

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Agent Management Specification

Document title	FIPA Agent Management Specification		
Document number	XC00023H	Document source	FIPA Agent Management
Document status	Experimental	Date of this status	2001/10/03
Supersedes	FIPA00002, FIPA00017, FIPA00019		
Contact	fab@fipa.org		
Change history			
2000/07/31	Approved for Experimental; removed conflicting description of :name parameter in the <code>agent-identifier</code> object		
2000/08/28	Made all parameters of the DF, AMS and Service descriptions optional to allow them to be used with the <code>search</code> function		
2001/08/10	Line numbering added		

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

Geneva, Switzerland

Notice

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

17 **Foreword**

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

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

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

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

36 **Contents**

37	1	Scope	1
38	2	Agent Management Reference Model	2
39	3	Agent Naming	4
40	3.1	Transport Addresses	4
41	3.2	Name Resolution	4
42	4	Agent Management Services	6
43	4.1	Directory Facilitator	6
44	4.1.1	Overview	6
45	4.1.2	Management Functions Supported by the Directory Facilitator	6
46	4.1.3	Federated Directory Facilitators	6
47	4.2	Agent Management System	7
48	4.2.1	Overview	7
49	4.2.2	Management Functions Supported by the Agent Management System	7
50	4.2.3	Management Functions Supported by Agents	8
51	4.3	Message Transport Service	8
52	5	Agent Platform	9
53	5.1	Agent Life Cycle	9
54	5.2	Agent Registration	10
55	6	Agent Management Ontology	12
56	6.1	Object Descriptions	12
57	6.1.1	Agent Identifier Description	12
58	6.1.2	Directory Facilitator Agent Description	13
59	6.1.3	Service Description	13
60	6.1.4	Search Constraints	14
61	6.1.5	Agent Management System Agent Description	14
62	6.1.6	Agent Platform Description	14
63	6.1.7	Property Template	15
64	6.2	Function Descriptions	15
65	6.2.1	Registration of an Object with an Agent	15
66	6.2.2	Deregistration of an Object with an Agent	16
67	6.2.3	Modification of an Object Registration with an Agent	16
68	6.2.4	Search for an Object Registration with an Agent	16
69	6.2.5	Retrieve an Agent Platform Description	18
70	6.2.6	Terminate an Agent	18
71	6.3	Exceptions	18
72	6.3.1	Exception Selection	19
73	6.3.2	Exception Classes	19
74	6.3.3	Not Understood Exception Predicates	19
75	6.3.4	Refusal Exception Propositions	20
76	6.3.5	Failure Exception Propositions	20
77	7	Agent Management Content Language	21
78	8	References	22
79	9	Informative Annex A — Dialogue Examples	23
80	10	Informative Annex D — ChangeLog	30

81 **1 Scope**

82 This document is part of the FIPA specifications covering agent management for inter-operable agents. This
83 specification incorporates and further enhances the FIPA 98 Agent Management Specification [FIPA00002]. The FIPA
84 Agent Message Transport Specification [FIPA00067] represent a companion specification.

85
86 This document contains specifications for agent management including agent management services, agent
87 management ontology and agent platform message transport. This document is primarily concerned with defining open
88 standard interfaces for accessing agent management services. The internal design and implementation of intelligent
89 agents and agent management infrastructure is not mandated by FIPA and is outside the scope of this specification.

90
91 The document provides a series of examples to illustrate the agent management functions defined.
92

2 Agent Management Reference Model

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

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

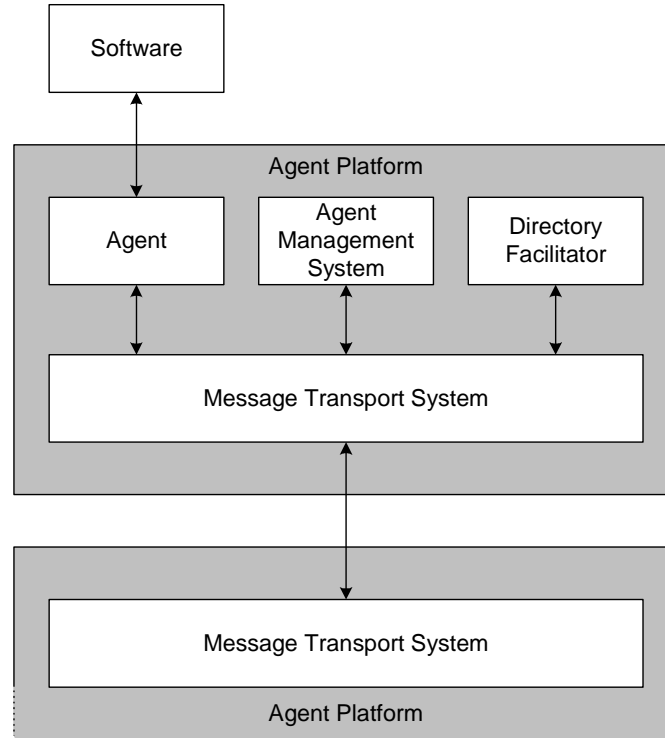


Figure 1: Agent Management Reference Model

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

- An **Agent** is the fundamental actor on an AP which combines one or more service capabilities into a unified and integrated execution model that may include access to external software, human users and communications facilities. An agent may have certain resource brokering capabilities for accessing software (see [FIPA00079]).

An agent must have at least one owner, for example, based on organisational affiliation or human user ownership, and an agent may support several notions of identity. An Agent Identifier (AID) labels an agent so that it may be distinguished unambiguously within the Agent Universe. An agent may be registered at a number of transport addresses at which it can be contacted and it may have certain resource brokering capabilities for accessing software.

- A **Directory Facilitator (DF)** is a mandatory component of the AP. The DF provides yellow pages services to other agents. Agents may register their services with the DF or query the DF to find out what services are offered by other agents. Multiple DFs may exist within an AP and may be federated.
- An **Agent Management System (AMS)** is a mandatory component of the AP. The AMS exerts supervisory control over access to and use of the AP. Only one AMS will exist in a single AP. The AMS maintains a directory of AIDs

124 which contain transport addresses (amongst other things) for agents registered with the AP. The AMS offers white
125 pages services to other agents. Each agent must register with an AMS in order to get a valid AID.
126

- 127 • An **Message Transport Service (MTS)** is the default communication method between agents on different APs
128 (see [FIPA00067]).
129
- 130 • An **Agent Platform (AP)** provides the physical infrastructure in which agents can be deployed. The AP consists of
131 the machine(s), operating system, agent support software, FIPA agent management components (DF, AMS and
132 MTS) and agents.
133

134 The internal design of an AP is an issue for agent system developers and is not a subject of standardisation within
135 FIPA. AP's and the agents which are native to those APs, either by creation directly within or migration to the AP,
136 may use any proprietary method of inter-communication.
137

138 It should be noted that the concept of an AP does not mean that all agents resident on an AP have to be co-located
139 on the same host computer. FIPA envisages a variety of different APs from single processes containing lightweight
140 agent threads, to fully distributed APs built around proprietary or open middleware standards.
141

142 FIPA is concerned only with how communication is carried out between agents who are native to the AP and agents
143 outside the AP or agents who dynamically register with an AP. Agents are free to exchange messages directly by
144 any means that they can support.
145

- 146 • **Software** describes all non-agent, executable collections of instructions accessible through an agent. Agents may
147 access software, for example, to add new services, acquire new communications protocols, acquire new security
148 protocols/algorithms, acquire new negotiation protocols, access tools which support migration, etc.
149
150

151 3 Agent Naming

152 The FIPA agent naming reference model identifies an agent through an extensible collection of parameter-value pairs,
153 called an Agent Identifier (AID). An AID comprises¹:

- 154
- 155 • A name.
- 156
- 157 • Other parameters, such as transport addresses, name resolution service addresses, and so on.
- 158

159 The extensible nature of an AID allows it to be augmented to accommodate other requirements, such as social names,
160 nick names, roles, etc. which can then be attached to services within the AP.

161
162 AIDs are primarily intended to be used to identify agents inside the envelope of a message, specifically within the `:to`
163 and `:from` parameters (see [FIPA00067]). The definition of the AID object and its parameters is given in section 6.1.1,
164 *Agent Identifier Description*.

165
166 The parameter values of an AID can be edited or modified by an agent, for example, to update the sequence of name
167 resolution servers or transport addresses in an AID. However, the mandatory parameters can only be changed by the
168 agent to whom the AID belongs.

169
170 The `:name` parameter of an AID is a globally unique identifier that can be used as a unique referring expression of the
171 agent. One of the simplest mechanisms is to construct it from the actual name of the agent and its home agent platform
172 address² (HAP), separated by the '@' character.

173

174 3.1 Transport Addresses

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

178
179 The EBNF syntax of a transport addresses is the same as for a URL given in [RFC2396]. [FIPA00067] describes the
180 semantics of message delivery with regard to transport addresses.

181

182 3.2 Name Resolution

183 Name resolution is a service that is provided by the AMS through the `search` function. The `:resolvers` parameter of
184 the AID contains a sequence of AIDs at which the AID of the agent can ultimately be resolved into a transport address
185 or set of transport address.

186

187 An example name resolution pattern might be:

188

- 189 1. AgentA wishes to send a message to AgentB, whose AID is:

190

```
191 (agent-identifier
192   :name AgentB@bar.com
193   :resolvers (sequence
194     (agent-identifier
195       :name ams@foo.com
196       :addresses (sequence iiop://foo.com/acc))))
```

197

198 and AgentA wishes to know additional transport addresses that have been given for AgentB.

¹ 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.

² The HAP of an agent is the AP on which the agent was created.

199
200
201
202
203
204
205
206
207
208
209
210
211

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

212 4 Agent Management Services

213 4.1 Directory Facilitator

214 4.1.1 Overview

215 A DF is a mandatory component of an AP that provides a yellow pages directory service to agents. It is the trusted,
 216 benign custodian of the agent directory. It is trusted in the sense that it must strive to maintain an accurate, complete
 217 and timely list of agents. It is benign in the sense that it must provide the most current information about agents in its
 218 directory on a non-discriminatory basis to all authorised agents. At least one DF must be resident on each AP (the
 219 default DF). However, an AP may support any number of DFs and DFs may register with each other to form
 220 federations.

221
 222 Every agent that wishes to publicise its services to other agents, should find an appropriate DF and request the
 223 **registration** of its agent description. There is no intended future commitment or obligation on the part of the registering
 224 agent implied in the act of registering. For example, an agent can refuse a request for a service which is advertised
 225 through a DF. Additionally, the DF cannot guarantee the validity or accuracy of the information that has been registered
 226 with it, neither can it control the life cycle of any agent. An object description must be supplied containing values for all
 227 of the mandatory parameters of the description. It may also supply optional and private parameters, containing non-
 228 FIPA standardised information that an agent developer might want included in the directory. The **deregistration**
 229 function has the consequence that there is no longer a commitment on behalf of the DF to broker information relating to
 230 that agent. At any time, and for any reason, the agent may request the DF to **modify** its agent description.

231
 232 An agent may **search** in order to request information from a DF. The DF does not guarantee the validity of the
 233 information provided in response to a search request, since the DF does not place any restrictions on the information
 234 that can be registered with it. However, the DF may restrict access to information in its directory and will verify all
 235 access permissions for agents which attempt to inform it of agent state changes.

236
 237 The default DF on an AP has a reserved AID of:

```
238 (agent-identifier
239   :name df@hap
240   :addresses (sequence hap_transport_address))
241
242
```

243 4.1.2 Management Functions Supported by the Directory Facilitator

244 In order to access the directory of agent descriptions managed by the DF, each DF must be able to perform the
 245 following functions, when defined on the domain of objects of type `df-agent-description` in compliance with the
 246 semantics described in section 6.1.2, *Directory Facilitator Agent Description*:

- 247 • register
- 248
- 249 • deregister
- 250
- 251 • modify
- 252
- 253 • search
- 254
- 255

256 4.1.3 Federated Directory Facilitators

257 The DF encompasses a search mechanism that searches first locally and then extends the search to other DFs, if
 258 allowed. The default search mechanism is assumed to be a depth-first search across DFs. For specific purposes,
 259 optional constraints can be used as described in section 6.1.4, *Search Constraints* such as the number of answers
 260 (`:df-search-results`). The federation of DFs for extending searches can be achieved by DFs registering with each
 261 other with `fipa-df` as the value of the `:type` parameter in the `service-description`.

262 4.2 Agent Management System

263 4.2.1 Overview

264 An AMS is a mandatory component of the AP and only one AMS will exist in a single AP. The AMS is responsible for
 265 managing the operation of an AP, such as the creation of agents, the deletion of agents, deciding whether an agent can
 266 dynamically register with the AP and overseeing the migration of agents to and from the AP (if agent mobility is
 267 supported by the AP). Since different APs have different capabilities, the AMS can be queried to obtain a description of
 268 its AP. A life cycle is associated with each agent on the AP (see section 5.1, *Agent Life Cycle*) which is maintained by
 269 the AMS.

270
 271 The AMS represents the managing authority of an AP and if the AP spans multiple machines, then the AMS represents
 272 the authority across all machines. An AMS can request that an agent performs a specific management function, such as
 273 `quit` (that is, terminate all execution on its AP) and has the authority to forcibly enforce the function if such a request is
 274 ignored.

275
 276 The AMS maintains an index of all the agents that are currently resident on an AP, which includes the AID of agents.
 277 Residency of an agent on the AP implies that the agent has been registered with the AMS. Each agent, in order to
 278 comply with the FIPA reference model, must **register** with the AMS of its HAP. Registration with the AMS, implies
 279 authorisation to access the MTS of the AP in order to send or receive messages. The AMS will check the validity of the
 280 passed agent description and, in particular, the local uniqueness of the agent name in the AID.

281
 282 Agent descriptions can be later **modified** at any time and for any reason. Modification is restricted by authorisation of
 283 the AMS. The life of an agent with an AP terminates with its **deregistration** from the AMS. After deregistration, the AID
 284 of that agent can be removed by the directory and can be made available to other agents who should request it.

285
 286 Agent description can be **searched** with the AMS and access to the directory of `ams-agent-descriptions` is further
 287 controlled by the AMS; no default policy is specified by this specification.

288
 289 The AMS is also the custodian of the AP description that can be retrieved by requesting the action `get-`
 290 `description`.

291
 292 The AMS on an AP has a reserved AID of:

```
293 (agent-identifier
294   :name ams@hap
295   :addresses (sequence hap_transport_address))
296
297
```

298 4.2.2 Management Functions Supported by the Agent Management System

299 An AMS must be able to perform the following functions, in compliance with the semantics described in section 6.1.5,
 300 *Agent Management System Agent Description* (the first four functions are defined within the scope of the AMS, only on
 301 the domain of objects of type `ams-agent-description` and the last on the domain of objects of type `ap-`
 302 `description`):

- 303
- 304 • `register`
- 305
- 306 • `deregister`
- 307
- 308 • `modify`
- 309
- 310 • `search`
- 311
- 312 • `get-description`
- 313

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

- 316
- 317 • Suspend agent,
- 318
- 319 • Terminate agent,
- 320
- 321 • Create agent,
- 322
- 323 • Resume agent execution,
- 324
- 325 • Invoke agent,
- 326
- 327 • Execute agent, and,
- 328
- 329 • Resource management.
- 330

331 **4.2.3 Management Functions Supported by Agents**

332 Mandatory agent functions:

- 333
- 334 • `quit`
- 335

336 This function is described in section 6.2.6, *Terminate an Agent*.

337

338 **4.3 Message Transport Service**

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

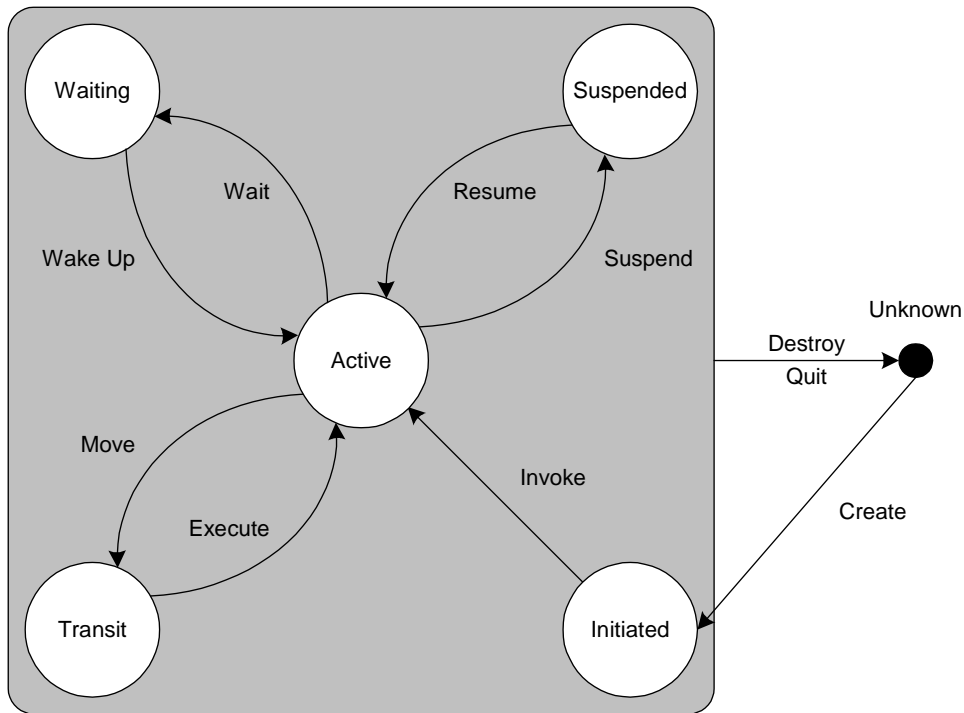
342 5 Agent Platform

343 5.1 Agent Life Cycle

344 FIPA agents exist physically on an AP and utilise the facilities offered by the AP for realising their functionalities. In this
 345 context, an agent, as a physical software process, has a physical life cycle that has to be managed by the AP.
 346

347 The life cycle of a FIPA agent is (see *Figure 2*):
 348

- 349 • **AP Bounded**
 350 An agent is physically managed within an AP and the life cycle of a static agent is therefore always bounded to a
 351 specific AP.
 352
- 353 • **Application Independent**
 354 The life cycle model is independent from any application system and it defines only the states and the transitions of
 355 the agent service in its life cycle.
 356
- 357 • **Instance-Oriented**
 358 The agent described in the life cycle model is assumed to be an instance (that is, an agent which has unique name
 359 and is executed independently).
 360
- 361 • **Unique**
 362 Each agent has only one AP life cycle state at any time and within only one AP.
 363



364
 365
 366 **Figure 2: Agent Life Cycle**
 367

368 The followings are the responsibility that an AMS, on behalf of the AP, has with regard to message delivery in each
 369 state of the life cycle of an agent:
 370

- 371 • **Active**
 372 The MTS delivers messages to the agent as normal.
 373

- 374 • **Initiated/Waiting/Suspended**
 375 The MTS either buffers messages until the agent returns to the active state or forwards messages to a new location
 376 (if a forward is set for the agent).
 377
- 378 • **Transit**
 379 The MTS either buffers messages until the agent becomes active (that is, the move function failed on the original
 380 AP or the agent was successfully started on the destination AP) or forwards messages to a new location (if a
 381 forward is set for the agent). Notice that Only mobile agents can enter the **Transit** state. This ensures that a
 382 stationary agent executes all of its instructions on the node where it was invoked.
 383
- 384 • **Unknown**
 385 The MTS either buffers messages or rejects them, depending upon the policy of the MTS and the transport
 386 requirements of the message.
 387

388 The state transitions of agents can be described as:

- 389
- 390 • **Create**
 391 The creation or installation of a new agent.
 392
- 393 • **Invoke**
 394 The invocation of a new agent.
 395
- 396 • **Destroy**
 397 The forceful termination of an agent. This can only be initiated by the AMS and cannot be ignored by the agent.
 398
- 399 • **Quit**
 400 The graceful termination of an agent. This can be ignored by the agent.
 401
- 402 • **Suspend**
 403 Puts an agent in a suspended state. This can be initiated by the agent or the AMS.
 404
- 405 • **Resume**
 406 Brings the agent from a suspended state. This can only be initiated by the AMS.
 407
- 408 • **Wait**
 409 Puts an agent in a waiting state. This can only be initiated by an agent.
 410
- 411 • **Wake Up**
 412 Brings the agent from a waiting state. This can only be initiated by the AMS.
 413

414 The following two transitions are only used by mobile agents (see [FIPA00005]):

- 415
- 416 • **Move**
 417 Puts the agent in a transitory state. This can only be initiated by the agent.
 418
- 419 • **Execute**
 420 Brings the agent from a transitory state. This can only be initiated by the AMS.
 421

422 5.2 Agent Registration

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

- 424
- 425 • The agent was created on the AP.
 - 426
 - 427 • The agent migrated to the AP, for those APs which support agent mobility (see [FIPA00005]).

428

429

- The agent explicitly registered with the AP, assuming that the AP both supports dynamic registration and is willing to register the new agent. Dynamic registration is where an agent which has a HAP wishes to register on another AP as a local agent.

431

432

433

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

437

438

439

For example:

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

```
(request
  :sender
    (agent-identifier
      :name discovery-agent@bar.com
      :addresses (sequence iiop://bar.com/acc))
  :receiver (set
    (agent-identifier
      :name ams@foo.com
      :addresses (sequence iiop://foo.com/acc)))
  :ontology FIPA-Agent-Management
  :language FIPA-SL0
  :protocol FIPA-Request
  :content
    (action
      (agent-identifier
        :name ams@foo.com
        :addresses (sequence iiop://foo.com/acc))
      (register
        (:ams-description
          :name
            (agent-identifier
              :name discovery-agent@bar.com
              :addresses (sequence iiop://bar.com/acc))
            ...)))
```

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

470 **6 Agent Management Ontology**

471 **6.1 Object Descriptions**

472 This section describes a set of frames, that represent the classes of objects in the domain of discourse within the
 473 framework of the FIPA-Agent-Management ontology.

474
 475 The following terms are used to describe the objects of the domain:

- 477 • **Frame.** This is the mandatory name of this entity, that must be used to represent each instance of this class.
- 478
- 479 • **Ontology.** This is the name of the ontology, whose domain of discourse includes the parameters described in the
 480 table.
- 481
- 482 • **Parameter.** This is the mandatory name of a parameter of this frame.
- 483
- 484 • **Description.** This is a natural language description of the semantics of each parameter.
- 485
- 486 • **Presence.** This indicates whether each parameter is mandatory or optional.
- 487
- 488 • **Type.** This is the type of the values of the parameter: Integer, Word, String, URL, Term, Set or Sequence.
- 489
- 490 • **Reserved Values.** This is a list of FIPA-defined constants that can assume values for this parameter.
- 491

492 **6.1.1 Agent Identifier Description**

493 This type of object represents the identification of the agent.

494

Frame	agent-identifier			
Ontology	FIPA-Agent-Management			
Parameter	Description	Presence	Type	Reserved Values
name	The symbolic name of the agent.	Mandatory	Word	df@hap ams@hap
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	

495

496 **6.1.2 Directory Facilitator Agent Description**

497 This type of object represents the description that can be registered with the DF yellow-page service.

498

Frame Ontology	df-agent-description FIPA-Agent-Management			
Parameter	Description	Presence	Type	Reserved Values
name	The identifier of the agent.	Optional	agent-identifier	
services	A list of services supported by this agent.	Optional	Set of service-description	
protocol	A list of interaction protocols supported by the agent.	Optional	Set of String	See [FIPA00025]
ontology	A list of ontologies known by the agent.	Optional	Set of String	FIPA-Agent-Management
language	A list of content languages known by the agent.	Optional	Set of String	FIPA-SL FIPA-SL0 FIPA-SL1 FIPA-SL2

499

500 **6.1.3 Service Description**

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

502

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 fipa-ams
protocol	A list of interaction protocols supported by the service.	Optional	Set of String	
ontology	A list of ontologies supported by the service.	Optional	Set of String	FIPA-Agent-Management
language	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	

503

504 **6.1.4 Search Constraints**

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

506

Frame Ontology	search-constraints FIPA-Agent-Management			
Parameter	Description	Presence	Type	Reserved Values
max-depth	The maximum depth of propagation of the search to federated directories. This value should not be negative.	Optional	Integer	
max-results	The maximum number of results to return for the search. This value should not be negative.	Optional	Integer	

507

508 **6.1.5 Agent Management System Agent Description**

509 This type of object represents the agent descriptions treated by an AMS agent.

510

Frame Ontology	ams-agent-description FIPA-Agent-Management			
Parameter	Description	Presence	Type	Reserved Values
name	The identifier of the agent.	Optional	agent-identifier	
ownership	The owner of the agent.	Optional	String	
state	The life cycle state of the agent.	Optional	String	initiated active suspended waiting transit

511

512 **6.1.6 Agent Platform Description**

Frame Ontology	ap-description FIPA-Agent-Management			
Parameter	Description	Presence	Type	Reserved Values
name	The name of the AP.	Mandatory	String	
dynamic	The support for dynamic registration of the AP.	Optional	Boolean	
mobility	The support for mobility of the AP.	Optional	Boolean	
transport-profile	The description MTS capabilities of the AP.	Optional	ap-transport-description	See [FIPA00067]

513

514 **6.1.7 Property Template**

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

516

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	

517

518 **6.2 Function Descriptions**

519 The following tables define usage and semantics of the functions that are part of the FIPA-Agent-Management ontology and that are supported by the agent management services and agents on the AP.

520

521 The following terms are used to describe the functions of the FIPA-Agent-Management domain:

522

523

524

- 524 • **Function.** This is the symbol that identifies the function in the ontology.
- 525
- 526 • **Ontology.** This is the name of the ontology, whose domain of discourse includes the function described in the table.
- 527
- 528
- 529 • **Supported by.** This is the type of agent that supports this function.
- 530
- 531 • **Description.** This is a natural language description of the semantics of the function.
- 532
- 533 • **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.
- 534
- 535
- 536 • **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.
- 537
- 538
- 539 • **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.
- 540
- 541

541

542 **6.2.1 Registration of an Object with an Agent**

Function	register
Ontology	FIPA-Agent-Management
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	df-agent-description / ams-agent-description
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

543

544 **6.2.2 Deregistration of an Object with an Agent**

Function	deregister
Ontology	FIPA-Agent-Management
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	df-agent-description / ams-agent-description
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

545

546 **6.2.3 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	df-agent-description / ams-agent-description
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

547

548 **6.2.4 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	object-description-template ³ search-constraints
Range	Set of objects. In particular, a set of df-agent-descriptions (for the DF) and a set of ams-agent-descriptions (for the AMS).
Arity	2

549

550 **6.2.4.1 Matching Criterion**

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

553

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

558

559 A registered object matches an object template if:

560

³ Where \times is Cartesian product.

- 561 1. The class name of the object (that is, the object type) is the same as the class name of the object description
 562 template, and,
 563
 564 2. Each parameter of the object template is matched by a parameter of the object description.
 565

566 A parameter matches a parameter template if the parameter name is the same as the template parameter name, and
 567 its value matches the value template.
 568

569 Since the value of a parameter is a term, the rules for a term to match another term template must be given. Before, it
 570 must be acknowledged that the values of the parameters of descriptions kept by the AMS or by the DF can only be
 571 either `SLConstants`, `SLSets`, `SLSequences` or other object descriptions (for example, a `service-description`).
 572

573 The `search` action evaluates functional expressions before the object template is matched against the descriptions
 574 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
 575 (for example, `(plus 2 3)`), then what is seen by the matching process is the result of evaluating the functional term
 576 within the context of the receiving agent. A constant matches a constant template if they are equal.
 577

578 Informally, a sequence matches a sequence template if the elements of the sequence template are matched by
 579 elements of the sequence appearing in the same order. Formally, the following recursive rules apply:
 580

- 581 1. An empty sequence matches an empty sequence, and,
 582
 583 2. The sequence `(cons x sequence1)`⁴ matches the sequence template `(cons y sequence2)` if:
 584 • `x` matches `y` and `sequence1` matches `sequence2`, or,
 585 • `sequence1` matches `(cons y sequence2)`.
 586

587 Finally, a set matches a set template if each element of the set template is matched by an element of the set template.
 588 Notice that it is possible that the same element of the set matches more than one element of the set template.
 589

590 6.2.4.2 Matching Example

591 The following DF agent description:

```
592 (df-agent-description
593   :name
594     (agent-identifier
595       :name CameraProxyl@foo.com
596       :addresses (sequence iiop://foo.com/acc))
597   :services (set
598     (service-description
599       :name description-delivery-1
600       :type description-delivery
601       :ontology (set Traffic-Surveillance-Domain)
602       :properties (set
603         (property
604           :name camera-id
605           :value camera1)
606         (property
607           :name baud-rate
608           :value 1MHz)))
609     (service-description
610       :name agent-feedback-information-1
611       :type agent-feedback-information
612       :ontology (set traffic-surveillance-domain)
613       :properties (set
614         (property
```

⁴ `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)`.

```

616         :name camera-id
617         :value camera1)))
618 :protocol (set FIPA-Request FIPA-Query)
619 :ontology (set Traffic-Surveillance-Domain FIPA-Agent-Management)
620 :language (set FIPA-SL)
621

```

will match the following DF agent description template:

```

623 (df-agent-description
624   :services (set
625     (service-description
626       :type description-delivery
627       :ontology (set Traffic-Surveillance-Domain)
628       :properties (set
629         (property
630           :name camera-id
631           :value camera1))
632       :language (set FIPA-SL FIPA-SL1))
633   )
634

```

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.

6.2.5 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

641

6.2.6 Terminate an Agent

Function	quit
Ontology	FIPA-Agent-Management
Supported by	All agents
Description	An AMS can ask an agent to terminate all execution on a given AP. Also, an agent can request the AMS to terminate the execution of an agent.
Domain	agent-identifier
Range	The execution of this function results in a change of state in the AMS but it has no explicit range set.
Arity	1

643

6.3 Exceptions

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 example, when an AID that has been already registered is re-registered. These exceptions are represented as predicates that become true. This section describes all the predicates of the domain of discourse of the `FIPA-Agent-Management` ontology that represent exceptions of the interactions.

6.3.1 Exception Selection

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

- If the communicative act is not understood by the receiving agent, then the replied communicative act is `not-understood`.
- If the requested action is not supported by the receiving agent, then the communicative act is `refuse`.
- 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`.
- 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`.
- 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`.

6.3.2 Exception Classes

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

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

6.3.3 Not Understood Exception Predicates

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

683

684 **6.3.4 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 unrequired function argument.
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.

685

686 **6.3.5 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.

687

688 **7 Agent Management Content Language**

689 Agent Management uses `FIPA-SL0` as a content language which is defined in [FIPA00008].

690 **8 References**

- 691 [FIPA00008] FIPA SL Content Language Specification. Foundation for Intelligent Physical Agents, 2000.
692 <http://www.fipa.org/specs/fipa00008/>
- 693 [FIPA00025] FIPA Interaction Protocol Library Specification. Foundation for Intelligent Physical Agents, 2000.
694 <http://www.fipa.org/specs/fipa00025/>
- 695 [FIPA00026] FIPA Request Interaction Protocol Specification. Foundation for Intelligent Physical Agents, 2000.
696 <http://www.fipa.org/specs/fipa00026/>
- 697 [FIPA00067] FIPA Agent Message Transport Service Specification. Foundation for Intelligent Physical Agents, 2000.
698 <http://www.fipa.org/specs/fipa00067/>
- 699 [FIPA00079] FIPA Agent Software Integration Specification. Foundation for Intelligent Physical Agents, 2000.
700 <http://www.fipa.org/specs/fipa00079/>
- 701 [RFC2396] Uniform Resource Identifiers: Generic Syntax. Request for Comments, 1992.
702 <http://www.ietf.org/rfc/rfc2396.txt>

9 Informative Annex A — Dialogue Examples

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

```

706 (request
707   :sender
708     (agent-identifier
709       :name dummy@foo.com
710       :addresses (sequence iiop://foo.com/acc))
711   :receiver (set
712     (agent-identifier
713       :name ams@foo.com
714       :addresses (sequence iiop://foo.com/acc)))
715   :language FIPA-SL0
716   :protocol FIPA-Request
717   :ontology FIPA-Agent-Management
718   :content
719     (action
720       (agent-identifier
721         :name ams@foo.com
722         :addresses (sequence iiop://foo.com/acc))
723       (register
724         (ams-agent-description
725           :name
726             (agent-identifier
727               :name dummy@foo.com
728               :addresses (sequence iiop://foo.com/acc))
729           :state active))))))

```

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

```

733 (agree
734   :sender
735     (agent-identifier
736       :name ams@foo.com
737       :addresses (sequence iiop://foo.com/acc))
738   :receiver (set
739     (agent-identifier
740       :name dummy@foo.com
741       :addresses (sequence iiop://foo.com/acc)))
742   :language FIPA-SL0
743   :protocol FIPA-Request
744   :ontology FIPA-Agent-Management
745   :content
746     ((action
747       (agent-identifier
748         :name ams@foo.com
749         :addresses (sequence iiop://foo.com/acc))
750       (register
751         (ams-agent-description
752           :name
753             (agent-identifier
754               :name dummy@foo.com
755               :addresses (sequence iiop://foo.com/acc))
756           :state active))))
757     true))
758
759 (inform
760   :sender
761     (agent-identifier
762       :name ams@foo.com

```

```

763         :addresses (sequence iiop://foo.com/acc))
764 :receiver (set
765   (agent-identifier
766     :name dummy@foo.com
767     :addresses (sequence iiop://foo.com/acc)))
768 :language FIPA-SL0
769 :protocol FIPA-Request
770 :ontology FIPA-Agent-Management
771 :content
772   (done
773     (action
774       (agent-identifier
775         :name ams@foo.com
776         :addresses (sequence iiop://foo.com/acc))
777       (register
778         (ams-agent-description
779           :name
780             (agent-identifier
781               :name dummy@foo.com
782               :addresses (sequence iiop://foo.com/acc))
783             :state active))))))

```

3. Next, *dummy* registers its services with the default DF of the AP:

```

786 (request
787   :sender
788     (agent-identifier
789       :name dummy@foo.com
790       :addresses (sequence iiop://foo.com/acc))
791   :receiver (set
792     (agent-identifier
793       :name df@foo.com
794       :addresses (sequence iiop://foo.com/acc)))
795   :language FIPA-SL0
796   :protocol FIPA-Request
797   :ontology FIPA-Agent-Management
798   :content
799     (action
800       (agent-identifier
801         :name df@foo.com
802         :addresses (sequence iiop://foo.com/acc))
803       (register
804         (df-agent-description
805           :name
806             (agent-identifier
807               :name dummy@foo.com
808               :addresses (sequence iiop://foo.com/acc))
809             :protocol (set FIPA-Request Application-Protocol)
810             :ontology (set meeting-scheduler)
811             :language (set FIPA-SL0 KIF)
812             :services (set
813               (service-description
814                 :name profiling
815                 :type user-profiling
816                 :ontology (set meeting-scheduler)
817                 :properties (set
818                   (property
819                     :name learning-algorithm
820                     :value BBN)
821                   (property
822                     :name max-nodes
823                     :value 1000000))))))))))
824

```

825 4. The AMS agrees and then informs *dummy* of the successful execution of the action:

```

826
827 (agree
828   :sender
829     (agent-identifier
830      :name df@foo.com
831      :addresses (sequence iiop://foo.com/acc))
832   :receiver (set
833     (agent-identifier
834      :name dummy@foo.com
835      :addresses (sequence iiop://foo.com/acc)))
836   :language FIPA-SL0
837   :protocol FIPA-Request
838   :ontology FIPA-Agent-Management
839   :content
840     ((action
841      (agent-identifier
842       :name df@foo.com
843       :addresses (sequence iiop://foo.com/acc))
844      (register
845       (df-agent-description
846        :name
847          (agent-identifier
848           :name dummy@foo.com
849           :addresses (sequence iiop://foo.com/acc))
850          :protocol (set FIPA-Request Application-Protocol)
851          :ontology (set meeting-scheduler)
852          :language (set FIPA-SL0 KIF)
853          :services (set
854            (service-description
855             :name profiling
856             :type user-profiling
857             :ontology (set meeting-scheduler)
858             :properties (set
859               (property
860                :name learning-algorithm
861                :value BBN)
862               (property
863                :name max-nodes
864                :value 1000000))))))))))
865     true))
866
867 (inform
868   :sender
869     (agent-identifier
870      :name df@foo.com
871      :addresses (sequence iiop://foo.com/acc))
872   :receiver (set
873     (agent-identifier
874      :name dummy@foo.com
875      :addresses (sequence iiop://foo.com/acc)))
876   :language FIPA-SL0
877   :protocol FIPA-Request
878   :ontology FIPA-Agent-Management
879   :content
880     (done
881      (action
882       (agent-identifier
883        :name df@foo.com
884        :addresses (sequence iiop://foo.com/acc))
885       (register
886        (df-agent-description
887         :name
888          (agent-identifier

```

```

889         :name dummy@foo.com
890         :addresses (sequence iiop://foo.com/acc))
891 :protocol (set FIPA-Request Application-Protocol)
892 :ontology (set meeting-scheduler)
893 :language (set FIPA-SL0 KIF)
894 :services (set
895   (service-description
896     :name profiling
897     :type user-profiling
898     :ontology (set meeting-scheduler)
899     :properties (set
900       (property
901         :name learning-algorithm
902         :value BBN)
903       (property
904         :name max-nodes
905         :value 10000000)))))))))
906

```

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

```

908 (request
909   :sender
910     (agent-identifier
911       :name dummy@foo.com
912       :addresses (sequence iiop://foo.com/acc))
913   :receiver (set
914     (agent-identifier
915       :name df@foo.com
916       :addresses (sequence iiop://foo.com/acc)))
917   :language FIPA-SL0
918   :protocol FIPA-Request
919   :ontology FIPA-Agent-Management
920   :content
921     (action
922       (agent-identifier
923         :name df@foo.com
924         :addresses (sequence iiop://foo.com/acc))
925       (search
926         (df-agent-description
927           :ontology (set meeting-scheduler)
928           :language (set FIPA-SL0 KIF)
929           :services (set
930             (service-description
931               :name profiling
932               :type meeting-scheduler-service)))
933         (search-constraints
934           :min-depth 2))))))
936 (agree
937   :sender
938     (agent-identifier
939       :name df@foo.com
940       :addresses (sequence iiop://foo.com/acc))
941   :receiver (set
942     (agent-identifier
943       :name dummy@foo.com
944       :addresses (sequence iiop://foo.com/acc)))
945   :language FIPA-SL0
946   :protocol FIPA-Request
947   :ontology FIPA-Agent-Management
948   :content
949     ((action
950       (agent-identifier

```

```

952         :name df@foo.com
953         :addresses (sequence iiop://foo.com/acc))
954     (search
955         (df-agent-description
956           :ontology (set meeting-scheduler)
957           :language (set FIPA-SL0 KIF)
958           :services (set
959             (service-description
960               :name profiling
961               :type meeting-scheduler-service))
962           (search-constraint :max-depth 2))))
963     true))
964
965 (inform
966   :sender
967     (agent-identifier
968       :name df@foo.com
969       :addresses (sequence iiop://foo.com/acc))
970   :receiver (set
971     (agent-identifier
972       :name dummy@foo.com
973       :addresses (sequence iiop://foo.com/acc)))
974   :language FIPA-SL0
975   :protocol FIPA-Request
976   :ontology FIPA-Agent-Management
977   :content
978     (result
979       (action
980         (agent-identifier
981           :name df@foo.com
982           :addresses (sequence iiop://foo.com/acc))
983       (search
984         (df-agent-description
985           :ontology (set meeting-scheduler)
986           :language (set FIPA-SL0 KIF)
987           :services (set
988             (service-description
989               :name profiling
990               :type meeting-scheduler-service))
991         (search-constraint :max-depth 2))))
992       (set
993         (df-agent-description
994           :name
995             (agent-identifier
996               :name scheduler-agent@foo.com
997               :addresses (sequence iiop://foo.com/acc))
998           :ontology (set meeting-scheduler FIPA-Agent-Management)
999           :languages (set FIPA-SL0 FIPA-SL1 KIF)
1000          :services (set
1001            (service-description
1002              :name profiling
1003              :type meeting-scheduler-service)
1004            (service-description
1005              :name profiling
1006              :type user-profiling-service))))))

```

1007 6. Now *dummy* tries to modify the description of *scheduler-agent* with the DF, but the DF refuses because *dummy* is
 1008 not authorised:

```

1009
1010 (request
1011   :sender
1012     (agent-identifier
1013       :name dummy@foo.com
1014       :addresses (sequence iiop://foo.com/acc))
1015   :receiver (set
1016     (agent-identifier
1017       :name df@foo.com
1018       :addresses (sequence iiop://foo.com/acc)))
1019   :language FIPA-SL0
1020   :protocol FIPA-Request
1021   :ontology FIPA-Agent-Management
1022   :content
1023     (action
1024       (agent-identifier
1025         :name df@foo.com
1026         :addresses (sequence (iiop://foo.com/acc))
1027       (modify
1028         (df-agent-description
1029           :name
1030             (agent-identifier
1031               :name scheduler-agent@foo.com
1032               :addresses (sequence iiop://foo.com/acc))
1033             :ontology (set meeting-scheduler)
1034             :language (set FIPA-SL0 KIF)
1035             :services (set
1036               (service-description
1037                 :name profiling
1038                 :type meeting-scheduler-service))))))
1039
1040 (refuse
1041   :sender
1042     (agent-identifier
1043       :name df@foo.com
1044       :addresses (sequence iiop://foo.com/acc))
1045   :receiver (set
1046     (agent-identifier
1047       :name dummy@foo.com
1048       :addresses (sequence iiop://foo.com/acc)))
1049   :language FIPA-SL0
1050   :protocol FIPA-Request
1051   :ontology FIPA-Agent-Management
1052   :content
1053     ((action
1054       (agent-identifier
1055         :name df@foo.com
1056         :addresses (sequence iiop://foo.com/acc))
1057       (modify
1058         (df-agent-description
1059           :name
1060             (agent-identifier
1061               :name scheduler-agent@foo.com
1062               :addresses (sequence iiop://foo.com/acc))
1063             :ontology (set meeting-scheduler)
1064             :language (set FIPA-SL0 KIF)
1065             :services (set
1066               (service-description
1067                 :name profiling
1068                 :type meeting-scheduler-service))))))
1069     (unauthorised)))

```

1070 7. Finally, *dummy* tries to deregister its description with the DF, but the message is ill-formed and the DF does not
 1071 understand (because the DF does not understand the propose performative):
 1072

```

1073 (propose
1074   :sender
1075     (agent-identifier
1076       :name dummy@foo.com
1077       :addresses (sequence iiop://foo.com/acc))
1078   :receiver (set
1079     (agent-identifier
1080       :name df@foo.com
1081       :addresses (sequence iiop://foo.com/acc)))
1082   :language FIPA-SL0
1083   :protocol FIPA-Request
1084   :ontology FIPA-Agent-Management
1085   :content
1086     (action
1087       (agent-identifier
1088         :name df@foo.com
1089         :addresses (sequence iiop://foo.com/acc))
1090       (deregister
1091         (df-agent-description
1092           :name
1093             (agent-identifier
1094               :name dummy@foo.com
1095               :addresses (sequence iiop://foo.com/acc))))))
1096
1097 (not-understood
1098   :sender
1099     (agent-identifier
1100       :name df@foo.com
1101       :addresses (sequence iiop://foo.com/acc))
1102   :receiver (set
1103     (agent-identifier
1104       :name dummy@foo.com
1105       :addresses (sequence iiop://foo.com/acc)))
1106   :language FIPA-SL0
1107   :protocol FIPA-Request
1108   :ontology FIPA-Agent-Management
1109   :content
1110     (propose
1111       :sender
1112         (agent-identifier
1113           :name dummy@foo.com
1114           :addresses (sequence iiop://foo.com/acc))
1115       :receiver (set
1116         (agent-identifier
1117           :name df@foo.com
1118           :addresses (sequence iiop://foo.com/acc)))
1119       :language FIPA-SL0
1120       :protocol FIPA-Request
1121       :ontology FIPA-Agent-Management
1122       :content
1123         (action
1124           (agent-identifier
1125             :name df@foo.com
1126             :addresses (sequence iiop://foo.com/acc))
1127           (deregister
1128             (df-agent-description
1129               :name
1130                 (agent-identifier
1131                   :name dummy@foo.com
1132                   :addresses (sequence iiop://foo.com/acc))))))
1133       (unsupported-act propose)))

```


1134 **10 Informative Annex D — ChangeLog**

1135 **10.1 2001/10/03 - version H by FIPA Architecture Board**

1136 Page 24, line 825: Changed incorrect reference of AMS to DF.