

1
2
3
4

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

5
6

FIPA Agent Management Specification

7
8
9
10
11
12
13
14
15
16
17

Document title	FIPA Agent Management Specification		
Document number	XC00023I	Document source	FIPA TC Agent Management
Document status	Experimental	Date of this status	2002/11/01
Supersedes	FIPA00002, FIPA00017, FIPA00019		
Contact	fab@fipa.org		
Change history	See <i>Informative Annex B — ChangeLog</i>		

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 and
25 universities that are active in the field of agents. FIPA makes the results of its activities available to all interested parties
26 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 to
30 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 process
34 of specification may be found in the FIPA Document Policy [f-out-00000] and the FIPA Specifications Policy [f-out-
35 00003]. A complete overview of the FIPA specifications and their current status may be found on the FIPA Web site.

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

39 Contents

40	1	Scope.....	1
41	2	Agent Management Reference Model.....	2
42	3	Agent Naming.....	4
43	3.1	Transport Addresses.....	4
44	3.2	Name Resolution.....	4
45	4	Agent Management Services.....	6
46	4.1	Directory Facilitator.....	6
47	4.1.1	Overview.....	6
48	4.1.2	Management Functions Supported by the Directory Facilitator.....	6
49	4.1.3	Federated Directory Facilitators.....	6
50	4.2	Agent Management System.....	7
51	4.2.1	Overview.....	7
52	4.2.2	Management Functions Supported by the Agent Management System.....	7
53	4.3	Message Transport Service.....	8
54	5	Agent Platform.....	9
55	5.1	Agent Life Cycle.....	9
56	5.2	Agent Registration.....	10
57	5.2.1	Registration Lease Times.....	11
58	6	Agent Management Ontology.....	13
59	6.1	Object Descriptions.....	13
60	6.1.1	Agent Identifier Description.....	13
61	6.1.2	Directory Facilitator Agent Description.....	14
62	6.1.3	Service Description.....	14
63	6.1.4	Search Constraints.....	15
64	6.1.5	Agent Management System Agent Description.....	15
65	6.1.6	Agent Platform Description.....	15
66	6.1.7	Agent Service Description.....	16
67	6.1.8	Property Template.....	16
68	6.2	Function Descriptions.....	16
69	6.2.1	Registration of an Object with an Agent.....	17
70	6.2.2	Deregistration of an Object with an Agent.....	17
71	6.2.3	Modification of an Object Registration with an Agent.....	17
72	6.2.4	Search for an Object Registration with an Agent.....	17
73	6.2.5	Retrieve an Agent Platform Description.....	19
74	6.3	Exceptions.....	19
75	6.3.1	Exception Selection.....	20
76	6.3.2	Exception Classes.....	20
77	6.3.3	Not Understood Exception Predicates.....	20
78	6.3.4	Refusal Exception Propositions.....	21
79	6.3.5	Failure Exception Propositions.....	21
80	7	Agent Management Content Language.....	22
81	8	References.....	23
82	9	Informative Annex A — Dialogue Examples.....	24
83	10	Informative Annex B — ChangeLog.....	31
84	10.1	2001/10/03 - version H by FIPA Architecture Board.....	31
85	10.2	2002/11/01 - version I by TC X2S.....	31

86 **1 Scope**

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

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

94
95 The document provides a series of examples to illustrate the agent management functions defined.
96

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.

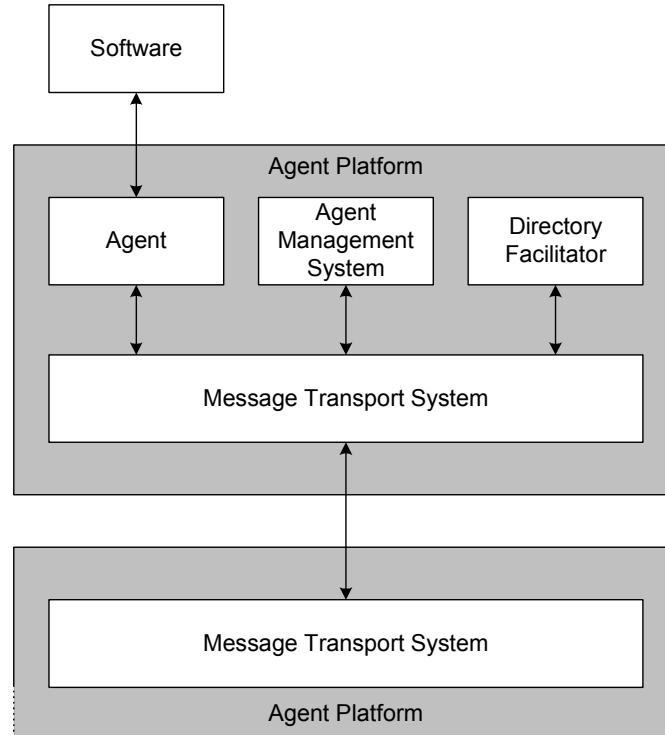


Figure 1: Agent Management Reference Model

The agent management reference model consists of the following logical components¹, each representing a capability set (these can be combined in physical implementations of APs):

- An **agent** is a computational process that implements the autonomous, communicating functionality of an application. Agents communicate using an Agent Communication Language. An Agent is the fundamental actor on an AP which combines one or more service capabilities, as published in a service description, into a unified and integrated execution model. An agent must have at least one owner, for example, based on organisational affiliation or human user ownership, and an agent must support at least one notion of identity. This notion of identity is the Agent Identifier (AID) that 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.
- A **Directory Facilitator (DF)** is an optional component of the AP, but if it is present, it must be implemented as a DF service (see Section 4.1). 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. The DF is a reification of the Agent Directory Service in [FIPA00001].
- 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

¹ The functionalities of these components are a specialization of the AA notion of Service [see FIPA00001].

127 which contain transport addresses (amongst other things) for agents registered with the AP. The AMS offers white
128 pages services to other agents. Each agent must register with an AMS in order to get a valid AID. The AMS is a
129 reification of the Agent Directory Service in [FIPA00001].
130

- 131 • An **Message Transport Service (MTS)** is the default communication method between agents on different APs (see
132 [FIPA00067]).
133
- 134 • An **Agent Platform (AP)** provides the physical infrastructure in which agents can be deployed. The AP consists of
135 the machine(s), operating system, agent support software, FIPA agent management components (DF, AMS and
136 MTS) and agents.
137

138 The internal design of an AP is an issue for agent system developers and is not a subject of standardisation within
139 FIPA. AP's and the agents which are native to those APs, either by creation directly within or migration to the AP,
140 may use any proprietary method of inter-communication.
141

142 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
143 on the same host computer. FIPA envisages a variety of different APs from single processes containing lightweight
144 agent threads, to fully distributed APs built around proprietary or open middleware standards.
145

146 FIPA is concerned only with how communication is carried out between agents who are native to the AP and agents
147 outside the AP. Agents are free to exchange messages directly by any means that they can support.
148

- 149 • **Software** describes all non-agent, executable collections of instructions accessible through an agent. Agents may
150 access software, for example, to add new services, acquire new communications protocols, acquire new security
151 protocols/algorithms, acquire new negotiation protocols, access tools which support migration, etc.
152

153 3 Agent Naming

154 The FIPA agent naming reference model identifies an agent through an extensible collection of parameter-value pairs²,
 155 called an Agent Identifier (AID). The extensible nature of an AID allows it to be augmented to accommodate other
 156 requirements, such as social names, nick names, roles, etc. which can then be attached to services within the AP. An
 157 AID comprises³ (see Section 6.1.1):

- 158
- 159 • The `name` parameter, which is a globally unique identifier that can be used as a unique referring expression of the
 160 agent. One of the simplest mechanisms is to construct it from the actual name of the agent and its home agent
 161 platform address⁴ (HAP), separated by the `@` character. This is a reification of the notion of an Agent Name from
 162 [FIPA00001].
- 163
- 164 • The `addresses` parameter, which is a list of transport addresses where a message can be delivered (see Section
 165 3.1). This is a reification of the notion of a Locator from [FIPA00001].
- 166
- 167 • The `resolvers` parameter, which is a list of name resolution service addresses (see Section 3.2).
- 168

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

173

174 Two AIDs are considered to be equivalent if their `name` parameters are the same.

175

176 3.1 Transport Addresses

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

180

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

183

184 3.2 Name Resolution

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

188

189 An example name resolution pattern might be:

- 190
- 191 1. *agent-a* wishes to send a message to *agent-b*, whose AID is:
- 192

```

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

² 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.

³ 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.

⁴ The HAP of an agent is the AP on which the agent was created.

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

199
200
201
202
203
204
205
206
207
208
209
210
211
212
213

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

2. Therefore, *agent-a* 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 *agent-b* registered with it, then it returns a `result` message containing the AMS agent description of *agent-b*; if not, then a `failed` message is returned.
4. Upon receipt of the `result` message, *agent-a* 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 *agent-b*.
5. *agent-a* can now send a message to *agent-b* by inserting the `addresses` parameter into the AID of *agent-b*.

214 4 Agent Management Services

215 4.1 Directory Facilitator

216 4.1.1 Overview

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

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

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

238
 239 The default DF on an AP has a reserved AID of:

```
240 (agent-identifier
241   :name df@hap_name6
242   :addresses (sequence hap_transport_address))
243
244
```

245 4.1.2 Management Functions Supported by the Directory Facilitator

246 In order to access the directory of agent descriptions managed by the DF, each DF must be able to perform the
 247 following functions, when defined on the domain of objects of type `df-agent-description` in compliance with the
 248 semantics described in Section 6.1.2:

- 249
- 250 • register
- 251
- 252 • deregister
- 253
- 254 • modify
- 255
- 256 • search
- 257

258 4.1.3 Federated Directory Facilitators

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

264

265

266

267

268

269

270

271

272

273

274

When a DF receives a search action, it may determine whether it needs to propagate this search to other DFs that are registered with it⁷. It should only forward searches where the value of the `max-depth` parameter is greater than 1 and where it has not received a prior search with the same `search-id` parameter. If it does forward the search action, then it must use the following rules:

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

275

4.2 Agent Management System

276

4.2.1 Overview

277

278

279

280

281

282

An AMS is a mandatory component of the AP and only one AMS will exist in a single AP. The AMS is responsible for managing the operation of an AP, such as the creation of agents, the deletion of agents and overseeing the migration of agents to and from the AP (if agent mobility is supported by the AP). Since different APs have different capabilities, the AMS can be queried to obtain a description of its AP. A life cycle is associated with each agent on the AP (see Section 5.1) which is maintained by the AMS.

283

284

285

286

287

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

288

289

290

291

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

292

293

294

295

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

296

297

298

299

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

300

301

302

303

304

305

The AMS on an AP has a reserved AID of:

```
(agent-identifier
 :name ams@hap_name8
 :addresses (sequence hap_transport_address))
```

306

307

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

308

4.2.2 Management Functions Supported by the Agent Management System

309

310

311

312

313

An AMS must be able to perform the following functions, in compliance with the semantics described in Section 6.1.5 (the first four functions are defined within the scope of the AMS, only on the domain of objects of type `ams-agent-description` and the last on the domain of objects of type `ap-description`):

- `register`

⁷ Some DFs may not support federated search, in which case the `max-result`, `max-depth` and `search-id` parameters have no effect.

⁸ The `hap_name` should be replaced with the name of the HAP that is published in the `ap-description`.

314

315 • deregister

316

317 • modify

318

319 • search

320

321 • get-description

322

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

325

326 • Suspend agent,

327

328 • Terminate agent,

329

330 • Create agent,

331

332 • Resume agent execution,

333

334 • Invoke agent,

335

336 • Execute agent, and,

337

338 • Resource management.

339

340 **4.3 Message Transport Service**

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

344

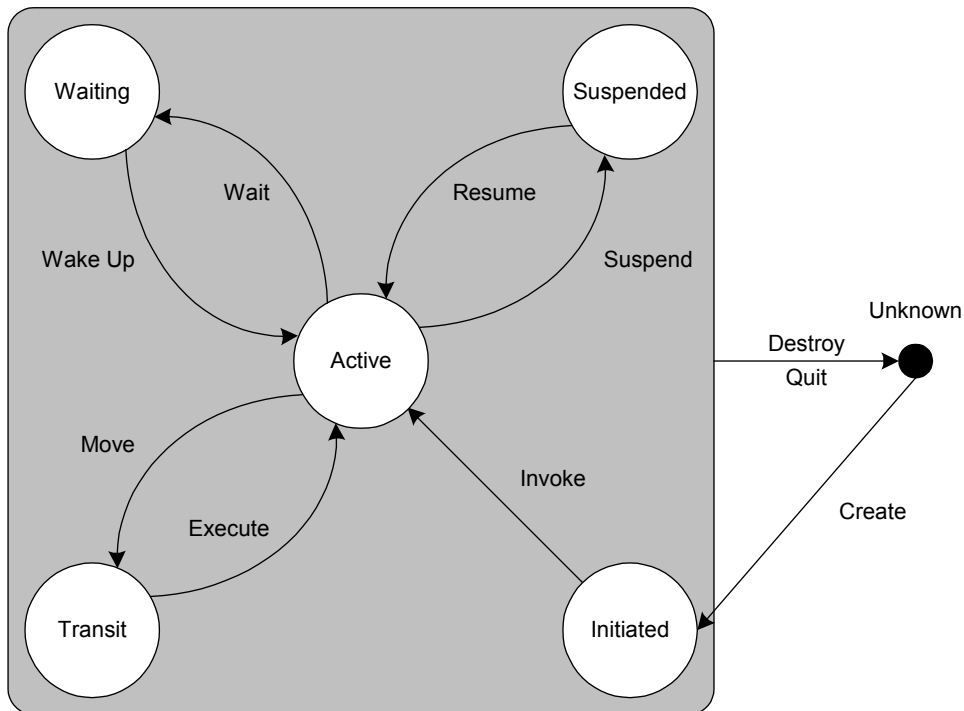
345 5 Agent Platform

346 5.1 Agent Life Cycle

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

351
 352 The life cycle of a FIPA agent is (see *Figure 2*):
 353

- 354 • **AP Bounded**
 355 An agent is physically managed within an AP and the life cycle of a static agent is therefore always bounded to a
 356 specific AP.
- 357 • **Application Independent**
 358 The life cycle model is independent from any application system and it defines only the states and the transitions of
 359 the agent service in its life cycle.
- 360 • **Instance-Oriented**
 361 The agent described in the life cycle model is assumed to be an instance (that is, an agent which has unique name
 362 and is executed independently).
- 363 • **Unique**
 364 Each agent has only one AP life cycle state at any time and within only one AP.
 365
 366
 367
 368



369
 370
 371 **Figure 2: Agent Life Cycle**
 372

373 The followings are the responsibility that an AMS, on behalf of the AP, has with regard to message delivery in each
 374 state of the life cycle of an agent:
 375

- 376 • **Active**

377 The MTS delivers messages to the agent as normal.

378
379 • **Initiated/Waiting/Suspended**

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

382
383 • **Transit**

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

388
389 • **Unknown**

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

392
393 The state transitions of agents can be described as:

394
395 • **Create**

396 The creation or installation of a new agent.

397
398 • **Invoke**

399 The invocation of a new agent.

400
401 • **Destroy**

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

403
404 • **Quit**

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

406
407 • **Suspend**

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

409
410 • **Resume**

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

412
413 • **Wait**

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

415
416 • **Wake Up**

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

418
419 The following two transitions are only used by mobile agents:

420
421 • **Move**

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

423
424 • **Execute**

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

426
427 **5.2 Agent Registration**

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

- 429
430 • The agent was created on the AP.

- 431
- 432 • The agent migrated to the AP, for those APs which support agent mobility.
- 433
- 434 • The agent explicitly registered with the AP.
- 435

436 Agent registration involves registering an AID with the AMS. When an agent is either created or 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 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

440

441 For example:

442

```

443 (request
444   :sender
445     (agent-identifier
446      :name discovery-agent@bar.com
447      :addresses (sequence iiop://bar.com/acc))
448   :receiver (set
449             (agent-identifier
450              :name ams@foo.com
451              :addresses (sequence iiop://foo.com/acc)))
452   :ontology fipa-agent-management
453   :language fipa-sl0
454   :protocol fipa-request
455   :content
456     "(action
457      (agent-identifier
458       :name ams@foo.com
459       :addresses (sequence iiop://foo.com/acc))
460      (register
461       (:ams-description
462        :name
463         (agent-identifier
464          :name discovery-agent@bar.com
465          :addresses (sequence iiop://bar.com/acc))
466         ...)))")

```

467

468 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`.

469

470

471

472

473 5.2.1 Registration Lease Times

474 To enable the DF to manage a maintainable number of registrations over a long period of time, the DF may implement lease times using the `lease-time` parameter of a `df-agent-description`. A lease time is either a duration of 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 can be removed from the DF registration database.

475

476

477

478

479 When an agent wishes to register with a DF, it can specify a lease time which is how long it would like the registration 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-time` parameter in the content of the `inform` reply will be the same. Consequently, when the lease time expires, the registration will be silently removed by the DF. On the other hand, if the lease time is not acceptable to the DF, then the DF can include a *new* lease time as the value of the `lease-time` parameter in the content of the `inform` reply. This is the case when an agent does not specify a lease time in its registration.

480

481

482

483

484

485

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

489
 490 For example, and agent may register the following `df-agent-description`:

```
491 (request
492   ...
493   :content
494     ((action
495       (agent-identifier
496         :name df@foo.com
497         :addresses (sequence iiop://foo.com/acc))
498       (register
499         (df-agent-description
500           :name
501             (agent-identifier
502               :name dummy@foo.com
503               :addresses (sequence iiop://foo.com/acc))
504             :protocols fipa-request
505             :ontologies (set fipa-agent-management)
506             :languages (set fipa-sl0)
507             :lease-time +000000000T600000000T
508             ..."))
509     ..."))
```

510
 511 Then if the DF agrees to this lease time, it will reply with and `inform` which contains the same value for the `lease-time`
 512 parameter:

```
513 (inform
514   ...
515   :content
516     ((done
517       (action
518         (agent-identifier
519           :name df@foo.com
520           :addresses (sequence iiop://foo.com/acc))
521         (register
522           (df-agent-description
523             :name
524               (agent-identifier
525                 :name dummy@foo.com
526                 :addresses (sequence iiop://foo.com/acc))
527               :protocols (set fipa-request application-protocol)
528               :ontologies (set meeting-scheduler)
529               :languages (set fipa-sl0 kif)
530               :lease-time +000000000T600000000T
531               ..."))
532     ..."))
```

533
 534 If an agent wishes to renew a lease time, then it can use the `modify` action to specify a new value for the `lease-time`
 535 parameter. The verification of this lease time goes through the same procedure mentioned in the last paragraph: if it is
 536 okay, then the value of the `lease-time` parameter in the content of the `inform` reply will be the same, if it is not okay,
 537 the value of the `lease-time` parameter in the content of the `inform` reply will be a new value which is acceptable to
 538 the DF.

539

540 6 Agent Management Ontology

541 6.1 Object Descriptions

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

545
 546 This ontology does not specify any specific positional order to encode the parameters of the objects. Therefore, it is
 547 required to encode objects in SL by specifying both the parameter name and the parameter value (see Section 3.6 of
 548 [FIPA00008]).

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

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

567 6.1.1 Agent Identifier Description

568 This type of object represents the identification of the agent. The `addresses` parameter and the name resolution
 569 mechanism (see Section 3.2), is a reification of the notion of Locator from [FIPA00001]. See also Section 3.3.7 in FIPA
 570 Agent Message Transport Service [FIPA00067] specifications.

571

Frame	agent-identifier			
Ontology	fipa-agent-management			
Parameter	Description	Presence	Type	Reserved Values
name	The symbolic name of the agent.	Mandatory	word	<code>df@hap_name</code> <code>ams@hap_name</code>
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	

572

573 **6.1.2 Directory Facilitator Agent Description**574 This type of object represents the description that can be registered with the DF service. This is a reification of the
575 Agent Directory Entry from [FIPA00001].
576

Parameter	Description	Presence	Type	Reserved Values
name	The identifier of the agent.	Optional	agent-identifier ¹⁰	
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-s1 fipa-s10 fipa-s11 fipa-s12
lease-time	The duration or time at which the lease for this registration will expire ¹¹ .	Optional	datetime ¹²	

577

578 **6.1.3 Service Description**579 This type of object represents the description of each service registered with the DF.
580

Parameter	Description	Presence	Type	Reserved Values
name	The name of the service.	Optional	string	
type	The type of the service.	Optional	string	fipa-df ¹³ fipa-ams
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	

581

¹⁰ 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.

¹¹ The default value for a lease time is assumed to be unlimited.

¹² 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.

¹³ These reserved values denote agents that provide the DF or AMS services as defined Section 4.

582 **6.1.4 Search Constraints**

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

584

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 ¹⁴ . A negative value indicates that the sender agent is willing to allow the search to propagate across all DFs.	Optional	integer			
max-results	The maximum number of results to return for the search ¹⁵ . 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			

585

586 **6.1.5 Agent Management System Agent Description**

587 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].

588

589

Frame Ontology	ams-agent-description fipa-agent-management	Parameter	Description	Presence	Type	Reserved Values
name	The identifier of the agent.	Optional	agent-identifier ¹⁶			
ownership	The owner of the agent.	Optional	string			
state	The life cycle state of the agent.	Optional	string		initiated active suspended waiting transit	

590

591 **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			
ap-services	The set of services provided by this AP to the resident agents.	Optional	Set of ap-service			

592

¹⁴ The default value for max-depth is 0.¹⁵ The default value for max-results is 1.¹⁶ 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.

593 **6.1.7 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	

594

595 **6.1.8 Property Template**

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

597

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	

598

599 **6.2 Function Descriptions**600 The following tables define usage and semantics of the functions that are part of the `fipa-agent-management`
601 ontology and that are supported by the agent management services and agents on the AP.

602

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

606

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

608

- 609 • **Function.** This is the symbol that identifies the function in the ontology.
- 610
- 611 • **Ontology.** This is the name of the ontology, whose domain of discourse includes the function described in the
612 table.
- 613
- 614 • **Supported by.** This is the type of agent that supports this function.
- 615
- 616 • **Description.** This is a natural language description of the semantics of the function.
- 617
- 618 • **Domain.** This indicates the domain over which the function is defined. The arguments passed to the function must
619 belong to the set identified by the domain.
- 620
- 621 • **Range.** This indicates the range to which the function maps the symbols of the domain. The result of the function is
622 a symbol belonging to the set identified by the range.
- 623
- 624 • **Arity.** This indicates the number of arguments that a function takes. If a function can take an arbitrary number of
625 arguments, then its arity is undefined.
- 626

627 **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

628

629 **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

630

631 **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

632

633 **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	df-agent-description / ams-agent-description ¹⁷ 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

¹⁷ Where \times is Cartesian product.

634

635 6.2.4.1 Matching Criterion

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

638

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

643

644 A registered object matches an object template if:

645

646 1. The class name of the object (that is, the object type) is the same as the class name of the object description
 647 template, and,

648

649 2. Each parameter of the object template is matched by a parameter of the object description.

650

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

653

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

658

659 The `search` action evaluates functional expressions before the object template is matched against the descriptions
 660 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
 661 (for example, `(plus 2 3)`), then what is seen by the matching process is the result of evaluating the functional term
 662 within the context of the receiving agent. A constant matches a constant template if they are equal.

663

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

666

667 1. An empty sequence matches an empty sequence, and,

668

669 2. The sequence `(cons x sequence1)`¹⁸ matches the sequence template `(cons y sequence2)` if:

670

- 671 • `x` matches `y` and `sequence1` matches `sequence2`, or,

672

- 673 • `sequence1` matches `(cons y sequence2)`.

674

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

677

678 6.2.4.2 Matching Example

679 The following DF agent description:

680

```
681 (df-agent-description
```

```
682   :name
```

```
683     (agent-identifier
```

```
684       :name cameraproxy1@foo.com
```

```
685       :addresses (sequence iiop://foo.com/acc))
```

¹⁸ `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)`.

```

686 :services (set
687   (service-description
688     :name description-delivery-1
689     :type description-delivery
690     :ontologies (set traffic-surveillance-domain)
691     :properties (set
692       (property
693         :name camera-id
694         :value camera1)
695       (property
696         :name baud-rate
697         :value 1)))
698   (service-description
699     :name agent-feedback-information-1
700     :type agent-feedback-information
701     :ontologies (set traffic-surveillance-domain)
702     :properties (set
703       (property
704         :name camera-id
705         :value camera1))))
706 :protocols (set fipa-request fipa-query)
707 :ontologies (set traffic-surveillance-domain fipa-agent-management)
708 :languages (set fipa-sl)
709

```

will match the following DF agent description template:

```

711 (df-agent-description
712   :services (set
713     (service-description
714       :type description-delivery
715       :ontologies (set traffic-surveillance-domain)
716       :properties (set
717         (property
718           :name camera-id
719           :value camera1))
720       :languages (set fipa-sl fipa-sl1))
721   )
722 )

```

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	<code>get-description</code>
Ontology	<code>fipa-agent-management</code>
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	<code>ap-description</code>
Arity	0

729

730

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 propositions that evaluate to true under the exceptional circumstances. This section describes the standard set of

736 predicates (defined over a set of arguments) and propositional symbols in the domain of discourse of the `fipa-`
 737 `agent-management` ontology.
 738

739 6.3.1 Exception Selection

740 The following rules are adopted to select the appropriate communicative act that will be returned in when a
 741 management action causes an exception:
 742

- 743 • If the communicative act is not understood by the receiving agent, then the replied communicative act is `not-`
 744 `understood`.
- 745
- 746 • If the requested action is not supported by the receiving agent, then the communicative act is `refuse`.
- 747
- 748 • If the requested action is supported by the receiving agent but the sending agent is not authorised to request the
 749 function, then the communicative act is `refuse`.
- 750
- 751 • If the requested function is supported by the receiving agent and the client agent is authorised to request the
 752 function but the function is syntactically or semantically ill-specified, then the communicative act is `refuse`.
- 753
- 754 • In all the other cases the receiving agent sends to the sending agent a communicative act of type `agree`.
 755 Subsequently if any condition arises that prevents the receiving agent from successfully completing the requested
 756 function, then the communicative act is `failure`.
 757

758 6.3.2 Exception Classes

759 There are four main classes or exceptions that can be generated in response to a management action request:
 760

- 761 • `unsupported`: The communicative act and the content has been understood by the receiving agent, but it is not
 762 supported.
- 763
- 764 • `unrecognised`: The content has not been understood by the receiving agent.
- 765
- 766 • `unexpected`: The content has been understood by the receiving agent, but it includes something that was
 767 unexpected.
- 768
- 769 • `missing`: The content has been understood by the receiving agent, but something that was expected is missing.
 770

771 6.3.3 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.
--------------------	--------	--

772

773

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 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.

774

775

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.

776

777 **7 Agent Management Content Language**

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

779

780

8 References

- 781 [FIPA00001] FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents, 2000.
782 <http://www.fipa.org/specs/fipa00001/>
- 783 [FIPA00008] FIPA SL Content Language Specification. Foundation for Intelligent Physical Agents, 2000.
784 <http://www.fipa.org/specs/fipa00008/>
- 785 [FIPA00025] FIPA Interaction Protocol Library Specification. Foundation for Intelligent Physical Agents, 2000.
786 <http://www.fipa.org/specs/fipa00025/>
- 787 [FIPA00026] FIPA Request Interaction Protocol Specification. Foundation for Intelligent Physical Agents, 2000.
788 <http://www.fipa.org/specs/fipa00026/>
- 789 [FIPA00067] FIPA Agent Message Transport Service Specification. Foundation for Intelligent Physical Agents, 2000.
790 <http://www.fipa.org/specs/fipa00067/>
- 791 [FIPA00079] FIPA Agent Software Integration Specification. Foundation for Intelligent Physical Agents, 2000.
792 <http://www.fipa.org/specs/fipa00079/>
- 793 [RFC2396] Uniform Resource Identifiers: Generic Syntax. Request for Comments, 1992.
794 <http://www.ietf.org/rfc/rfc2396.txt>
795

9 Informative Annex A — Dialogue Examples

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

```

798 (request
799   :sender
800     (agent-identifier
801       :name dummy@foo.com
802       :addresses (sequence iiop://foo.com/acc))
803   :receiver (set
804     (agent-identifier
805       :name ams@foo.com
806       :addresses (sequence iiop://foo.com/acc)))
807   :language fipa-sl0
808   :protocol fipa-request
809   :ontology fipa-agent-management
810   :content
811     "(action
812       (agent-identifier
813         :name ams@foo.com
814         :addresses (sequence iiop://foo.com/acc))
815       (register
816         (ams-agent-description
817           :name
818             (agent-identifier
819               :name dummy@foo.com
820               :addresses (sequence iiop://foo.com/acc))
821             :state active))))")

```

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

```

825 (agree
826   :sender
827     (agent-identifier
828       :name ams@foo.com
829       :addresses (sequence iiop://foo.com/acc))
830   :receiver (set
831     (agent-identifier
832       :name dummy@foo.com
833       :addresses (sequence iiop://foo.com/acc)))
834   :language fipa-sl0
835   :protocol fipa-request
836   :ontology fipa-agent-management
837   :content
838     "(action
839       (agent-identifier
840         :name ams@foo.com
841         :addresses (sequence iiop://foo.com/acc))
842       (register
843         (ams-agent-description
844           :name
845             (agent-identifier
846               :name dummy@foo.com
847               :addresses (sequence iiop://foo.com/acc))
848             :state active)))
849     true)")
851 (inform
852   :sender
853     (agent-identifier
854       :name ams@foo.com
855       :addresses (sequence iiop://foo.com/acc))

```

```

856 :receiver (set
857   (agent-identifier
858     :name dummy@foo.com
859     :addresses (sequence iiop://foo.com/acc)))
860 :language fipa-s10
861 :protocol fipa-request
862 :ontology fipa-agent-management
863 :content
864   "((done
865     (action
866       (agent-identifier
867         :name ams@foo.com
868         :addresses (sequence iiop://foo.com/acc))
869       (register
870         (ams-agent-description
871           :name
872             (agent-identifier
873               :name dummy@foo.com
874               :addresses (sequence iiop://foo.com/acc))
875             :state active))))))")
876

```

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

```

877
878 (request
879   :sender
880     (agent-identifier
881       :name dummy@foo.com
882       :addresses (sequence iiop://foo.com/acc))
883   :receiver (set
884     (agent-identifier
885       :name df@foo.com
886       :addresses (sequence iiop://foo.com/acc)))
887   :language fipa-s10
888   :protocol fipa-request
889   :ontology fipa-agent-management
890   :content
891     "((action
892       (agent-identifier
893         :name df@foo.com
894         :addresses (sequence iiop://foo.com/acc))
895       (register
896         (df-agent-description
897           :name
898             (agent-identifier
899               :name dummy@foo.com
900               :addresses (sequence iiop://foo.com/acc))
901             :protocols (set fipa-request application-protocol)
902             :ontologies (set meeting-scheduler)
903             :languages (set fipa-s10 kif)
904             :services (set
905               (service-description
906                 :name profiling
907                 :type user-profiling
908                 :ontologies (set meeting-scheduler)
909                 :properties (set
910                   (property
911                     :name learning-algorithm
912                     :value bbn)
913                   (property
914                     :name max-nodes
915                     :value 1000000))))))))))")
916

```

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

```

917 (agree
918   :sender
919     (agent-identifier
920      :name df@foo.com
921      :addresses (sequence iiop://foo.com/acc))
922   :receiver (set
923     (agent-identifier
924      :name dummy@foo.com
925      :addresses (sequence iiop://foo.com/acc)))
926   :language fipa-sl0
927   :protocol fipa-request
928   :ontology fipa-agent-management
929   :content
930     "((action
931      (agent-identifier
932       :name df@foo.com
933       :addresses (sequence iiop://foo.com/acc))
934      (register
935       (df-agent-description
936        :name
937          (agent-identifier
938           :name dummy@foo.com
939           :addresses (sequence iiop://foo.com/acc))
940          :protocols (set fipa-request application-protocol)
941          :ontologies (set meeting-scheduler)
942          :languages (set fipa-sl0 kif)
943          :services (set
944            (service-description
945             :name profiling
946             :type user-profiling
947             :ontologies (set meeting-scheduler)
948             :properties (set
949              (property
950               :name learning-algorithm
951               :value bbn)
952              (property
953               :name max-nodes
954               :value 10000000))))))
955      true))"
956   )
957 (inform
958   :sender
959     (agent-identifier
960      :name df@foo.com
961      :addresses (sequence iiop://foo.com/acc))
962   :receiver (set
963     (agent-identifier
964      :name dummy@foo.com
965      :addresses (sequence iiop://foo.com/acc)))
966   :language fipa-sl0
967   :protocol fipa-request
968   :ontology fipa-agent-management
969   :content
970     "((done
971      (action
972       (agent-identifier
973        :name df@foo.com
974        :addresses (sequence iiop://foo.com/acc))
975      (register
976       (df-agent-description
977        :name
978          (agent-identifier
979           :name
980            (agent-identifier

```

```

981         :name dummy@foo.com
982         :addresses (sequence iiop://foo.com/acc))
983     :protocols (set fipa-request application-protocol)
984     :ontologies (set meeting-scheduler)
985     :languages (set fipa-sl0 kif)
986     :services (set
987         (service-description
988             :name profiling
989             :type user-profiling
990             :ontologies (set meeting-scheduler)
991             :properties (set
992                 (property
993                     :name learning-algorithm
994                     :value bbn)
995                 (property
996                     :name max-nodes
997                     :value 10000000)))))))))")
998

```

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

```

1000
1001 (request
1002   :sender
1003     (agent-identifier
1004       :name dummy@foo.com
1005       :addresses (sequence iiop://foo.com/acc))
1006   :receiver (set
1007     (agent-identifier
1008       :name df@foo.com
1009       :addresses (sequence iiop://foo.com/acc)))
1010   :language fipa-sl0
1011   :protocol fipa-request
1012   :ontology fipa-agent-management
1013   :content
1014     "(action
1015       (agent-identifier
1016         :name df@foo.com
1017         :addresses (sequence iiop://foo.com/acc))
1018       (search
1019         (df-agent-description
1020           :ontologies (set meeting-scheduler)
1021           :languages (set fipa-sl0 kif)
1022           :services (set
1023             (service-description
1024               :name profiling
1025               :type meeting-scheduler-service)))
1026         (search-constraints
1027           :min-depth 2))))")
1028
1029 (agree
1030   :sender
1031     (agent-identifier
1032       :name df@foo.com
1033       :addresses (sequence iiop://foo.com/acc))
1034   :receiver (set
1035     (agent-identifier
1036       :name dummy@foo.com
1037       :addresses (sequence iiop://foo.com/acc)))
1038   :language fipa-sl0
1039   :protocol fipa-request
1040   :ontology fipa-agent-management
1041   :content
1042     "(action
1043       (agent-identifier

```

```

1044         :name df@foo.com
1045         :addresses (sequence iiop://foo.com/acc))
1046     (search
1047       (df-agent-description
1048         :ontologies (set meeting-scheduler)
1049         :languages (set fipa-sl0 kif)
1050         :services (set
1051           (service-description
1052             :name profiling
1053             :type meeting-scheduler-service))
1054         (search-constraint :max-depth 2))))
1055     true)")
1056
1057 (inform
1058   :sender
1059     (agent-identifier
1060       :name df@foo.com
1061       :addresses (sequence iiop://foo.com/acc))
1062   :receiver (set
1063     (agent-identifier
1064       :name dummy@foo.com
1065       :addresses (sequence iiop://foo.com/acc)))
1066   :language fipa-sl0
1067   :protocol fipa-request
1068   :ontology fipa-agent-management
1069   :content
1070     "(result
1071       (action
1072         (agent-identifier
1073           :name df@foo.com
1074           :addresses (sequence iiop://foo.com/acc))
1075         (search
1076           (df-agent-description
1077             :ontologies (set meeting-scheduler)
1078             :languages (set fipa-sl0 kif)
1079             :services (set
1080               (service-description
1081                 :name profiling
1082                 :type meeting-scheduler-service))
1083             (search-constraint :max-depth 2))))
1084         (set
1085           (df-agent-description
1086             :name
1087               (agent-identifier
1088                 :name scheduler-agent@foo.com
1089                 :addresses (sequence iiop://foo.com/acc))
1090             :ontologies (set meeting-scheduler fipa-agent-management)
1091             :languages (set fipa-sl0 fipa-sl1 kif)
1092             :services (set
1093               (service-description
1094                 :name profiling
1095                 :type meeting-scheduler-service)
1096               (service-description
1097                 :name profiling
1098                 :type user-profiling-service)))))))))")
1099

```

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

```

l100 (request
l101   :sender
l102     (agent-identifier
l103       :name dummy@foo.com
l104       :addresses (sequence iiop://foo.com/acc))
l105   :receiver (set
l106     (agent-identifier
l107       :name df@foo.com
l108       :addresses (sequence iiop://foo.com/acc)))
l109   :language fipa-s10
l110   :protocol fipa-request
l111   :ontology fipa-agent-management
l112   :content
l113     "(action
l114       (agent-identifier
l115         :name df@foo.com
l116         :addresses (sequence (iiop://foo.com/acc))
l117       (modify
l118         (df-agent-description
l119           :name
l120             (agent-identifier
l121               :name scheduler-agent@foo.com
l122               :addresses (sequence iiop://foo.com/acc))
l123             :ontologies (set meeting-scheduler)
l124             :languages (set fipa-s10 kif)
l125             :services (set
l126               (service-description
l127                 :name profiling
l128                 :type meeting-scheduler-service)))))))))")
l129
l130 (refuse
l131   :sender
l132     (agent-identifier
l133       :name df@foo.com
l134       :addresses (sequence iiop://foo.com/acc))
l135   :receiver (set
l136     (agent-identifier
l137       :name dummy@foo.com
l138       :addresses (sequence iiop://foo.com/acc)))
l139   :language fipa-s10
l140   :protocol fipa-request
l141   :ontology fipa-agent-management
l142   :content
l143     "(action
l144       (agent-identifier
l145         :name df@foo.com
l146         :addresses (sequence iiop://foo.com/acc))
l147       (modify
l148         (df-agent-description
l149           :name
l150             (agent-identifier
l151               :name scheduler-agent@foo.com
l152               :addresses (sequence iiop://foo.com/acc))
l153             :ontologies (set meeting-scheduler)
l154             :languages (set fipa-s10 kif)
l155             :services (set
l156               (service-description
l157                 :name profiling
l158                 :type meeting-scheduler-service)))))))))
l159   unauthorised)")

```


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

```

l163 (propose
l164   :sender
l165     (agent-identifier
l166       :name dummy@foo.com
l167       :addresses (sequence iiop://foo.com/acc))
l168   :receiver (set
l169     (agent-identifier
l170       :name df@foo.com
l171       :addresses (sequence iiop://foo.com/acc)))
l172   :language fipa-s10
l173   :protocol fipa-request
l174   :ontology fipa-agent-management
l175   :content
l176     "(action
l177       (agent-identifier
l178         :name df@foo.com
l179         :addresses (sequence iiop://foo.com/acc))
l180       (deregister
l181         (df-agent-description
l182           :name
l183             (agent-identifier
l184               :name dummy@foo.com
l185               :addresses (sequence iiop://foo.com/acc))))))")
l186 (not-understood
l187   :sender
l188     (agent-identifier
l189       :name df@foo.com
l190       :addresses (sequence iiop://foo.com/acc))
l191   :receiver (set
l192     (agent-identifier
l193       :name dummy@foo.com
l194       :addresses (sequence iiop://foo.com/acc)))
l195   :language fipa-s10
l196   :protocol fipa-request
l197   :ontology fipa-agent-management
l198   :content
l199     "(propose
l200       :sender
l201         (agent-identifier
l202           :name dummy@foo.com
l203           :addresses (sequence iiop://foo.com/acc))
l204       :receiver (set
l205         (agent-identifier
l206           :name df@foo.com
l207           :addresses (sequence iiop://foo.com/acc)))
l208       :language fipa-s10
l209       :protocol fipa-request
l210       :ontology fipa-agent-management
l211       :content
l212         \"((action
l213           (agent-identifier
l214             :name df@foo.com
l215             :addresses (sequence iiop://foo.com/acc))
l216           (deregister
l217             (df-agent-description
l218               :name
l219                 (agent-identifier
l220                   :name dummy@foo.com
l221                   :addresses (sequence iiop://foo.com/acc))))))\"
l222       (unsupported-act propose)))")

```

10 Informative Annex B — ChangeLog

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

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

10.2 2002/11/01 - version I by TC X2S

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

|278 **Page 14, line 512:** Removed **mobility** parameter from **ap-description**
|279 **Page 14, line 512:** Removed **dynamic** parameter from **ap-description**
|280 **Page 14, line 512:** Changed name of **transport-profile** parameter to **ap-service**
|281 **Page 14, line 512:** Changed the plurality of the **address** parameter
|282 Page 15, line 521: Added note on how to encode objects in SL
|283 Page 14, line 548: Modified `search` action to handle both `ams-agent-description` and `df-agent-`
|284 `description`
|285 Page 17, line 588: Removed the incorrect word 'template' at the end of the sentence
|286 Page 17, line 609: Changed `1MHZ` to `1` in example
|287 **Page 18, line 642:** Removed **quit** function
|288 **Page 18, lines 647-649:** Changed the exception model from predicates which return true to propositions that
|289 **evaluate to true**
|290