

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Agent Software Integration Specification

Document title	FIPA Agent Software Integration Specification		
Document number	XC00079B	Document source	FIPA Architecture Board
Document status	Experimental	Date of this status	2001/08/10
Supersedes	FIPA00012		
Contact	fab@fipa.org		
Change history			
2000/06/14	Approved for Experimental		
2001/08/10	Line numbering added		

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

Geneva, Switzerland

Notice

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

19 **Foreword**

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

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

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

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

38 Contents

39	1	Scope	1
40	2	Overview	2
41	3	Agent Software Integration Reference Model	4
42	3.1	Agent Resource Broker	5
43	3.1.1	Conformance of an Agent Resource Broker	6
44	3.2	Wrapper Service	6
45	3.2.1	Conformance of a Wrapper Service	7
46	4	Agent Resource Broker Ontology	9
47	4.1	Object Descriptions	9
48	4.1.1	Service Description	9
49	4.1.2	Communication Properties	10
50	4.2	Function Descriptions	10
51	4.2.1	Registration of a Software Object	11
52	4.2.2	Deregistration of a Software Object	11
53	4.2.3	Modification of a Software Object Registration	11
54	4.2.4	Search for an Software Object Registration	11
55	4.3	Predicates	12
56	4.3.1	Registered	12
57	4.3.2	Member	12
58	4.4	Exceptions	12
59	4.4.1	Failure Exception Propositions	12
60	5	Wrapper Ontology	13
61	5.1	Object Descriptions	13
62	5.2	Function Descriptions	13
63	5.2.1	Initialise a Software System	13
64	5.2.2	Terminate a Connection to a Software System	13
65	5.2.3	Store the State of a Software System	13
66	5.2.4	Retrieval of the State of a Software System	14
67	5.2.5	Subscribe to a Software System Event	14
68	5.2.6	Unsubscribe from a Software System Event	14
69	5.2.7	Suspend a Software System	14
70	5.2.8	Resume a Software System	15
71	5.2.9	Invoke an Action on a Software System	15
72	5.2.10	Achieve a Predicate	15
73	5.3	Predicates	15
74	5.3.1	Member	15
75	5.3.2	Parameter	15
76	5.3.3	Subscribed	15
77	5.3.4	Operation	16
78	5.4	Exceptions	16
79	5.4.1	Failure Exception Propositions	16
80	6	References	17
81			

81 **1 Scope**

82 This document provides a specification which deals with technologies enabling the integration of services provided by
83 non-agent software into a multi-agent community. It defines in general the relationship between agents and software
84 systems.

85
86 The purpose of this specification is twofold: it allows agents to describe, broker and negotiate over software systems,
87 and it allows new software services to be dynamically introduced into an agent community. This specification defines a
88 reference model, identifies agent roles (for example, broker, client, etc.) and the messages/actions which define each of
89 these roles. It builds upon [FIPA00061] and [FIPA00023].

90
91 This specification operates at the agent communication level and does not define any mappings to specific software
92 architectures; such mappings are considered outside the scope of FIPA.

93
94 This specification enables developers to build:

95
96 Wrappers for software services which are to be utilised and/or controlled by a community of agents ("public
97 services"),

98
99 Agents which provide the Agent Resource Broker (ARB) service to allow registration in a query repository and
100 management of such software services, and,

101
102 Agents ready to access such public services.

103
104 It is also intended to be used in the future by third-party developers wishing to implement new software systems ready
105 to be used by FIPA-compliant agents.

106

2 Overview

In most significant applications, agents may have a need to obtain a service by other entities in the system. Sometimes, such services could be provided by other agents. However, there are and in the future there will continue to be a wealth of non-agent software systems which provide useful services. If agents are to be truly useful they must be able to interface with and control existing software system such as databases, web-browsers, set-top boxes, speech synthesis programs and so forth.

This specification defines how software resources can be described, shared and dynamically controlled in an agent community. Software systems are characterised by software descriptions which define the nature of the software system and how to connect to it. The rationale behind this specification is to allow agents to openly share and trade software resources with each other. Allowing agents to communicate about software resources, means agents can inform each other about the existence of new software resources and thereby facilitate the dynamic inclusion and management of new software systems. This provides agents with a method by which they can dynamically acquire new capabilities.

FIPA concerns itself with how agents can connect to and control external software systems, that is systems which are external to and independent of an agents execution context. By way of contrast, internal attachment to software, where the software is included in an agents execution context is not considered in FIPA as it would require assumptions to be made about the internal implementation of agents.

Software systems come in all shapes and sizes. Many different types of interfaces are possible each with their own particular networking protocol, strengths and weaknesses. Furthermore, there are a number of emerging distribution technologies such as CORBA, DCOM and Java-RMI which are creating (competing) standards for the integration of software systems and resources. To simplify this situation and to provide the freedom to agent-programmers, this specification does not mandate the use of any particular API or distribution technology, rather it treats software integration at the agent-communication level. That is in terms of the types and contents of messages exchanged between agents. To support this, two new agent roles have been identified (see *Figure 1*):

An **Agent Resource Broker (ARB)** agent brokers a set of software descriptions to interested agents. Clients query it about what software services are available.

A **Wrapper agent** allows an agent to connect to a software system uniquely identified by a software description. Client agents can relay commands to the Wrapper agent and have them invoked on the underlying software system. The role provided by the Wrapper agent provides a single generic way for agents to interact with software systems.

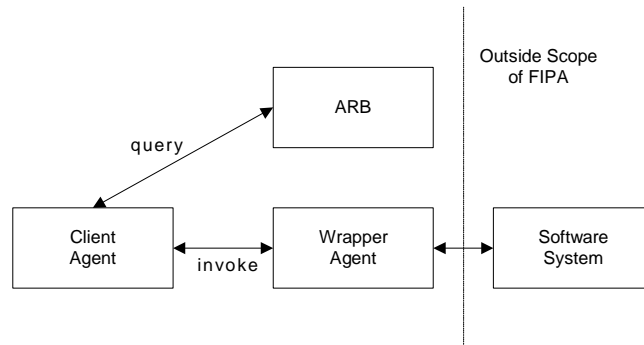


Figure 1: General Agent Software Integration Scenario

146
147
148
149

In this document we refer to ARB and Wrapper agents. However, these are defined as agent capabilities rather than explicit agent types (see *Figure 2*). Each capability is defined by an ontology (defining the syntax and semantics of a set of actions and predicates) which are supported by an agent fulfilling the corresponding ARB or Wrapper role¹.

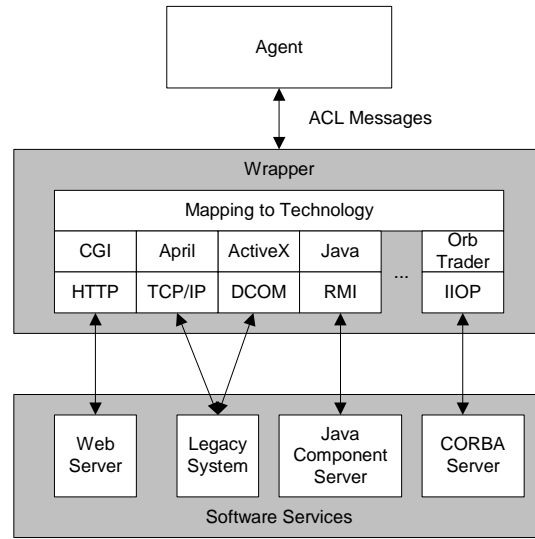


Figure 2: Layered Model for a Wrapper

150
151
152
153
154
155
156
157
158
159
160
161
162
163

Figure 3 shows three examples of possible Wrapper Agents. The top Wrapper agent provides a dedicated mapping to a legacy database over, for example say, the TCP/IP protocol. The top Wrapper agent will set-up a connection to the legacy database and will translate invocation requests from the client agent into operations on the legacy database. The bottom Wrapper agent provides a mapping to the application-level HTTP protocol, enabling the client agent to access internet resources from web-servers. Finally, the middle Wrapper agent provides a mapping to a CORBA standard Object Request Broker (ORB) allowing the client agent to manipulate an SQL database over an ORB bus. This Wrapper agent could be specialised to accessing just SQL databases using CORBA ORBs or it could be a more general Wrapper agent which supports dynamic connection to any system which has been registered with the ORB's Implementation Repository.

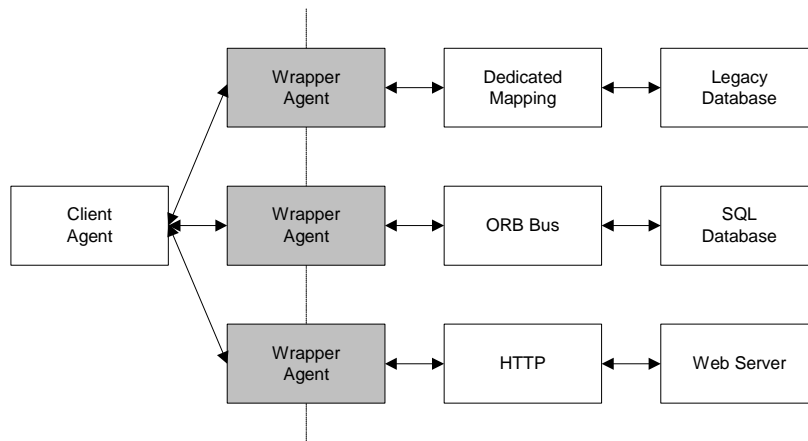


Figure 3: Example Scenario for Wrapper Agents and Software Systems

164
165
166
167
168
169
170

This specification provides details about how to find and interface with software systems in a manner which is FIPA-compliant.

¹ This specification is only concerned with the interactions between agents. How a Wrapper agent actually connects to and invokes operations on a software system is the responsibility of individual Wrapper agent developers. Wrapper agents can be specific in that they only support specific types of software systems, or they may be able to support connections to a number of different software system types.

3 Agent Software Integration Reference Model

The agent software integration reference model extends the entities defined in [FIPA00023] to include two new agent roles (see *Figure 4*):

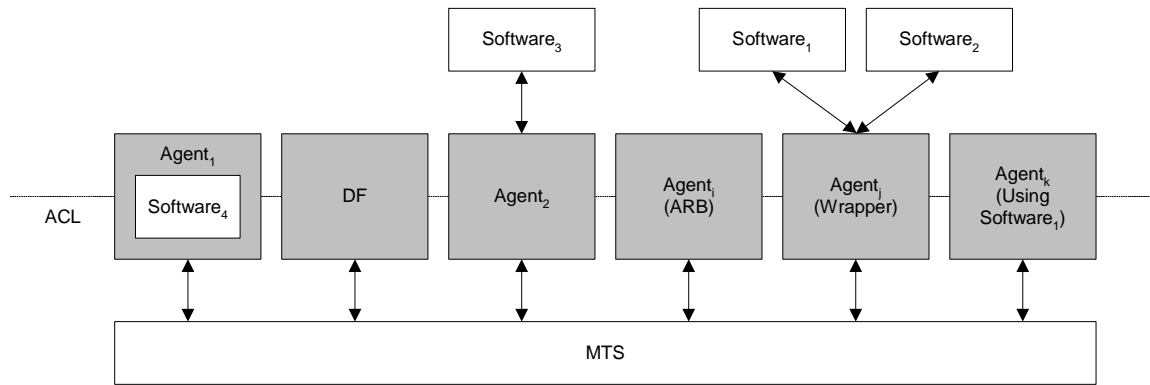


Figure 4: Agent Software Integration Reference Model

The ARB agent ($Agent_i$ in *Figure 4*) is an agent which supports the ARB capability as defined in this specification. An ARB agent brokers a set of software descriptions to interested client agents and an ARB advertises this service by registering with the DF.

Software services are described by textual software descriptions which list the properties of the software service. Part of the software description will describe where the software is located and how to interface with it (for example, networking protocols, encoding types supported). An agent providing the ARB interface supports the *FIPA-ARB* ontology (see section 4, *Agent Resource Broker Ontology*) with commands and predicates for registering and searching for software services.

The Wrapper agent ($Agent_j$ in the figure) is an agent which can dynamically interface with a software system uniquely described by a software description. The Wrapper agent will allow client agents to invoke commands on the underlying software system, translating the commands contained in ACL messages into operations on the underlying software system. Wrapper agents may be able to support multiple connections to software systems simultaneously².

A Wrapper agent supports the *FIPA-Wrapper* ontology (see section 4, *Agent Resource Broker Ontology*) with commands and predicates for initialising and issuing requests to software systems.

How a Wrapper agent is implemented and what interface exists between the Wrapper agent and the underlying software system that provides the software service is a matter for Wrapper developers and third-party tool support vendors.

A key point to remember is that Wrapper agents have the ability to dynamically manage new software devices. This is the conceptual difference between a Wrapper agent and an agent which upgrades a software service to being an agent-level service. This difference will of course be reflected in the DF. To illustrate the point consider two agents: The first agent has the capability to send and receive email and accordingly it will advertise this service in its DF entry. The second agent has the capability of connecting to an email service, it is a Wrapper agent and will accept a description of the software service required (in this case the location of the mail host and the networking protocol to use). The first agent will allow a client to send and receive email since it has a static connection to a given email server. The second agent will allow an agent to dynamically connect to a remote email service identified by a software description.

² A Wrapper agent which supports the full *FIPA-Wrapper* ontology is considered to provide more than a simple bridging function to an external software system. Such an agent implicitly provides a management functionality.

211 Client Agents ($Agent_k$ in *Figure 4*) are agents which wish to use the services provided by a software system, for
 212 example $Software_1$. They can query the DF in order to find out if an agent exists which provides an ARB service in the
 213 agent domain. Next they can query the ARB agent to see if there is a software system (identified by a software
 214 description) which meets its requirements ($Software_1$). If $Agent_k$ cannot interface directly with the software system
 215 identified by the software description returned by the ARB ($Software_1$), then it must obtain the services of an agent that
 216 can (a Wrapper agent). $Agent_k$ queries the DF to find out if there is an agent which supports the Wrapper capability for
 217 the specific software system which the software description identifies. In this example, the DF returns $Agent_j$ in
 218 response to the query. $Agent_k$ then contacts $Agent_j$ (the Wrapper agent) to initiate control of $Software_1$. The Wrapper
 219 agent ($Agent_j$) will then invoke operations on the underlying software system in response to requests sent to it by
 220 $Agent_k$.

221
 222 Some agents ($Agent_2$ in *Figure 4*) can directly interface to software systems ($Software_3$) and thus do not need work
 223 through a Wrapper agent. Such capabilities are outside the scope of this specification. It should be noted, that $Agent_2$
 224 could have obtained the address of $Software_3$ from the ARB agent.

225
 226 Other agents ($Agent_1$ in *Figure 4*) can embed private software within their execution context. This is outside the scope of
 227 this specification.

228
 229 The following is a summary of steps necessary to support the reference model:

- 230
 231 1. $Agent_i$ registers with the DF. It advertises the fact that it provides an ARB service by providing a service description
 232 with `fipa-arb` as the value of the `:type` parameter.
- 233
 234 2. $Agent_j$ registers with the DF. It advertises the fact that it provides a Wrapper service by providing a service
 235 description with `fipa-wrapper` as the value of the `:type` parameter.
- 236
 237 3. $Agent_k$ queries the DF for an agent which provides an ARB service. The DF returns the name of $Agent_i$ as satisfying
 238 the query.
- 239
 240 4. $Agent_k$ queries the $Agent_i$ for a software system which matches some specific requirements, for example a $Group_3$
 241 fax-server. $Agent_i$ returns a software description which uniquely identifies a specific software service.
- 242
 243 5. $Agent_k$ queries the DF for an agent which can provide a Wrapper service to a $Group_3$ fax-server. The DF returns the
 244 name of $Agent_j$ as satisfying the query.
- 245
 246 6. $Agent_k$ requests that $Agent_j$ initialise a connection to the $Group_3$ fax server identified by the service description (from
 247 step 4).
- 248
 249 7. $Agent_k$ requests that $Agent_j$ invoke a certain operation on the $Group_3$ fax server.
- 250
 251 8. $Agent_k$ requests that $Agent_j$ close the connection to the $Group_3$ fax server.
- 252

253 3.1 Agent Resource Broker

254 The Agent Resource Broker (ARB) is a special service that can be provided by an agent. Every agent in the domain is
 255 allowed to support this service, however it is mandatory that every agent platform which wants to support FIPA-
 256 compliant software sharing must have at least one agent that provides this ARB service. This service must be
 257 registered with the DF in order to be advertised in the agent domain.

258
 259 Every ARB agent is able to understand the `FIPA-ARB` ontology as specified in section 4, *Agent Resource Broker*
 260 *Ontology*. Therefore, in order to find an agent which provides ARB service, agents must query the Directory Facilitator
 261 (DF), whose address is by default known by all agents in the domain.

262
 263 An agent which offers the ARB service wishes to broker a set of software services for direct use by other agents.
 264 However, an ARB may not wish to simply hand over a software description in response to a query from an interested
 265 agent. It may wish to negotiate over the terms and conditions of use of the software system, request authorisation or

266 even provide permanent or evaluation keys for use with the software system. Such negotiation is application-
 267 dependent.
 268

269 **3.1.1 Conformance of an Agent Resource Broker**

270 A FIPA-compliant ARB agent must at least:

271
 272 Register the ARB service description with the DF with `fipa-arb` as the `:type` and `FIPA-ARB` as the `:ontology`,
 273

274 Implement the actions described in the `FIPA-ARB` ontology according to the behaviour and parameters specified in
 275 section 4, *Agent Resource Broker Ontology*,

276
 277 Implement and assert the predicates described in the `FIPA-ARB` ontology according to the semantics specified in
 278 section 4.3, *Predicates*,

279
 280 Create and store registration predicates in response to a successful register-software action,

281
 282 Understand the `request` [FIPA00037] communicative act to request the execution of one of the ARB actions;
 283

284 Understand the `query-if` [FIPA00037] and `query-ref` [FIPA00037] communicative acts to query its knowledge
 285 by using the `query` predicate;

286
 287 Implement the `FIPA-Request` [FIPA00026] and `FIPA-Query` [FIPA00027] interaction protocols, and,

288
 289 Implement the `not-understood` [FIPA00037], `agree` [FIPA00037], `refuse` [FIPA00037], `failure`
 290 [FIPA000437] and `inform` [FIPA00037] communicative acts in order to respond to requests and queries according
 291 to the `FIPA-Request` and `FIPA-Query` interaction protocols.
 292

293 Even if these requirements guarantee FIPA compliance, of course they are not sufficient to guarantee the usefulness of
 294 the ARB agent to the agent domain.
 295

296 **3.2 Wrapper Service**

297 Wrapper services are provided by agents. The Wrapper service allows an agent to:

298
 299 Request a dynamic connection to a software,
 300

301 Invoke operations on the software system,
 302

303 To be informed of the results of operations,
 304

305 To query the properties of the software system,
 306

307 Set the parameters of a software system,
 308

309 Subscribe to events of the software system,
 310

311 Manage the state of the service, and,
 312

313 Terminate the service.
 314

315 An agent can request of an agent which provides a Wrapper service to dynamically connect to a software system
 316 uniquely identified by a software service description. The ARB service supports the sharing and brokering of such
 317 software descriptions.
 318

319 An agent providing a Wrapper service (the Wrapper agent) can be specific to a type of software system (specifically it
 320 commits to a given software system ontology). In addition, a Wrapper can be specific about the types of connection and
 321 communication protocols it can support when interfacing with a software system, for example, HTTP, SMTP, etc. This
 322 allows client agents who wish to use the services of a Wrapper agent to discriminate between Wrapper agents on the
 323 basis of both the software systems supported and the types of connections supported.

324
 325 Wrapper agents may be able to support multiple software types and multiple service instances simultaneously. In order
 326 to allow a Wrapper agents to distinguish between concurrent services, Wrapper agents will return a `:service-`
 327 `instance-id` to the client agent on the successful completion of an `init` action. Most of the actions supported by the
 328 FIPA-Wrapper ontology require the inclusion of this `:service-instance-id`.

329
 330 A Wrapper agent has freedom on how it chooses to "wrap" a software system. The most basic integration scenario
 331 would model a software system simply as a collection of operations which can be performed on the software system.

332
 333 A more sophisticated Wrapper agent can divide the operations into three general categories. Specific actions and
 334 predicates have been included in the FIPA-Wrapper ontology to reflect and support this distinction, that is, to provide
 335 Wrapper agents with the necessary vocabulary to support such distinctions should Wrapper agent-designers wish to
 336 support them. The three categories are:

337
 338 1. Event Notification

339 The software system asynchronously notifies every agent subscribed to an event when that specific event occurs.
 340 The actions `software-subscribe` and `software-unsubscribe` actions support this activity. The
 341 `subscribed` predicate supports the querying to which events an agent is subscribed.

342
 343 2. Sensing Functions

344 The agent can require to the wrapper to be informed of the result of a function call which does not change the state
 345 of the environment and of the software system itself. The `query-ref` and `query-if` communicative acts and the
 346 `parameter` predicate support this activity.

347
 348 3. Effecting Actions

349 The agent can require the Wrapper agent to perform an action. The `invoke` action supports this function for
 350 domain-dependent operations. The `achieve` action provides a generic way to set the parameters of a software
 351 service.

352
 353 Such a categorisation allows the interfaces to different software systems to be treated in a generic component-based
 354 manner. There is a generic method for discovering what event types, parameters and operations are supported using
 355 the `query-ref` and `query-if` communicative acts in conjunction with the predicates supported by the FIPA-
 356 Wrapper ontology. The actions of the FIPA-Wrapper ontology provide a single generic way to subscribe to and
 357 unsubscribe from events, to modify parameters and to invoke operations.

358
 359 As mentioned already, a Wrapper agent does not have to provide such a component-based interface to a software
 360 system.

361
 362 **3.2.1 Conformance of a Wrapper Service**

363 A FIPA-compliant Wrapper agent must:

364
 365 Register the Wrapper service description with the DF with `fipa-wrapper` as the `:type` and FIPA-Wrapper as
 366 the `:ontology`,

367
 368 Implement the actions described in the FIPA-Wrapper ontology according to the behaviour and parameters
 369 specified in section 5, *Wrapper Ontology*,

370
 371 Implement and assert the predicates described in the FIPA-Wrapper ontology according to the semantics
 372 specified in section 5.3, *Predicates*,

373
374
375
376
377
378
379
380
381
382
383
384

Understand the `request` communicative act to request the execution of one of these Wrapper actions,

Understand the `query-if` and `query-ref` communicative acts to query its asserted predicates by using the `FIPA-Wrapper` predicates,

Implement the `FIPA-Request` and `FIPA-query` interaction protocols, and,

Implement the `not-understood`, `agree`, `refuse`, `failure`, `inform` communicative acts in order to respond to requests and queries according to the `FIPA-request` and `FIPA-Query` interaction protocols.

384 4 Agent Resource Broker Ontology

385 4.1 Object Descriptions

386 This section describes a set of frames, that represent the classes of objects in the domain of discourse within the
 387 framework of the FIPA-ARB ontology.

388
 389 The following terms are used to describe the objects of the domain:

391 **Frame.** This is the mandatory name of this entity, that must be used to represent each instance of this class.

392
 393 **Ontology.** This is the name of the ontology, whose domain of discourse includes the parameters described in the
 394 table.

395
 396 **Parameter.** This is the mandatory name of a parameter of this frame.

397
 398 **Description.** This is a natural language description of the semantics of each parameter.

399
 400 **Presence.** This indicates whether each parameter is mandatory or optional.

401
 402 **Type.** This is the type of the values of the parameter: Integer, Word, String, URL, Term, Set or Sequence.

403
 404 **Reserved Values.** This is a list of FIPA-defined constants that can assume values for this parameter.
 405

406 4.1.1 Service Description

407 This type of object represents the description of each service registered with the DF.
 408

Frame	service-description			
Ontology	FIPA-ARB			
Parameter	Description	Presence	Type	Reserved Values
name	The name of the service.	Mandatory	String	
type	The type of the service.	Mandatory	String	fipa-arb fipa-wrapper
ontology	A list of ontologies supported by the service.	Optional	Set of String	FIPA-ARB FIPA-Wrapper
protocol	A list of interaction protocols supported by the service.	Optional	Set of String	
properties	A list of properties that discriminate the service.	Optional	Set of property	
communication-properties	A list of communication methods of the service.	Mandatory	Set of communication-properties	

409
 410

410 4.1.2 Communication Properties

411 This object represents the generic, service-independent properties which describe how to connect to the service³.
 412 Communication properties are independent of any given communication protocol and they should be complete and
 413 provide the minimum information required for an agent to successfully connect directly to a software system.

414

Parameter	Description	Presence	Type	Reserved Values
net-protocol	The network protocol of the service.	Mandatory	String	IIOP SMTP HTTP
address	The transport address of the service.	Mandatory	URL	
message-body-format	The format of the message body.	Mandatory	String	FIPA-String-ACL
message-body-encoding	The encoding of the message body.	Mandatory	String	See [ISO2022]

415

416 4.2 Function Descriptions

417 The following tables define usage and semantics of the functions that are part of the FIPA-ARB ontology and that are
 418 supported by the ARB service.

419

420 The following terms are used to describe the functions of the FIPA-ARB domain:

421

422 **Function.** This is the symbol that identifies the function in the ontology.

423

424 **Ontology.** This is the name of the ontology, whose domain of discourse includes the function described in the
 425 table.

426

427 **Supported by.** This is the type of agent that supports this function.

428

429 **Description.** This is a natural language description of the semantics of the function.

430

431 **Domain.** This indicates the domain over which the function is defined. The arguments passed to the function must
 432 belong to the set identified by the domain.

433

434 **Range.** This indicates the range to which the function maps the symbols of the domain. The result of the function is
 435 a symbol belonging to the set identified by the range.

436

437 **Arity.** This indicates the number of arguments that a function takes. If a function can take an arbitrary number of
 438 arguments, then its arity is undefined.

439

440

³ It is not mandatory to return all of the communication properties in response to a query. These could be withheld by the ARB pending a successful negotiation over terms and conditions of the service.

440 **4.2.1 Registration of a Software Object**

Function	register
Ontology	FIPA-ARB
Supported by	ARB
Description	The execution of this function has the effect of registering a new software object into the knowledge base of the executing agent.
Domain	service-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

441

442 **4.2.2 Deregistration of a Software Object**

Function	deregister
Ontology	FIPA-ARB
Supported by	ARB
Description	An agent may deregister an object in order to remove all of its attributes from a directory.
Domain	service-name
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

443

444 **4.2.3 Modification of a Software Object Registration**

Function	modify
Ontology	FIPA-ARB
Supported by	ARB
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.
Domain	service-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

445

446 **4.2.4 Search for an Software Object Registration**

Function	search
Ontology	FIPA-ARB
Supported by	ARB
Description	An agent may search for an object template in order to request information from an agent, in particular from an ARB. A successful search can return one or more service descriptions that satisfy the search criteria and a null set is returned where no agent entries satisfy the search criteria.
Domain	service-description
Range	Set of <code>service-descriptions</code>
Arity	1

447

448

448 **4.3 Predicates**449 **4.3.1 Registered**

450 When an ARB agent performs a register action, it asserts the predicate:

451
452 `(registered service-description)`453
454 This predicate can be subsequently queried through the used of the `query-if` and `query-ref` communicative acts.
455456 **4.3.2 Member**

457 This predicate can be used to bind sets of expressions to iota-supplied variables:

458
459 `(member element set)`

460

461 **4.4 Exceptions**

462 The exceptions for the FIPA-ARB ontology follow the same form and rules as specified in [FIPA00023].

463 **4.4.1 Failure Exception Propositions**

Communicative Act Ontology	failure FIPA-ARB	
Predicate symbol	Arguments	Description
<code>service-name-in-use</code>	String	The specified service name is already in use; the string identifies the service name.

464

465

465 5 Wrapper Ontology

466 5.1 Object Descriptions

467 This section describes a set of frames, that represent the classes of objects in the domain of discourse within the
468 framework of the FIPA-Wrapper ontology.

469 The FIPA-Wrapper ontology shares the `service-description` and `communication-properties` objects
470 defined in section 4, *Agent Resource Broker Ontology*.
471
472

473 5.2 Function Descriptions

474 The following tables define usage and semantics of the functions that are part of the FIPA-Wrapper ontology and that
475 are supported by the Wrapper service.
476

477 5.2.1 Initialise a Software System

Function	init
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may initialise the underlying software system before use. The first argument allows an agent to differentiate between software services and the second argument can contain parameters to the software for initialisation. A successful initialisation returns the service instance identifier of the software system.
Domain	<code>service-description</code> , Set of <code>property</code>
Range	<code>service-instance-id</code>
Arity	2

478

479 5.2.2 Terminate a Connection to a Software System

Function	close
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may close the connection to a software system when it has finished with it. The first argument is the specific identifier of the software system and the second argument can contain parameters to the software for the closure of the connection.
Domain	<code>service-instance-id</code> , Set of <code>property</code>
Range	The execution of this function results in a change of the state, but it has no explicit result. Therefore there is no range set.
Arity	2

480

481 5.2.3 Store the State of a Software System

Function	store
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may store the state of a software system. A successful store results in the state identifier associated with the software system being returned.
Domain	<code>service-instance-id</code>
Range	<code>state-id</code>
Arity	1

482

483 **5.2.4 Retrieval of the State of a Software System**

Function	restore
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may restore the state of a software system.
Domain	service-instance-id
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

484

485 **5.2.5 Subscribe to a Software System Event**

Function	software-subscribe
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may subscribe to an event of a software system.
Domain	service-instance-id, event-name
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	2

486

487 **5.2.6 Unsubscribe from a Software System Event**

Function	software-subscribe
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may unsubscribe from an event of a software system.
Domain	service-instance-id, event-name
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	2

488

489 **5.2.7 Suspend a Software System**

Function	suspend
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may suspend the operation of a software system.
Domain	service-instance-id
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

490

491

491 **5.2.8 Resume a Software System**

Function	resume
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may resume the operation of a software system.
Domain	service-instance-id
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

492

493 **5.2.9 Invoke an Action on a Software System**

Function	invoke
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may invoke an action on a software system.
Domain	service-instance-id, functional-expression
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	2

494

495 **5.2.10 Achieve a Predicate**

Function	achieve
Ontology	FIPA-Wrapper
Supported by	Wrapper agent
Description	An agent may assert as true a predicate.
Domain	predicate
Range	The execution of this function results in domain-dependent result
Arity	1

496

497 **5.3 Predicates**498 **5.3.1 Member**499 The definition of this predicate is the same as that given in section 4.3.2, *Member*.

500

501 **5.3.2 Parameter**

502 When a Wrapper agent initialises the connection to a software service, a parameter predicate for each of the set of available parameters of the software system is asserted:

503

504 *(parameter service-instance-id parameter-name value)*

505

507 **5.3.3 Subscribed**

508 When the connection to a software system is initialised, for each event type supported by the software system, all event subscriptions are cancelled:

509

510 *(not (subscribed service-instance-id event-name))*

511

512

513 When an event type is subscribed to⁴, a subscribed predicate is asserted:
 514
 515 (subscribed *service-instance-id event-name*)
 516

517 **5.3.4 Operation**

518 When a Wrapper agent initialises a service, an operation predicate is asserted for each operation supported by the
 519 software system⁵:
 520
 521 (operation *service-instance-id operation-name (set argument-type)*)
 522

523 **5.4 Exceptions**

524 The exceptions for the FIPA-Wrapper ontology follow the same form and rules as specified in [FIPA00023].

525 **5.4.1 Failure Exception Propositions**

Communicative Act Ontology	failure FIPA-Wrapper	
Predicate symbol	Arguments	Description
service-in-use	String	The specified service is already in use; the string identifies the service.
service-suspended	String	The specified service has been suspended; the string identifies the service.
service-already-suspended	String	The specified service is already suspended; the string identifies the service.
service-unreachable	String	The specified service is not reachable; the string identifies the service.
maximum-number-of-instances-exceeded		The number of service instances has been exceeded.
not-storable	String	The specified state identifier is not storable; the string identifies the state.
not-retrievable	String	The specified state identifier is not retrievable; the string identifies the state.
not-suspendable	String	The specified service cannot be suspended; the string identifies the service.
not-resumable	String	The specified service cannot be resumed; the string identifies the service.
exceeded	String	A resource has been exceeded; the string identifies the resource.

526
 527

⁴ Not all software systems will support event services.

⁵ A Wrapper agent can retract these operations if the operation subsequently becomes unavailable.

6 References

- 527
- 528 [FIPA00023] FIPA Agent Management Specification, Foundation for Intelligent Physical Agents, 2000.
529 <http://www.fipa.org/specs/fipa00023/>
- 530 [FIPA00026] FIPA Request Interaction Protocol Specification, Foundation for Intelligent Physical Agents, 2000.
531 <http://www.fipa.org/specs/fipa00026/>
- 532 [FIPA00027] FIPA Query Interaction Protocol Specification, Foundation for Intelligent Physical Agents, 2000.
533 <http://www.fipa.org/specs/fipa00027/>
- 534 [FIPA00037] FIPA Communicative Act Library Specification, Foundation for Intelligent Physical Agents, 2000.
535 <http://www.fipa.org/specs/fipa00037/>
- 536 [FIPA00061] FIPA ACL Message Structure Specification, Foundation for Intelligent Physical Agents, 2000.
537 <http://www.fipa.org/specs/fipa00061/>
- 538 [ISO2022] Information Technology-Character Code Structure and Extension Techniques, International Standards
539 Organisation, 1994.
540 <http://www.iso.ch/cate/d22747.html>