

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Policies and Domains Specification

Document title	FIPA Domains and Policies Specification		
Document number	PC00089D	Document source	FIPA TC Architecture
Document status	Preliminary	Date of this status	2001/08/10
Supersedes	None		
Contact	arch@fipa.org		
Change history			
2000/12/15	An initial summary of use case scenarios and architectural elements that have been identified to support policy mechanisms in agent platforms		
2000/12/26	Edits and clean ups of initial text		
2001/01/03	Format conversion to FIPA 2000 compliance, edits and clean-up		
2001/01/29	Additional use cases and material concerning conversation policies		
2001/08/10	Line numbering added		

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

Geneva, Switzerland

Notice

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

17 **Foreword**

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

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

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

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

36 Contents

37	1	Introduction	1
38	1.1	Contents	1
39	1.2	Audience	2
40	1.3	Acknowledgements	2
41	2	Policies, Domains and Agent Platforms	3
42	3	Policy Scenarios	5
43	3.1	Access Use Case	5
44	3.1.1	Description	5
45	3.1.2	Scenario	5
46	3.2	Social Grouping Use Case	5
47	3.2.1	Description	5
48	3.2.2	Scenario	6
49	3.3	Obligation Use Case	6
50	3.3.1	Description	6
51	3.3.2	Scenario	6
52	3.4	Compositional Use Case	6
53	3.4.1	Description	6
54	3.4.2	Scenario	7
55	3.5	Refrain Use Case	7
56	3.5.1	Description	7
57	3.5.2	Scenario	7
58	3.6	Content Use Case	7
59	3.6.1	Description	7
60	3.6.2	Scenario	7
61	3.7	System Configuration Use Case	8
62	3.7.1	Description	8
63	3.7.2	Scenario	8
64	3.8	Cooperation use case	8
65	3.8.1	Description	8
66	3.8.2	Scenario	8
67	3.9	Delegation Use Case	8
68	3.9.1	Description	8
69	3.9.2	Scenario	9
70	3.10	Meta Order Use Case	9
71	3.10.1	Description	9
72	3.10.2	Scenario	9
73	3.11	High Order Use Case	9
74	3.11.1	Description	9
75	3.11.2	Scenario	9
76	3.12	Trust Use Case	9
77	3.12.1	Description	9
78	3.12.2	Scenario	9
79	4	Architectural Elements Needed to Support Policies	11
80	4.1	Introduction	11
81	4.2	Policy Structures	11
82	4.2.1	Policy	11
83	4.2.2	Policy Language	11
84	4.2.3	Policy Library	11
85	4.2.4	Interpretation Engine	12
86	4.2.5	Distribution Mechanism	12
87	4.2.6	Conversation Policy	12
88	4.3	Enforcement mechanisms	12

89	4.3.1	Guards.....	13
90	4.3.2	Sanctions.....	13
91	4.3.3	Policy Exception	13
92	4.3.4	Reputation Service	13
93	4.3.5	Policy Domain.....	13
94	4.3.6	Domain Manager	13
95	4.4	Domain Management.....	14
96	4.4.1	Directory Functions.....	14
97	4.4.2	Conflict Resolution.....	14
98	4.4.3	Policy Derivation.....	14
99	4.4.4	Policy Change Notification.....	14
100	4.4.5	Querying of Domain Policies	14
101	5	Elements for Testing Conformance	15
102	6	Communicative Acts	16
103	6.1	Promise.....	16
104	7	References.....	17
105			

1 Introduction

This document gives a set of use cases and abstract architectural elements that can be used to guide the specification of policy mechanisms in concrete Agent Platform architectures. It is based on and derives from the FIPA Abstract Architecture Specification [FIPA00001].

As this specification is defined in terms of generic abstractions rather than specific concrete elements, this specification would require reification in order to derive specifications for particular types of agent platform. However, we anticipate that the architectural elements identified here will be present in all agent platforms that support policy mechanisms.

The basic services offered by an agent platform¹ to an agent are unconstrained. An agent may register any attributes that it chooses through the Agent Directory Service; it may use the Agent Message Transport Service to communicate with any reachable agent using any available transport and messages of any size or encoding, and it may operate on behalf of any principal.

In practice, however, developers and users of multi-agent systems often wish to place strong constraints on the behaviour of agents within agent environments. This especially means being able to apply and enforce these constraints and policies across distributed agents and systems.

Typical constraints that we may wish to enforce include:

- Requiring that an agent use a particular encoding for its messages.

- Preventing an agent from communicating with non-local agents (agents which lie outside some domain, in the transport addressing sense of the word).

- Requiring than an agent select a particular quality of service (e.g. encryption, non-repudiation) when communicating with non-local agents.

- Preventing an agent from registering certain attributes with the Agent Directory Service unless it is operating on behalf of a particular principal.

- Limiting the total number of agents registered with a platform.

- Restricting access to certain host directories or setting ceilings on the amount of system resources that can be used.

All of these constraints may be expressed as constraints over agent platform services. There may be other types of constraint that we wish to apply to an agent *X* for example, requiring the use of a particular conversation policy when interacting with a particular class of agent, or preventing an agent from transmitting confidential data to a non-local agent *X* but these lie outside the scope of this specification.

1.1 Contents

This document is organized into the following sections:

- This **Introduction**.

- The **Scope and Methodology** section explains the background of this work, its purpose, and the methodology that has been followed.

- The **Policies, Domains and Agent Platforms** section is a description of the concepts and considerations necessary to the creation of policy driven agent systems.

¹ The Abstract Architecture specification does not refer to an Agent Platform. However, we use the term here informally to mean the set of inter-related services that are offered to an agent in order for it to discover other agents, and to communicate with those agents.

156
157
158
159
160
161

The **Policy Scenarios** section contains a selection of use cases illustrating the contexts in which policies are applied to agent systems.

The **Architectural Elements** section describes architecture components required.

162 **1.2 Audience**

163 The primary audience for this document is developers of concrete specifications for agent systems – specifications
164 grounded in particularly technologies, representations, and programming models. It may also be read by the users of
165 concrete specifications including implementers of agent platforms, agent systems, and gateways between agent
166 systems.

167 This document describes abstract architectural elements for guiding the creation of policy mechanisms in intentional
168 multi-agent systems. It assumes that the reader has a good understanding about the basic principles of multi-agent
169 systems. It does not provide the background material to help the reader assess whether multi-agent systems are an
170 appropriate model for their system design, nor does it provide background material on topics such as Agent
171 Communication Languages, BDI systems, or distributed computing platforms.
172
173

174 **1.3 Acknowledgements**

175 TBD.
176
177
178

178 2 Policies, Domains and Agent Platforms

179 A set of constraints is termed an **Agent Service Policy**. In this specification, we are only concerned with policies that
 180 are *public*, i.e., accessible to systems, *machine readable*, i.e., can be processed by computer systems and in particular
 181 *declarative*, i.e., amenable to inference.

182
 183 Policies may be expressed in a variety of languages. At one extreme they may be written in some propositional or
 184 constraint language such as SL2, in terms of some kind of agent platform service ontology. There are a wide variety of
 185 simpler schemes, each of which gives up some types of expressivity. The choice of language will be affected by (at
 186 least) the following considerations:

187	Composability	The ability to combine two or more policies.
188		
189	Computability	The ability to compute the legality of some service request.
190		
191	Efficiency	The resource cost of evaluating the legality of a request.
192		
193	Consistency	Whether it is possible to express \exists or detect \exists inconsistent or contradictory requirements.
194		
195	Expressivity	Whether it is possible to express the required constraints in the language.
196		
197	Equivalency	Whether it is possible to compute the functional equivalence of two policies (and so, for 198 example, reduce "legality of request" to "membership of some class associated with a given 199 policy").
200		
201		

202 We assume that there are fundamentally two kinds of **policy constraints**: those relating to **permissions** and those
 203 relating to **obligations**. Not all platforms require both kinds of policies; however this specification introduces
 204 architectural elements that correspond to both forms. These policies are often related: by entering into particular
 205 obligations an agent may acquire specific permissions; and vice versa: when an agent is given permission to access a
 206 shared resource, it may incur obligations as a result.

207
 208 Associated with policies and the mechanisms required to support policy application is the concept of a **contract**. A
 209 contract is an agreement entered into by agents and services to be constrained by one or more sets of policy
 210 constraints. In addition to the architectural elements needed by platforms to support policy mechanisms a given
 211 reification of these architectural elements may also require the **promise** communicative act. A promise is a speech act
 212 uttered by an agent when it agrees to abide by a set of policy constraints.

213
 214 Many policies are applied at the point where an agent invokes a service of the agent platform (what about invariants?).
 215 The constraints on the use of a service can be of many kinds: constraints on the parameters supplied by the agent (for
 216 example on the size or format of a message), and constraints based upon the state of the agent platform, including the
 217 history of the interactions between the agent and the platform. In order for a service request to be honored, an inference
 218 procedure must be used to verify that the applicability requirements of the service may be satisfied in relation to the
 219 policies in force.

220
 221 In principle, the inference procedure can be performed for every service request performed by every agent on a
 222 platform. However this may be prohibitively expensive from a computational standpoint. There are also situations when
 223 it is desirable to ask whether or not a request, or set of requests, would be permitted if an agent were to make them.
 224 (For example, a mobile agent might wish to know this before deciding whether to move to a particular agent platform.)

225
 226 It is common to associate **policy mechanisms** with **policy domains**. A policy domain is simply a set of agents that is
 227 characterized by a set of policies. However, there are many benefits to constructing explicit domains: as aids to
 228 efficiently applying policies for example. The infrastructure needed to support policy domains typically includes
 229 constructs such as **domain managers**, etc.

230

231 Policy domains enable agent users to be assured of policy uniformity across multiple platforms and hosts, as long as
232 semantically equivalent monitoring and enforcement mechanisms are available across those platforms and hosts.
233 Under these conditions, it follows that a given domain could extend across host boundaries and, conversely, multiple
234 domains could exist concurrently on the same host. With respect to platform independence, it should be possible for
235 agents running on the same platform to be in different domains (for example, a resident and a visiting mobile agent
236 running on the same platform may belong to different domains having more or less restrictive security privileges).

237
238 It is easy to imagine that agents might want to simultaneously belong to multiple domains. For example, it might be
239 useful to structure an agent application as a series of hierarchically nested sub-domains. It might also be useful in some
240 instances to specify a policy that precludes an agent from simultaneously belonging to more than one domain (e.g., if
241 two domains are governed by incompatible security policies). Simultaneous membership of agents in multiple domains
242 raises a number of currently unsolved technical issues.

243

243 **3 Policy Scenarios**

244 In this section, we identify a number of use case scenarios that illustrate many of the classical situations where policies
 245 and policy enforcement are relevant. They cover a range of situations that we may expect to encounter in policy
 246 application.

247
 248 The scenarios are abstract in nature, rather than examples of concrete situations. Their emphasis is on illustrating the
 249 many different situations that policies may be applied and the kind of architectural support that would be required on
 250 Agent Platforms in order to support them. For the sake of continuity and comparison each of the use case scenarios is
 251 expressed using aspects of lawyer-client interaction.

252
 253

254 **3.1 Access Use Case**

255 **3.1.1 Description**

256 Many policies relate to the provision of shared resources to agents. Shared resources are often constrained by quality
 257 of service constraints, access constraints and availability constraints. A key aspect of this class of policy scenarios is
 258 that an *owner* of each resource must be identifiable (which may or may not be an agent) and that an *owner* be
 259 responsible for applying any policy constraints to the resource.

260
 261
 262
 263

This scenario is characterized by a set of resources, methods for accessing those resources, ownership of the
 resources and quality of service constraints upon the resources.

264 **3.1.2 Scenario**

265 The resource may be viewed as an entity offering a selection of legal services: a lawyer agent. To apply for access to
 266 the resource, an agent must present its credentials and requirements to the lawyer.

267
 268
 269

The lawyer agent applies policy constraints to the request, relating to its contractual requirements of the client agent.

270 The result is a 'quality of service' specification that constrains the set of actions that the client agent may perform on the
 271 lawyer resource.

272

273 **3.2 Social Grouping Use Case**

274 **3.2.1 Description**

275 There may be policy constraints on the permissible communication between agents based on external attributes of
 276 those agents.

277

278 In many situations agents with access to one set of resources are not permitted to communicate with agents that have
 279 access to other resources. For example, in a merchant bank, agents (typically human agents) who have access to the
 280 stock market - i.e., are able to buy and sell stocks and shares, are not permitted access to financial services such as
 281 loan arrangements. This is the so-called 'Chinese Wall' encountered in larger merchant banks and represents the
 282 conditions that legislation imposes on merchant banks to allow them to do business in multiple sectors.

283
 284
 285
 286

These policy constraints are therefore strongly connected to groups of agents rather than the ability of individual agents
 to access resources.

287 **3.2.2 Scenario**

288 The different groups of agents in a merchant bank are divided into disjoint domains. An agent is required to register with
289 a domain, either the stock domain or the mortgage domain (say), in order to communicate with agents in those
290 domains.

291
292 An agent enters a domain by registering with the domain manager of that domain. Once registered, the agent is
293 permitted to send and receive messages from agents in the same domain. In general, an agent may be permitted to be
294 a member of several domains; depending on the policy constraints of the various domain managers.

295
296 This policy is enforced by preventing agents in one domain from communicating with agents in another domain.

297
298 In addition to preventing communication, other restrictions may include *hiding* agent descriptions: a directory service
299 can hide information about agents to non-member agents.

300

301 **3.3 Obligation Use Case**

302 **3.3.1 Description**

303 Agents may enter into agreements that oblige them into a certain future behaviour. Obligation constraints cannot be
304 enforced *a priori*, however sanctions can be applied to agents that fail to meet their obligations.

305

306 There are many situations where an agent may be obliged to perform a task: for example, a clock agent will enter into
307 an agreement to send a message at specific intervals, a database update agent will agree to inform the requester that
308 an update has taken place within the database and a file printing agent will agree to print a file within some interval or at
309 an agreed time.

310

311 An important service that can support obligations is the reputation service (see *Section 4.3.4, Reputation Service*). Such
312 services provide a means for agents to 'complain' about other agents' failure to meet obligations and for agents to verify
313 the reliability of other agents before entering into agreements.

314

315 **3.3.2 Scenario**

316 A lawyer agent agrees the terms of legal contract with a client. After contractual negotiation between the lawyer and
317 client the terms are submitted to a recognised reputation service.

318

319 The client agent notices that an action required of the lawyer agent has not taken place and files a complaint with the
320 reputation service.

321

322 A subsequent query to the reputation service reveals that the lawyer agent failed to complete on a contractual
323 obligation, thereby potentially affecting future agreements clients.

324

325 **3.4 Compositional Use Case**

326 **3.4.1 Description**

327 An agent may require or be required to enter into several conjunctive policies. The relationship between the individual
328 policy expressions may vary in strength, from weak influence to a strong propositional binding. Compositions may be
329 changed dynamically (in agreement with a policy authority) through the addition, modification or removal of constraint
330 clauses.

331

332 Constraint clauses may be: directly conjunctive to those describing the policy expression, at an upper-level changing
333 the context of the policy, or at a sub-level modifying an individual constraint by adding a conditional factor.

334

335 3.4.2 Scenario

336 An agent that has an active contract with a lawyer agent may wish to augment the contract policies with respect to a
337 specific legal scenario.

338
339 This implies retention of the original contract policy with an extension for the additional requirements, resulting in a new,
340 composed policy expression.
341

342 3.5 Refrain Use Case**343 3.5.1 Description**

344 An agent may be required to subjectively refrain from a particular action or set of actions according to the policy
345 constraints governing its interactions with other agents.

346
347 This implies that no direct intervention is required on behalf of another agent or policy authority. Rather the agent knows
348 that it must refrain from an action, perhaps one requested by another agent, in accordance with its policy constraints.
349

350 3.5.2 Scenario

351 A client agent wishes to express to a Lawyer agent that legal action should always be taken autonomously in regard to
352 a specific case instance, with the exception that on the satisfaction of certain constraints the Lawyer should refrain from
353 action.

354
355 The lawyer agent may be authorized to proceed with legal action with the constraint that no contact is to be made with
356 agent X at any time. The lawyer agent will exert a refrain if such an instance arises.
357

358 3.6 Content Use Case**359 3.6.1 Description**

360 Agents often apply policy constraints to their interactions with other agents. Policy driven agents such as these may
361 publish public policies to guide interactions with other agents.

362
363 For example, an agent may choose to constrain the form of messages it receives from other agents, and publish those
364 policies in a way that is revealed to certain other agents. This may perhaps include requirements that messages are
365 signed or have specific content attached.
366

367 3.6.2 Scenario

368 A Lawyer agent may only interact with a client agent if the messages contain a form of payment.

369
370 The client agent must therefore ensure that, in addition to any of its own requirements, any messages it sends to the
371 Lawyer agent contain some form of payment.
372

373 A third party, such as a bank service, may be involved to provide the client agent with appropriate modification to its
374 messages thereby ensuring the Lawyer agent recognize the payment portion of the message content.
375

376 **3.7 System Configuration Use Case**

377 **3.7.1 Description**

378 In addition to individual agents entering into individual obligations, a group or system of agents (and services) may enter
379 into coordinated performance related obligations. For example, a group of agents may guarantee to provide high
380 availability for an explicit period.

381
382 Such obligations may not, in fact, be honoured by individual agents but by the agent system as a whole; and therefore
383 will typically require monitoring and maintenance services.
384

385 **3.7.2 Scenario**

386 A group of agents is required to offer continuous high availability, with automatic reconfiguration as necessary.

387
388 A monitoring agent is used to observe the health of this group of agents and exert control if necessary. For example, if it
389 observes that one or more agents are not performing as expected, it can compensate by adjusting the properties of the
390 offending agents or by launching additional agents to offset the performance deficit.

391
392 Such a group service may be governed by service level agreements established between the agents and the monitoring
393 agent.
394

395 **3.8 Cooperation use case**

396 **3.8.1 Description**

397 This is an agreement to agree between agents. For example, an agent may enter a non-antagonistic posture
398 agreement with other agents incorporating guarantees and obligations on future behavior. This amounts to a sharing of
399 goals between agents.

400 **3.8.2 Scenario**

401 In the Lawyer-client scenario, the client can pay a retainer to the Lawyer agent thereby creating a co-operational stance
402 between the two.

403
404 The client can then make requests without submitting further payment for the duration of the contract.

405
406 This may require use of a reputation service to which a client agent may submit a complaint if the Lawyer agent refuses
407 a request covered by the cooperation agreement.
408

409 **3.9 Delegation Use Case**

410 **3.9.1 Description**

411 An agreement where an agent delegates authority or obligation. For example, an agent may choose or be forced to
412 defer authority on a particular stance, to another agent or group of agents. In the case where an agent segments a
413 policy governed task and delegates it across a number of other agents, the policy should be transposed according to
414 the actions of each delegated task segment.

415
416 In terms of contractual obligations, an agent may delegate only if the authority governing the obligation is aware of and
417 accepts the action.
418

419 3.9.2 Scenario

420 The Lawyer agent may delegate a contractual obligation to a 'legal specialist' agent, perhaps operating within the same
421 legal entity.

422
423 The delegatee is then required to meet the contractual obligations (or agreed subset thereof) specified by the delegator
424 and agreed with the policy authority.

425
426 This requires the policy authority managing the contract, say a guard mechanism, to accept the delegation and make
427 appropriate changes to the contract terms.

428

429 3.10 Meta Order Use Case**430 3.10.1 Description**

431 A meta order policy governs the nature of other policies. For example, it may specify that all agreements between
432 agents must involve a 'consideration' on both sides. (In Anglo-Saxon law, it is not possible to have a contract without
433 something of value being exchanged between all participating parties.)

434

435 3.10.2 Scenario

436 In the Lawyer-client scenario; the exchange of information between the two parties may be governed by the mutual-
437 consideration meta order policy.

438
439 In such a case, the reputation service must ensure that the client agent receives information from the Lawyer agent
440 sufficient to represent any payment made.

441
442 Therefore, when a reputation service is asked to validate an agreement, it must verify that it contains an co-exchange of
443 appropriate value. It will refuse to validate non-conforming agreements.

444

445 3.11 High Order Use Case**446 3.11.1 Description**

447 A higher order constraint is parameterized by other constraints. This is a form of dependency amongst constraints;
448 however, it is different to normal conjunction (which is implied by policy inheritance for example), in that a higher-order
449 policy refers explicitly to a 'policy variable'.

450

451 3.11.2 Scenario

452 A contract specifies that in the event of a dispute, the conflict resolution procedure associated with the domain that a
453 particular agent is in should be used.

454

455 3.12 Trust Use Case**456 3.12.1 Description**

457 Multiple levels of security may govern the relationships between agents and establishing a level of trust constrains the
458 type of agreement relationships agents can enter into. A particular trust level, indicated by a label or directly by a set of
459 policies, defines the constraints applicable to a given relationship.

460

461 3.12.2 Scenario

462 A new agent registers with a domain manager in order to interact with other agents within the domain.

463

464 The manager determines an appropriate trust level to assign the new agent and thereby a set of policies governing its
465 interaction with other agents within the domain.

466

467

471 4 Architectural Elements Needed to Support Policies

468 The elements of the policies and domains framework are defined here. For each element, the semantics are described
 469 informally followed by the relationships between the element and others.
 470

471 4.1 Introduction

472

473 4.2 Policy Structures

474 4.2.1 Policy

475 A policy is a constraint or set of constraints on the behaviour of agents and services.
 476

477 We are concerned with policies that are both public, i.e., available for inspection by third parties (although with obvious
 478 caveats regarding access control) and machine readable, i.e., a software system should be able to interpret a policy
 479 statement and determine legal courses of action.
 480

481 Types of constraints are defined as:
 482

483 **Structural constraints** specify policies about agents, their states, relationships, and communications that should not
 484 be violated. For example, a purchasing agent that is on probation may not place more than three orders. Or, there may
 485 never be more than seven agents bidding for a given item. The requirement that all messages must be encoded in a
 486 particular manner is another example of a structural constraint.
 487

488 **Operational constraints** specify policies about agent behaviour that should not be violated. For example, an Order
 489 agent may not close a particular order unless it has been shipped and paid for. Or, an Order may only be cancelled in it
 490 has not yet been shipped. Interaction protocols are also an example of operation constraints.
 491

492 4.2.2 Policy Language

493 The language used to express policy statements and contracts. Semantically, a policy statement and contract are
 494 equivalent: they express an agreement between an agent and other agents and/or services that constrain the behaviour
 495 of both.
 496

497 We are assuming that policy languages are *declarative*. This fits with the overall FIPA methodology as well as providing
 498 a number of substantial benefits to policy developers and policy mechanism implementers.
 499

500 Logically, a policy statement takes the form of a conjunction of implications: when a condition holds then an action is
 501 permitted, prohibited or whatever. In fact, the consequence of a policy rule need not be limited to single actions: it may
 502 also denote an enabling condition which allows other policy rules to trigger.
 503

504 In addition to standard predicate logic, we envisage a policy language having built-in ontologies for the concepts of
 505 *action*, *permission*, and *obligation*. For example, SL can be straightforwardly extended to include permission and
 506 obligation in a manner similar to its model for action.
 507

508 4.2.3 Policy Library

509 A set of rules that form coherent collections of policy statements. A policy library may introduce higher-level policy
 510 concepts (for example, National Security Classification) to simplify the task of generating specific policy rules for agents
 511 and services.
 512

513 4.2.4 Interpretation Engine

514 An interpretation engine is a mechanism for interpreting a set of policy rules and a proposed action to determine if the
515 action is legal according to the policy rules. It is possible that for certain classes of policy languages an interpretation
516 engine could also determine that a particular action is *required* at a given situation. However, in general this is a hard
517 problem.

518

519 The interpretation mechanism uses:

520

521 **Inference rules** that state if a certain facts are true, a conclusion can be stated of inferred.

522

523 **Computation policies** that define how to derive results via algorithms. For example, the net price of a product can be
524 computed as follows: (product price * (1 + tax percentage / 100)). Or, the set of all Managing Salesperson agents can
525 be computed as the intersection of all Manager agents and Salesperson agents.

526

527 4.2.5 Distribution Mechanism

528 A distribution mechanism is a means for distributing policy rules from originating authorities to mechanisms that have
529 the ability and responsibility of applying policies.

530

531 4.2.6 Conversation Policy

532 A number of research groups are working to extend the concept of FIPA *interaction protocols* to accommodate recent
533 research in agent conversation policies.

534

535 Conversations are sequences of messages involving two or more agents intended to bring about a particular set of
536 (perhaps jointly held) goals. In contrast to early agent communication research, agent researchers now acknowledge
537 that agent communication is better modelled when conversations rather than isolated messages are taken as the
538 primary unit of analysis.

539

540 Conversation policies are declarative specifications that govern specific instances of communications between agents
541 using an agent communication language. Contrary to current transition net approaches to specifying FIPA interaction
542 protocols, recent research suggests conversation policies are best represented as sets of fine-grained constraints on
543 ACL usage. These constraints define the computational process models that are implemented in agents. The key
544 notions here are:

545

546 Conversation policies provide a level of analysis that abstracts from the actual propositional content, agent
547 communication language, and implementation of individual conversations.

548

549 Conversation policies help ensure reliable communication between agents whilst simplifying any inference required
550 in determining which communicative act or other action should be made in response to a message.

551

552 The abstract architecture specifies a set of abstract objects that allow for the explicit representation of "a conversation",
553 i.e. a related set of messages between interlocutors that are logically related by some interaction pattern. It is desirable
554 that this property be achieved by the minimum of overhead at the infrastructure or message level; in particular, it is
555 important that interoperability remain un-compromised. For example, a concrete implementation may deliver messages
556 to conversation-specific queues based on an interpretation of the message envelope. To achieve interoperability with
557 an agent that does not support explicit conversations (i.e. which does not allow individual messages to be automatically
558 associated with a particular higher-level interaction pattern), it is necessary to specify the way in which the message
559 envelope must be processed in order to preserve conversational semantics.

560

561 4.3 Enforcement mechanisms

562 The two classes of constraints, corresponding to prohibitions and obligations, require different kinds of enforcement
563 mechanisms. The former can be supported with policy domains and the latter with reputation services.

564

565 4.3.1 Guards

566 An active computational element that interprets high level policies and ensures their enforcement in a platform-specific
567 way. Permissions are necessarily enforced in a different fashion than obligations. Permissions are granted or not before
568 an action is taken; whereas one can only monitor an agent's performance on its obligations and apply necessary
569 remedies after the fact. (Include examples of exception handling here, e.g., rollback.)
570

571 4.3.2 Sanctions

572 Violations of policy can result in remedies being applied to the offending agent; e.g., restrictions on the future behavior
573 of an agent, price controls, reduction in access. An indirect consequence of policy violation can also be that other
574 agents choose not to communicate with an offending agent. The most extreme form of sanction could be loss of domain
575 membership and even termination.
576

577 4.3.3 Policy Exception

578 An event raised as a consequence of a policy violation.
579

580 4.3.4 Reputation Service

581 Is a service that allows agents and services to monitor the public performance of agents and services in terms of their
582 compliance to publicly entered-into policy agreements.
583

584 A reputation service takes the role of a trusted third party that agents and service providers may use to monitor
585 compliance with agreements. Reputation services are one of the few mechanisms that are able to enforce obligations;
586 since obligations cannot be prevented but only required.
587

588 A typical use of a reputation service is for all parties to an agreement to 'escrow' their agreement with the reputation
589 service. If one of the parties determines that another party has defaulted on an obligation it may lodge a complaint with
590 the reputation service.
591

592 In software systems the concept of a legal remedy may seem moot; however, simply recording instances of default and
593 offering that information to others querying the service may be a powerful deterrence mechanism. If an agent defaults
594 on an obligation, other agents and services may become more reluctant to offer it facilities if they are able to query a
595 reputation service.
596

597 4.3.5 Policy Domain

598 A set of agents to which a given set of policies apply. In certain cases it may be possible to use domain membership as
599 a shorthand for applying the policy constraint inference procedures. In other words, the inference that a particular
600 service request is consistent with the policies in force in a given context may be reduced to the tests that (1) the domain
601 policies are consistent with the agent platform and (2) that the agent is a member of the domain.
602

603 A major purpose of Policy Domains is to ensure consistency of policy across a set of agents potentially running on
604 different agent platforms and hosts. This can be accomplished as long as semantically equivalent monitoring and
605 enforcement mechanisms are available across those platforms and hosts. Under these conditions, it follows that a given
606 domain could extend across host boundaries and, conversely, multiple domains could exist concurrently on the same
607 host. With respect to platform independence, it should be possible for agents running on the same platform to be in
608 different domains (for example, a resident and a visiting mobile agent running on the same platform may belong to
609 different domains having more or less restrictive security privileges).
610

611 4.3.6 Domain Manager

612 An agent domain consists of a unique instance of a domain manager along with any agents that are registered to it. The
613 function of a domain manager is to serve as a single point of administration for policy management, i.e., configure, re-
614 configure, store, publish and enforce where possible the set of policies declared for that domain.

615 4.4 Domain Management

616 It is possible to define domains that explicitly require registration, as well as domains that require no registration, or
617 subordinate registration to other elements of the FIPA environment, such as physical agent platforms. In this
618 specification, we are only concerned with policy domains that are active, and have explicit notions of domain
619 membership (i.e., there is an explicit list of agents that are members of a given domain).
620

621 4.4.1 Directory Functions

622 Provides services for agents register directory-entries. Other agents can search the directory-entries to find agents with
623 which they wish to interact. In other words, it provides services for registration, lookup, discovery, authentication, etc.
624

625 4.4.2 Conflict Resolution

626 Resolves conflicts that may arise between any combination of: agent, domain, host, and computational environment.
627

628 4.4.3 Policy Derivation

629 (rule generation)
630

631 4.4.4 Policy Change Notification

632 (broadcast/sub-domain broadcast).
633

634 4.4.5 Querying of Domain Policies

635

636

636 **5 Elements for Testing Conformance**

637

637 **6 Communicative Acts**

638 **6.1 Promise**

Summary	The action of agreeing to a contract; equivalent to signing the contract. The normal effect of promising is that the agent is bound by the terms of the contract.
Message Content	A proposition outlining the terms of the contract. Logically, a contract is a proposition, however it is normally in the form of conditional rules: if A is true then Y is true; where the various conditions of the contract and their consequences are outlined.
Description	<p>Promise is a general purpose agreement that an agent will abide by the terms of a contract. Normally, the promise does not directly imply that the agent will perform some action(s); but that if called upon under the right circumstances will agree to perform the actions.</p> <p>The agent sending the promise informs the receiver that it does intend to abide by the terms of the contract.</p>
Pragmatic Note	<p>The form of the proposition should be of the form of a conjunction of conditional rules. Each rule may be of the form:</p> <p style="text-align: center;"><i>if Condition then feasible (A, Action)</i></p> <p>where A is one of the agents party to the contract. Essentially, a promise is a promise to agree to act on future requests - provided that the preconditions of the rules apply.</p>
Formal Model	<pre><i, promise (j, Phi,)> <i, inform (j, Ii Phi,)> FP: RE: Bj Ii Phi</pre>
Examples	Fix me

639
640
641

641 **7 References**

642 [FIPA00001] FIPA Abstract Architecture Specification. Foundation for Intelligent Physical Agents, 2000.
643 <http://www.fipa.org/specs/fipa00001/>
644