiving agent cannot recognise the value of a parameter; the first string represents the object name and the second string represents the parameter name of the unrecognised parameter value.

931

932

7.3.5 Failure Exception Propositions

Communicative Act Ontology	failure fipa-agent-management	
Predicate symbol	Arguments	Description
already-registered		The sending agent is already registered with the receiving agent.
not-registered		The sending agent is not registered with the receiving agent.
internal-error	string	An internal error occurred; the string identifies the internal error.

933

934

934 **8 Agent Management Content Language**

935 Agent Management uses fipa-s10 as a content language which is defined in [FIPA00008].

936

937

937 9 References

- 938 [FIPA00001] FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents, 2000.
939 <http://www.fipa.org/specs/fipa0001/>
- 940 [FIPA00008] FIPA SL Content Language Specification. Foundation for Intelligent Physical Agents, 2000.
941 <http://www.fipa.org/specs/fipa0008/>
- 942 [FIPA00025] FIPA Interaction Protocol Library Specification. Foundation for Intelligent Physical Agents, 2000.
943 <http://www.fipa.org/specs/fipa0025/>
- 944 [FIPA00026] FIPA Request Interaction Protocol Specification. Foundation for Intelligent Physical Agents, 2000.
945 <http://www.fipa.org/specs/fipa0026/>
- 946 [[FIPA00035] FIPA Subscribe Interaction Protocol Specification. Foundation for Intelligent Physical Agents, 2000.
947 <http://www.fipa.org/specs/fipa00035/>
- 948 [FIPA00067] FIPA Agent Message Transport Service Specification. Foundation for Intelligent Physical Agents,
949 2000. <http://www.fipa.org/specs/fipa00067/>
- 950 [FIPA00079] FIPA Agent Software Integration Specification. Foundation for Intelligent Physical Agents, 2000.
951 <http://www.fipa.org/specs/fipa00079/>
- 952 [FIPA00095] FIPA JXTA Discovery Middleware Specification. Foundation for Intelligent Physical Agents, 2003.
953 <http://www.fipa.org/specs/fipa00095/>
- 954 [RFC2396] Uniform Resource Identifiers: Generic Syntax. Request for Comments, 1992.
955 <http://www.ietf.org/rfc/rfc2396.txt>
- 956 [JXTA] Project JXTA.
957 <http://www.jxta.org/>
- 958 [BT] Bluetooth.
959 <http://www.bluetooth.org/>

961 10 Informative Annex A — Dialogue Examples

962 1. The agent *dummy* is created and it registers with the AMS of its home AP:

```

963 (request
964   :sender
965     (agent-identifier
966       :name dummy@foo.com
967       :addresses (sequence iiop://foo.com/acc))
968   :receiver (set
969     (agent-identifier
970       :name ams@foo.com
971       :addresses (sequence iiop://foo.com/acc)))
972   :language fipa-s10
973   :protocol fipa-request
974   :ontology fipa-agent-management
975   :content
976     "((action
977       (agent-identifier
978         :name ams@foo.com
979         :addresses (sequence iiop://foo.com/acc))
980       (register
981         (ams-agent-description
982           :name
983             (agent-identifier
984               :name dummy@foo.com
985               :addresses (sequence iiop://foo.com/acc))
986             :state active))))"
987

```

988 2. The AMS agrees and then informs *dummy* of the successful execution of the action:

```

989
990 (agree
991   :sender
992     (agent-identifier
993       :name ams@foo.com
994       :addresses (sequence iiop://foo.com/acc))
995   :receiver (set
996     (agent-identifier
997       :name dummy@foo.com
998       :addresses (sequence iiop://foo.com/acc)))
999   :language fipa-s10
1000  :protocol fipa-request
1001  :ontology fipa-agent-management
1002  :content
1003    "((action
1004      (agent-identifier
1005        :name ams@foo.com
1006        :addresses (sequence iiop://foo.com/acc))
1007      (register
1008        (ams-agent-description
1009          :name
1010            (agent-identifier
1011              :name dummy@foo.com
1012              :addresses (sequence iiop://foo.com/acc))
1013              :state active)))
1014            true))"
1015
1016 (inform
1017   :sender
1018     (agent-identifier
1019       :name ams@foo.com
1020       :addresses (sequence iiop://foo.com/acc))
1021   :receiver (set
1022     (agent-identifier
1023       :name dummy@foo.com

```

```

1024      :addresses (sequence iiop://foo.com/acc)))
1025      :language fipa-s10
1026      :protocol fipa-request
1027      :ontology fipa-agent-management
1028      :content
1029      "((done
1030          (action
1031              (agent-identifier
1032                  :name ams@foo.com
1033                  :addresses (sequence iiop://foo.com/acc)))
1034          (register
1035              (ams-agent-description
1036                  :name
1037                      (agent-identifier
1038                          :name dummy@foo.com
1039                          :addresses (sequence iiop://foo.com/acc))
1040                  :state active))))"
1041
1042 3. Next, dummy registers its services with the default DF of the AP:
1043
1044  (request
1045      :sender
1046          (agent-identifier
1047              :name dummy@foo.com
1048              :addresses (sequence iiop://foo.com/acc)))
1049      :receiver (set
1050          (agent-identifier
1051              :name df@foo.com
1052              :addresses (sequence iiop://foo.com/acc)))
1053      :language fipa-s10
1054      :protocol fipa-request
1055      :ontology fipa-agent-management
1056      :content
1057      "((action
1058          (agent-identifier
1059              :name df@foo.com
1060              :addresses (sequence iiop://foo.com/acc)))
1061          (register
1062              (df-agent-description
1063                  :name
1064                      (agent-identifier
1065                          :name dummy@foo.com
1066                          :addresses (sequence iiop://foo.com/acc)))
1067                  :protocols (set fipa-request application-protocol)
1068                  :ontologies (set meeting-scheduler)
1069                  :languages (set fipa-s10 kif)
1070                  :services (set
1071                      (service-description
1072                          :name profiling
1073                          :type user-profiling
1074                          :ontologies (set meeting-scheduler)
1075                          :properties (set
1076                              (property
1077                                  :name learning-algorithm
1078                                  :value bbn)
1079                              (property
1080                                  :name max-nodes
1081                                  :value 10000000))))
1082                  :scope (set fipa-dm-jxta))))"
1083

```

1083 4. The DF agrees and then informs *dummy* of the successful execution of the action:

1084
 1085 (agree
 1086 :sender
 1087 (agent-identifier
 1088 :name df@foo.com
 1089 :addresses (sequence iiop://foo.com/acc))
 1090 :receiver (set
 1091 (agent-identifier
 1092 :name dummy@foo.com
 1093 :addresses (sequence iiop://foo.com/acc)))
 1094 :language fipa-s10
 1095 :protocol fipa-request
 1096 :ontology fipa-agent-management
 1097 :content
 1098 "((action
 1099 (agent-identifier
 1100 :name df@foo.com
 1101 :addresses (sequence iiop://foo.com/acc))
 1102 (register
 1103 (df-agent-description
 1104 :name
 1105 (agent-identifier
 1106 :name dummy@foo.com
 1107 :addresses (sequence iiop://foo.com/acc))
 1108 :protocols (set fipa-request application-protocol)
 1109 :ontologies (set meeting-scheduler)
 1110 :languages (set fipa-s10 kif)
 1111 :services (set
 1112 (service-description
 1113 :name profiling
 1114 :type user-profiling
 1115 :ontologies (set meeting-scheduler)
 1116 :properties (set
 1117 (property
 1118 :name learning-algorithm
 1119 :value bbn)
 1120 (property
 1121 :name max-nodes
 1122 :value 10000000)))
 1123 :scope (set fipa-dm-jxta))))
 1124 true))")
 1125
 1126 (inform
 1127 :sender
 1128 (agent-identifier
 1129 :name df@foo.com
 1130 :addresses (sequence iiop://foo.com/acc))
 1131 :receiver (set
 1132 (agent-identifier
 1133 :name dummy@foo.com
 1134 :addresses (sequence iiop://foo.com/acc)))
 1135 :language fipa-s10
 1136 :protocol fipa-request
 1137 :ontology fipa-agent-management
 1138 :content
 1139 "((done
 1140 (action
 1141 (agent-identifier
 1142 :name df@foo.com
 1143 :addresses (sequence iiop://foo.com/acc))
 1144 (register
 1145 (df-agent-description
 1146 :name
 1147 (agent-identifier
 1148 :name dummy@foo.com
 1149 :addresses (sequence iiop://foo.com/acc)))

```

1150      :protocols (set fipa-request application-protocol)
1151      :ontologies (set meeting-scheduler)
1152      :languages (set fipa-s10 kif)
1153      :services (set
1154          (service-description
1155              :name profiling
1156              :type user-profiling
1157              :ontologies (set meeting-scheduler)
1158              :properties (set
1159                  (property
1160                      :name learning-algorithm
1161                      :value bbn)
1162                  (property
1163                      :name max-nodes
1164                      :value 10000000)))
1165                  :scope (set fipa-dm-jxta))))"))
1166

```

1167 5. Then, *dummy* searches with the DF for a list of meeting scheduler agents:

```

1168
1169 (request
1170   :sender
1171     (agent-identifier
1172       :name dummy@foo.com
1173       :addresses (sequence iiop://foo.com/acc))
1174   :receiver (set
1175     (agent-identifier
1176       :name df@foo.com
1177       :addresses (sequence iiop://foo.com/acc)))
1178   :language fipa-s10
1179   :protocol fipa-request
1180   :ontology fipa-agent-management
1181   :content
1182     "((action
1183       (agent-identifier
1184         :name df@foo.com
1185         :addresses (sequence iiop://foo.com/acc))
1186     (search
1187       (df-agent-description
1188         :ontologies (set meeting-scheduler)
1189         :languages (set fipa-s10 kif)
1190         :services (set
1191             (service-description
1192               :name profiling
1193               :type meeting-scheduler-service)))
1194         (search-constraints
1195           :max-depth 2
1196           :dms (set fipa-dm-jxta))))")
1197
1198 (agree
1199   :sender
1200     (agent-identifier
1201       :name df@foo.com
1202       :addresses (sequence iiop://foo.com/acc))
1203   :receiver (set
1204     (agent-identifier
1205       :name dummy@foo.com
1206       :addresses (sequence iiop://foo.com/acc)))
1207   :language fipa-s10
1208   :protocol fipa-request
1209   :ontology fipa-agent-management
1210   :content
1211     "((action
1212       (agent-identifier
1213         :name df@foo.com
1214         :addresses (sequence iiop://foo.com/acc))
1215         (search
1216           (df-agent-description

```

```

1217      :ontologies (set meeting-scheduler)
1218      :languages (set fipa-s10 kif)
1219      :services (set
1220          (service-description
1221              :name profiling
1222              :type meeting-scheduler-service)))
1223      (search-constraints
1224          :max-depth 2
1225          :dms (set fipa-dm-jxta))))
1226      true)")
1227
1228  (inform
1229    :sender
1230      (agent-identifier
1231          :name df@foo.com
1232          :addresses (sequence iiop://foo.com/acc))
1233    :receiver (set
1234      (agent-identifier
1235          :name dummy@foo.com
1236          :addresses (sequence iiop://foo.com/acc)))
1237      :language fipa-s10
1238      :protocol fipa-request
1239      :ontology fipa-agent-management
1240      :content
1241        "((result
1242          (action
1243            (agent-identifier
1244                :name df@foo.com
1245                :addresses (sequence iiop://foo.com/acc))
1246            (search
1247              (df-agent-description
1248                  :ontologies (set meeting-scheduler)
1249                  :languages (set fipa-s10 kif)
1250                  :services (set
1251                      (service-description
1252                          :name profiling
1253                          :type meeting-scheduler-service)))
1254                  (search-constraints
1255                      :max-depth 2
1256                      :dms (set fipa-dm-jxta))))
1257            (set
1258              (df-agent-description
1259                :name
1260                  (agent-identifier
1261                      :name scheduler-agent@foo.com
1262                      :addresses (sequence iiop://foo.com/acc))
1263                  :ontologies (set meeting-scheduler fipa-agent-management)
1264                  :languages (set fipa-s10 fipa-s11 kif)
1265                  :services (set
1266                      (service-description
1267                          :name profiling
1268                          :type meeting-scheduler-service)
1269                      (service-description
1270                          :name profiling
1271                          :type user-profiling-service))))))"))
1272
1273

```

- 1273 6. Now *dummy* tries to modify the description of *scheduler-agent* with the DF, but the DF refuses because *dummy* is
 1274 not authorised:

```

1275
1276 (request
1277   :sender
1278     (agent-identifier
1279       :name dummy@foo.com
1280       :addresses (sequence iiop://foo.com/acc))
1281   :receiver (set
1282     (agent-identifier
1283       :name df@foo.com
1284       :addresses (sequence iiop://foo.com/acc)))
1285   :language fipa-s10
1286   :protocol fipa-request
1287   :ontology fipa-agent-management
1288   :content
1289     "((action
1290       (agent-identifier
1291         :name df@foo.com
1292         :addresses (sequence (iiop://foo.com/acc))
1293     (modify
1294       (df-agent-description
1295         :name
1296           (agent-identifier
1297             :name scheduler-agent@foo.com
1298             :addresses (sequence iiop://foo.com/acc))
1299             :ontologies (set meeting-scheduler)
1300             :languages (set fipa-s10 kif)
1301             :services (set
1302               (service-description
1303                 :name profiling
1304                 :type meeting-scheduler-service))))))"
1305
1306 (refuse
1307   :sender
1308     (agent-identifier
1309       :name df@foo.com
1310       :addresses (sequence iiop://foo.com/acc))
1311   :receiver (set
1312     (agent-identifier
1313       :name dummy@foo.com
1314       :addresses (sequence iiop://foo.com/acc)))
1315   :language fipa-s10
1316   :protocol fipa-request
1317   :ontology fipa-agent-management
1318   :content
1319     "((action
1320       (agent-identifier
1321         :name df@foo.com
1322         :addresses (sequence iiop://foo.com/acc))
1323     (modify
1324       (df-agent-description
1325         :name
1326           (agent-identifier
1327             :name scheduler-agent@foo.com
1328             :addresses (sequence iiop://foo.com/acc))
1329             :ontologies (set meeting-scheduler)
1330             :languages (set fipa-s10 kif)
1331             :services (set
1332               (service-description
1333                 :name profiling
1334                 :type meeting-scheduler-service))))"
1335   unauthorised"))

```

- 1336 7. Finally, *dummy* tries to deregister its description with the DF, but the message is ill-formed and the DF does not
 1337 understand (because the DF does not understand the propose performativve):
 1338

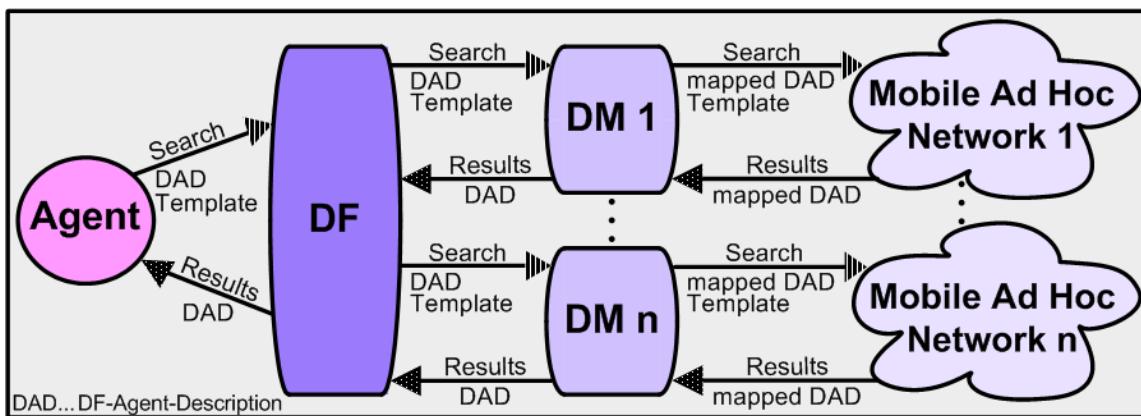
```

1339   (propose
1340     :sender
1341       (agent-identifier
1342         :name dummy@foo.com
1343         :addresses (sequence iiop://foo.com/acc))
1344     :receiver (set
1345       (agent-identifier
1346         :name df@foo.com
1347         :addresses (sequence iiop://foo.com/acc)))
1348     :language fipa-s10
1349     :protocol fipa-request
1350     :ontology fipa-agent-management
1351     :content
1352       "((action
1353         (agent-identifier
1354           :name df@foo.com
1355           :addresses (sequence iiop://foo.com/acc)))
1356       (deregister
1357         (df-agent-description
1358           :name
1359             (agent-identifier
1360               :name dummy@foo.com
1361               :addresses (sequence iiop://foo.com/acc))))))"
1362
1363 (not-understood
1364   :sender
1365     (agent-identifier
1366       :name df@foo.com
1367       :addresses (sequence iiop://foo.com/acc)))
1368   :receiver (set
1369     (agent-identifier
1370       :name dummy@foo.com
1371       :addresses (sequence iiop://foo.com/acc)))
1372     :language fipa-s10
1373     :protocol fipa-request
1374     :ontology fipa-agent-management
1375     :content
1376       "((propose
1377         :sender
1378           (agent-identifier
1379             :name dummy@foo.com
1380             :addresses (sequence iiop://foo.com/acc)))
1381       :receiver (set
1382         (agent-identifier
1383           :name df@foo.com
1384           :addresses (sequence iiop://foo.com/acc)))
1385         :language fipa-s10
1386         :protocol fipa-request
1387         :ontology fipa-agent-management
1388         :content
1389           \"\"((action
1390             (agent-identifier
1391               :name df@foo.com
1392               :addresses (sequence iiop://foo.com/acc))
1393             (deregister
1394               (df-agent-description
1395                 :name
1396                   (agent-identifier
1397                     :name dummy@foo.com
1398                     :addresses (sequence iiop://foo.com/acc))))))\""
1399   (unsupported-act propose)))"
1400

```

1400 11 Informative Annex B — Control Flow Example

1401 To further clarify the reference model for discovery in ad hoc networks described in section 5.1.1, an example will be
 1402 given to explain how the DF and its wrapped DMs work together. Figure 2 outlines the control flow of a DM based
 1403 discovery process of the DF. (Figure 4 abbreviates a df-agent-description as 'DAD'.)
 1404



1405 **Figure 4:** Control Flow of a DM based Discovery within the DF
 1406
 1407
 1408 For searching a certain agent service, first the agent must compose a df-agent-description search template
 1409 describing the service to search.
 1410
 1411 To find the service in one or more ad hoc networks, the agent uses the search functionality of the DF by providing the
 1412 composed df-agent-description search template. Eventually the agent also specifies certain DM(s) (via the dms
 1413 slot in search constraints object), which should be used for this search. This can be done after the agent has
 1414 queried the DF for all currently available DMs. Otherwise, the DF will use all available DMs by default.
 1415
 1416 The DF now uses the search functionality of each available DM, in turn by providing the df-agent-description
 1417 search template.
 1418
 1419 Each invoked DM maps the df-agent-description search template to the appropriate representation of the
 1420 corresponding technology and performs the search in the corresponding ad hoc network.
 1421
 1422 All available df-agent-descriptions, which have previously been registered with the DF, are represented in the
 1423 DM's appropriate form and are discoverable by agents on remote devices.
 1424
 1425 The results of such a search are matching df-agent-descriptions in the appropriate representation of the DM.
 1426 The DM is now mapping the results back to real df-agent-descriptions and returns them to the DF.
 1427
 1428 The DF is collecting the resulting df-agent-descriptions of all invoked DMs and is returning them to the calling
 1429 agent.
 1430
 1431 There are also cases where the DM is not able to provide a template based search. In that cases the DF has to take
 1432 over the missing functionality.
 1433

1433 12 Informative Annex C — Discovery Middleware

1434 Various technologies exist or will appear, which provide discovery in ad hoc networks. Examples of such DM
1435 technologies are JXTA and Bluetooth.

1436
1437 Each DM maintained by the DF must be described in a FIPA specification of its own. In detail, such a DM specification
1438 must describe how the DM functionality, which is imposed by the offered DF functions described in section 7.2, can be
1439 realised on the basis of the respective technology.

1440
1441 This specification does not specify how the actual interface between the DF and its DMs must look like. However,
1442 functionality details of a DM, which are important to consider during DM specification, are summarized next. There are
1443 also cases where the DM is not able to provide that functionality completely. In that cases the DF has to take over the
1444 missing functionality.

- 1445
- **DM ID:** The DF must be able to retrieve a DM's ID string. This enables the DF to differentiate between several
1446 DMs and to provide this information to the agents.
 - **Startup and Shutdown:** The DF must be able to start up and shut down a DM at runtime.
 - **Register and Deregister:** The DF must be able to register `df-agent-descriptions` within a DM and to
1452 deregister `df-agent-descriptions` from the DM.

1453
1454 The lease-time parameter of the `df-agent-description` should be used by a DM to determine the lifetime of
1455 the `df-agent-description`. If no lease-time parameter exists in the `df-agent-description`, the lifetime
1456 should be assumed unlimited. In this case, it is important to deregister the `df-agent-description` later on, in
1457 order to save system resources.

1458
1459 If the `df-agent-description` has already been registered, its lease-time should be renewed according to the
1460 value of the `lease-time` parameter.

- 1461
- **Search:** The DF must be able to search for `df-agent-descriptions` within a DM on the basis of a `df-agent-`
1463 `description` search template. The matching criterion to determine the set of objects that satisfy the search
1464 criteria must exactly be the same as specified for the search function in section 7.2.4.1.

1465
1466 As a means of both limiting the network load as well as the processing load of a device within an ad hoc network,
1467 the max-results parameter of the search-constraints frame (see section 7.1.4) should be used to constrain the
1468 number of returned results per agent platform.

- 1469
- **Subscribe and Unsubscribe:** The DF must be able to subscribe for and unsubscribe from `df-agent-`
1471 `descriptions`, which match a `df-agent-description` search template, within a DM. The matching criterion
1472 to determine the set of objects that satisfy the search criteria must exactly be the same as specified for the
1473 search function in section 7.2.4.1.

1475 13 Informative Annex D — ChangeLog

1476 13.1 2001/10/03 - version H by FIPA Architecture Board

1477 Page 24, line 825: Changed incorrect reference from AMS to DF

1478

1479 13.2 2002/11/01 - version I by TC X2S

1480	Entire document:	Removed all leading : from parameter names
1481	Entire document:	Changed all ontology terms to lowercase
1482	Entire document:	Various typo changes to all examples
1483	Entire document:	Changed references of <i>hap</i> to <i>hap_name</i>
1484	Entire document:	Fixed syntax of the examples by adding extra parenthesis in the content
1485	Page 2, line 105:	Added a footnote linking agent management services to the Abstract Architecture notion of service
1486	Page 2, lines 108-116:	Added a new definition for agent which is compatible with [FIPA00001]
1487	Page 2, line 118:	Removed the requirement that the DF is a mandatory component of the AP
1488	Page 2, line 120:	Added a link between the DF and the Agent Directory Service from [FIPA00001]
1489	Page 3, line 125:	Added a link between the AMS and the Agent Directory Service from [FIPA00001]
1490	Page 3, line 143:	Removed obsolete reference to dynamic registration
1491	Page 4, line 151:	Restructured section on Agent Naming to list all components of an AID and cross-reference with equivalents in [FIPA00001]
1492	Page 4, line 153:	Added a sentence describing AID equivalence
1493	Page 6, line 215:	Removed the requirement that the DF is a mandatory component of the AP
1494	Page 6, line 260:	Changed incorrect reference to <i>df-search-result</i> to <i>max-results</i>
1495	Page 6, line 261:	Added text on limiting the propagation of federated searches
1496	Page 7, lines 265-266:	Removed obsolete reference to dynamic registration
1497	Page 7, lines 278-280:	Removed sentences describing the requirements that the AMS must check all MTS message sends and receives
1498	Page 7, line 297:	Added a link between the <i>name</i> parameter of the AMS and the Service Root from [FIPA00001]
1499	Page 8, line 331:	Removed section on Mandatory Functions Supported by Agents (specifically quit)
1500	Page 9, line 345:	Added an explanatory sentence to the agent life cycle description
1501	Page 10, lines 414, 427:	Removed incorrect reference to [FIPA00005]
1502	Page 11, lines 429-431:	Removed obsolete reference to dynamic registration
1503	Page 11, lines 433-435:	Removed obsolete references to dynamic registration
1504	Page 11, line 469:	Added a section explaining registration lease times
1505	Page 12, line 472:	Added a note that references [FIPA00067] for the closure of <i>fipa-agent-management</i> ontology
1506	Page 13, lines 498, 502:	Modified the names of the following parameters: protocols, ontologies, languages
1507	Page 13, line 493:	Added a link between the <i>addresses</i> parameter and the Locator from [FIPA00001]
1508	Page 13, line 497:	Added a link between the <i>df-agent-description</i> and the Agent Directory Entry from [FIPA00001]
1509	Page 13, line 498:	Added a footnote requiring at least one AID to be present, except when searching
1510	Page 13, line 498:	Changed the plurality of the protocol, ontology and language parameters
1511	Page 13, line 498:	Added a new parameter, lease-time, to the df-agent-description
1512	Page 13, line 498:	Added a footnote explaining the suggested value of lease-time as a time duration
1513	Page 13, line 498:	Added a footnote explaining the default lease time value
1514	Page 13, line 502:	Changed the plurality of the protocol, ontology and language parameters
1515	Page 14, line 506:	Added a note on negative values for max-depth and max-results
1516	Page 14, line 506:	Added a search-id parameter to search-constraints
1517	Page 14, line 509:	Added a link between the <i>ams-agent-description</i> and the Agent Directory Entry from [FIPA00001]
1518	Page 14, Line 510:	Added a footnote requiring at least one AID to be present, except when searching
1519	Page 14, line 512:	Removed mobility parameter from ap-description

1527	Page 14, line 512:	Removed dynamic parameter from ap-description
1528	Page 14, line 512:	Changed name of transport-profile parameter to ap-service
1529	Page 14, line 512:	Changed the plurality of the address parameter
1530	Page 15, line 521:	Added note on how to encode objects in SL
1531	Page 14, line 548:	Modified search action to handle both ams-agent-description and df-agent-description
1532	Page 17, line 588:	Removed the incorrect word 'template' at the end of the sentence
1533	Page 17, line 609:	Changed 1MHZ to 1 in example
1534	Page 18, line 642:	Removed quit function
1535	Page 18, lines 647-649:	Changed the exception model from predicates which return true to propositions that evaluate to true
1536		
1537		
1538		

1539 13.3 2002/12/03 - version J by FIPA Architecture Board

1540	Entire document:	Promoted to Standard status
1541		

1542 13.4 2004/03/18 - version K by TC Ad hoc

1543	Page 1, line 95:	Added an explanatory sentence for ad hoc environments
1544	Page 2, line 127:	Added an explanatory sentence for ad hoc environments
1545	Page 6, line 226:	Added statement about DF being optional
1546	Page 6, line 250:	Added description about ad hoc reference model
1547	Page 6, line 255:	Added picture about ad hoc reference model
1548	Page 7, line 276:	Added introduction to new DF subscribe and DF introspection mechanism
1549	Page 8, line 307:	Added section about subscribe and unsubscribe mechanisms
1550	Page 17, line 223:	Added scope parameter to df-agent-description
1551	Page 17, line 225:	Added note about the backward compatibility of the parameter scope
1552	Page 18, line 730:	Added reserved value for type parameter of the service-description
1553	Page 18, line 736:	Extended the semantics and description (including footnote 20) of the max-depth parameter of the search-constraints
1554	Page 18, line 736:	Added a footnote to the search-constraints frame in order to explain how the max-results parameter must be used in the context of DMs
1555	Page 18, line 736:	Added the dms parameter to the search-constraints
1556	Page 27, line 942:	Added reference to FIPA Subscribe Interaction Protocol Specification
1557	Page 27, line 948:	Added reference to new FIPA specification 00095
1558	Page 27, line 952:	Added reference to the JXTA project webpages
1559	Page 27, line 954:	Added reference to the Bluetooth webpages
1560	Page 29, line 1078:	Added scope parameter to example
1561	Page 30, line 1119:	Added scope parameter to example
1562	Page 31, line 1161:	Added scope parameter to example
1563	Page 31, line 1192:	Added dms parameter to example
1564	Page 32, line 1221:	Added dms parameter to example
1565	Page 32, line 1252:	Added dms parameter to example
1566	Page 35, line 1396:	Added complete Informative Annex B
1567	Page 36, line 1429:	Added complete Informative Annex C
1568	Entire document:	Fixed typos
1569		
1570		
1571		