FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA 98 Specification

Part 1

Agent Management

Obsolete

Publication date: 23rd October 1998

Geneva, Switzerland

This is one part of the first version of the FIPA 98 Specification as released in October 1998.
The latest version of this document may be found on the FIPA web site:

http://www.fipa.org

Comments and questions regarding this document and the specifications therein should be addressed to:
fipa98@fipa.org

It is planned to introduce a web-based mechanism for submitting comments to the specifications.
Please refer to the web site for FIPA's latest policy and procedure for dealing with issues regarding the specification.

Contents

Foreword

The Foundation for Intelligent Physical Agents (FIPA) is a non-profit association registered in Geneva, Switzerland. FIPA's purpose is to promote the success of emerging agent-based applications, services and equipment. This goal is pursued by making available in a timely manner, internationally agreed specifications that maximise interoperability across agent-based applications, services and equipment. This is realised through the open international collaboration of member organisations, which are companies and universities active in the agent field. FIPA intends to make the results of its activities available to all interested parties and to contribute the results of its activities to appropriate formal standards bodies.

This specification has been developed through direct involvement of the FIPA membership. The 48 members of FIPA (October 1998) represent 13 countries world-wide.

Membership in FIPA is open to any corporation and individual firm, partnership, governmental body or international organisation without restriction. By joining FIPA each member declares himself individually and collectively committed to open competition in the development of agent-based applications, services and equipment. Associate Member status is usually chosen by those entities who want to be members of FIPA without using the right to influence the precise content of the specifications through voting.
The members are not restricted in any way from designing, developing, marketing and/or procuring agent-based applications, services and equipment. Members are not bound to implement or use specific agent-based standards, recommendations and FIPA specifications by virtue of their participation in FIPA.

This specification is published as FIPA 98 specifications ver 1.0. All these parts have undergone an intense review by members as well as non-members during the past year as preliminary versions have been available on the FIPA web site. FIPA members as well as many non-members have been conducting validation trials of the FIPA 97 specification during 1998 and will continue to subject the new output to further validation during the coming months. During 1999 FIPA will publish revised versions of the current specifications and is also planning to continue work on further specifications of agent based technology.

Introduction

The FIPA specifications represent the primary output of FIPA. It is important to appreciate that these specifications have been derived from examining requirements on agent technology posed by specific industrial applications chosen by FIPA so far, and described in Parts 4, 5, 6, and 7 of the FIPA 97 specifications.
FIPA specifies the interfaces of the different components in the environment with which an agent can interact, i.e. humans, other agents, non-agent software and the physical world. FIPA produces two kinds of specifications:

normative specifications mandating the external behavior of an agent and ensuring interoperability with other FIPA-specified subsystems;

informative specifications of applications providing guidance to industry on the use of FIPA technologies.

In October 1997, FIPA released its first set of specifications, called FIPA 97, Version 1.0. During 1998, comments on this specification were received. Based upon these comments, parts of FIPA 97 were superseded by a second version released in October 1998, introducing minor changes only.
Furthermore, in October 1998 FIPA released a new set of specifications, called FIPA 98, version 1.0, of which this document is a part.

The following tables provide an overview of the complete set of FIPA specifications.

Sorted by part:

		Released October 1997	***Released October 1998***	
Part		**FIPA 97 Version 1.0**	**FIPA 97 Version 2.0**	**FIPA 98 Version 1.0**
1	N	*Agent Management*	Agent Management	Agent Management Extensions
2	N	*ACL*	ACL	
3	N	Agent Software Integration		
4	I	Personal Travel Assistant		
5	I	Personal Assistant		
6	I	Audio Visual Entertainment & Broadcasting		
7	I	Network Management & Provision		
8	N			Human-Agent Interaction
10	N			Agent Security Management
11	N			Agent Management Support for Mobility
12	N			Ontology Service
13	I/M			Developer's Guide

N == normative; I == informative; M == methodology; *Italicised* == *superseded*

Sorted by topic:

Topic	**FIPA 97***(Version 1.0, unless otherwise indicated)*	**FIPA 98** Version 1,0
Agent Management	1. Basic System *(Version 2.0)*	1. Extension to Basic System 10. Agent Security Management 11. Agent Management Support for Mobility
Agent Communication	2. Agent Communication Language *(Version 2.0)*	8. Human-Agent Interaction 12. Ontology Service
Agent S/W Integration	3. Agent Software Integration	
Reference Applications	4. Personal Travel Assistant 5. Personal Assistant 6. Audio/Visual Entertainment & Broadcasting 7. Network Management & Provisioning	

The parts of the FIPA 98 specifications are briefly described below.

Part 1 - Agent Management

This part covers agent management for inter-operable agents, and is thus primarily concerned with defining open standard interfaces for accessing agent management services. It also specifies an agent management ontology and agent platform message transport. This specification incorporates and further enhances the FIPA 97, Part 1, Version 2.0 specification. The internal design and implementation of intelligent agents and agent management infrastructure is not mandated by FIPA and is outside the scope of this part.

Part 8 – Human-Agent Interaction

This part deals with the human-agent interaction part of an agent system. It specifies two agent services: User Dialog Management Service (UDMS) and User Personalization Service (UPS). A UDMS wraps many types of software components for user interfaces allowing for ACL level of interaction between agents and human users. A UPS can maintain user models and supports their construction by either accepting explicit information about the user or by learning from observations of user behavior.

Part 10 – Agent Security Management

Security risks exist throughout agent management: during registration, agent-agent interaction, agent configuration, agent-agent platform interaction, user-agent interaction and agent mobility. The Security Management specification identifies the key security threats in agent management and specifies facilities for securing agent-agent communication via the FIPA agent platform. This specification represents the minimal set of technologies required and is complementary to the existing FIPA 97 and FIPA 98, Part 1 specifications. This part does not mandate every FIPA-compliant agent platform to support agent security management.

Part 11 – Agent Management Support for Mobility

This specification represents a normative framework for supporting software agent mobility using the FIPA agent platform. This framework represents the minimal set of technologies required and is complementary to the existing FIPA 97 and FIPA 98, Part 1 specifications. Wherever possible, it refers to existing standards in this area. The framework supports additional non-mobile agent management operations such as agent configuration. The specification does not mandate that every FIPA-compliant agent platform must support agent mobility, nor does it cover the specific requirements for agents on mobile devices with intermittent connectivity, which is covered by the scope of

the existing FIPA Agent Management activity.

Part 12 – Ontology Service

This part deals with technologies enabling agents to manage explicit, declaratively represented ontologies. It specifies an ontology service provided to a community of agents by a dedicated Ontology Agent. It allows for discovering public ontologies in order to access and maintain them; translating expressions between different ontologies and/or different content languages; responding to queries for relationships between terms or between ontologies; and, facilitating identification of a shared ontology for communication between two agents.

The specification deals only with the communicative interface to such a service while internal implementation and capabilities are left to developers. The interaction protocols, communicative acts and, in general, the vocabulary that agents must adopt when using this service are defined. The specification does not mandate the storage format of ontologies, but only the way the ontology service is accessed. However, in order to specify the service, an explicit representation formalism, or meta-ontology, has been specified allowing communication of knowledge between agents.

Part 13 – FIPA 97 Developer's Guide

The Developer’s Guide is meant to be a companion document to the FIPA 97 specifications, and is intended to clarify areas of specific interest and potential confusion. Such areas include issues that span more than one of the normative parts of FIPA 97.

1 Scope

This document is part of the FIPA 1998 specifications covering agent management for inter-operable agents. This specification incorporates and further enhances the FIPA97 part 1 version 2 specification.

The Security Management (FIPA98 part 10) and the Agent Management Facilities for Mobility (FIPA98 part 11) specifications represent companion specifications.

This document contains specifications for agent management including: agent management services, agent management ontology, agent platform message transport.

This document is primarily concerned with defining open standard interfaces for accessing agent management services. The internal design and implementation of intelligent agents and agent management infrastructure is not mandated by FIPA and is outside the scope of this specification.

The document provides a series of examples to illustrate the agent management actions defined.

2 Normative reference(s)

Object Management Group : Common Object Request Broker Architecture (Version 2)
Internet Inter-ORB Protocol (IIOP) : Common Object Request Broker Architecture (Version 2)
FIPA – International standard for the inter-operation of software agents – Part 1: Agent Management (V.2.0).
FIPA – International standard for the inter-operation of software agents – Part 2: Agent Communication Language (V.2.0).
FIPA – International standard for the inter-operation of software agents – Part 3: Agent/Software Integration (V.2.0).
FIPA – International standard for the inter-operation of software agents – Part 10: Agent Management Support for Mobility (V.1.0).
FIPA – International standard for the inter-operation of software agents – Part 11: Agent Security Management (V.1.0).

3 Terms and definitions

For the purposes of this specification, the following terms and definitions apply:

Action
A basic construct which represents some activity which an agent may perform. A special class of actions is the communicative acts.

Agent
An Agent is the fundamental actor in a domain. It combines one or more service capabilities into a unified and integrated execution model which can include access to external software, human users and communication facilities.

Agent cloning
The process by which an agent creates a copy of itself on an agent platform.

Agent code
The set of instructions used by an agent.

Agent Communication Language (ACL)
A language with precisely defined syntax, semantics and pragmatics that is the basis of communication between independently designed and developed software agents. ACL is the primary subject of the FIPA 97 specification, part 2.

Agent Communication Channel (ACC)
The Agent Communication Channel is an agent which uses information provided by the Agent Management System to route messages between agents within the platform and to agents resident on other platforms.

Agent data
Any data associated with an agent.

Agent invocation
The process by which an agent can create another instance of an agent on an agent platform.

Agent Management System (AMS)
The Agent Management System is an agent which manages the creation, deletion, suspension, resumption, authentication and migration of agents on the agent platform and provides a “white pages” directory service for all agents resident on an agent platform. It stores the mapping between globally unique agent names (or GUID) and local transport addresses used by the platform.

Agent Platform
An Agent Platform provides an infrastructure in which agents can be deployed. An agent must be registered on a platform in order to interact with other agents on that platform or indeed other platforms. An AP consists of three capability sets ACC, AMS and default Directory Facilitator.

Agent Platform Security Manager (APSM)
An Agent Platform Security Manager is responsible for maintaining the agent platform security policy. The APSM is responsible for providing transport-level security and creating agent audit logs. The APSM negotiates the requested intra- and inter-domain security services of other APSM's in concert with the implemented distributed computing architectures, such as CORBA, COM, DCE, on behalf of an agent in its domain.

ARB Agent
An agent which provides the Agent Resource Broker (ARB) service. There must be at least one such an agent in each Agent Platform in order to allow the sharing of non-agent services.

Communicative Act
A special class of actions that correspond to the basic building blocks of dialogue between agents. A communicative act has a well-defined, declarative meaning independent of the content of any given act. CAs are modelled on speech act theory. Pragmatically, CAs are performed by an agent sending a message to another agent, using the message format described in FIPA97, part 2.

Content
That part of a communicative act which represents the domain dependent component of the communication. Note that "the content of a message" does not refer to "everything within the message, including the delimiters", as it does in some languages, but rather specifically to the domain specific component. In the ACL semantic model, a content expression may be composed from propositions, actions or IRE's.

Content Language
The *content* of a FIPA message refers to whatever the communicative act applies to. If, in general terms, the communicative act is considered as a sentence, the content is the grammatical object of the sentence. This content can be encoded in any language, the *content language*, denoted by the `:language` parameter of the communicative act.

Conversation
An ongoing sequence of communicative acts exchanged between two (or more) agents relating to some ongoing topic of discourse. A conversation may (perhaps implicitly) accumulate context that is used to determine the meaning of later messages in the conversation.

CORBA
Common Object Request Broker Architecture, an established standard allowing object-oriented distributed systems to communicate through the remote invocation of object methods.

Directory Facilitator
The Directory Facilitator [1] is an agent that provides a “yellow pages” directory service for the agents. It stores descriptions of the agents and the services they offer.

Explicit & Implicit
An ontology is *explicit* when it is specified in declarative form as a set of axioms and definitions (e.g. as a set of Ontolingua statements) that an agent can refer to (e.g. by means of an OKBC interface). An ontology is *implicit*, when the assumptions on the meaning of its vocabulary are only implicitly embedded in some piece of software.

Feasibility Precondition (FP)
The conditions (i.e. one or more propositions) which need be true before an agent can (plan to) execute an action.

Knowledge model
It is a specification of the set of primitives used by a certain class of representation languages. As such, a knowledge model can be considered as a meta-ontology. For instance, several ontology servers use an object oriented model of knowledge based on primitive notions like classes, frames, properties, constraints, axioms and functions. FIPA adopts for the specification of these notions the OKBC version 2.0.4 Knowledge Model, which is called FIPA-meta-ontology or FIPA knowledge model.

Illocutionary effect
See speech act theory.

Knowledge Querying and Manipulation Language (KQML)
A de facto (but widely used) specification of a language for inter-agent communication. In practice, several implementations and variations exist.

Local Agent Platform
The Local Agent Platform is the AP to which an agent is attached and which represents an ultimate destination for messages directed to that agent.

Message
An individual unit of communication between two or more agents. A message corresponds to a communicative act, in the sense that a message encodes the communicative act for reliable transmission between agents. Note that communicative acts can be recursively composed, so while the outermost act is directly encoded by the message, taken as a whole a given message may represent multiple individual communicative acts.

Message content
See content.

Message transport service
The message transport service is an abstract service provided by the agent management platform to which the agent is (currently) attached. The message transport service provides for the reliable and timely delivery of messages to their destination agents, and also provides a mapping from agent logical names to physical transport addresses.

Meta-ontology
For allowing a FIPA agent to communicate through ACL messages about ontologies, it is necessary to describe the concepts used to speak about an ontology. This description is called the meta-ontology. It is an ontology itself as it provides the ontology to refer to another ontology. Therefore, the meta-ontology should be powerful enough to deal with all potentially available ontologies and make explicit, at least informally, these concepts.

Mobile agent
An agent that is not reliant upon the agent platform where it began executing and can subsequently transport itself between agent platforms.

Mobility
The property or characteristic of an agent that allows it to travel between agent platforms.

Ontology
An ontology is an explicit specification of the structure of a certain domain (e.g. e-commerce, sport, …). For the practical goals of FIPA (that is enabling development and deployment of inter-operable agent-based applications), this includes a vocabulary (i.e. a list of logical constants and predicate symbols) for referring to the subject area, and a set of logical statements expressing the constraints existing in the domain and restricting the interpretation of the vocabulary. Ontologies therefore provide a vocabulary for representing and communicating knowledge about some topic and a set of relationships and properties that hold for the entities denoted by that vocabulary.

Ontology Agent
An agent that provides the Ontology Service specified in this specification. The main objective of the Ontology Agent is to offer to FIPA agents a unified view of the services offered by the different ontology servers. Its second objective is to allow an ontology server to be known by FIPA agents. Moreover some ontology agents can provide the agents with services such as translation facilities. Like any other FIPA agent, the ontology agent has to be registered to the DF and to provide the DF with the published ontologies and available services.

Ontology Name
The ontologies referred to by the agents can be provided by different ontology servers. Consequently, these ontology names are constructed from: the OA name, and the ontology logical name (given by the ontology designer e.g. “car“).

Ontology Server
Provider of an Ontology Service, not necessarily in the FIPA domain, or FIPA-compliant. Examples of ontology servers already existing outside FIPA are: Ontolingua, XML/RDF ontology servers, ODL databases ontologies servers. Access

to the services provided by these ontologies servers are based on various APIs such as the OKBC interface, the ODL interface or HTTP.

Ontology sharing problem
The problem of ensuring that two agents that wish to converse do, in fact, share a common ontology for the domain of discourse. Minimally, agents should be able to discover whether or not they share a mutual understanding of the domain constants.

Perlocutionary Effect
See speech act theory.

Personalization
An agent's ability to take individual preferences and characteristics of users into account and adapt its behavior to these factors.

Proposition
A statement which can be either true or false. A closed proposition is one which contains no variables, other than those defined within the scope of a quantifier.

Protocol
A common pattern of conversations used to perform some generally useful task. The protocol is often used to facilitate a simplification of the computational machinery needed to support a given dialogue task between two agents. Throughout this document, we reserve protocol to refer to dialogue patterns between agents, and networking protocol to refer to underlying transport mechanisms such as TCP/IP.

Rational Effect (RE)
The rational effect of an action is a representation of the effect that an agent can expect to occur as a result of the action being performed. In particular, the rational effect of a communicative act is the perlocutionary effect an agent can expect the CA to have on a recipient agent. Note that the recipient is not bound to ensure that the expected effect comes about; indeed it may be impossible for it to do so. Thus an agent may use its knowledge of the rational effect in order to plan an action, but it is not entitled to believe that the rational effect necessarily holds having performed the act.

Software Service
An instantiation of a connection to a software system.

Software System
A software entity which is not conformant to the FIPA Agent Management specification.

Speech Act
The notion of a speech act is derived from the linguistic analysis of human communication. It is based on the idea that with language the speaker not only makes statements, but also performs actions, e.g. a request or an assertion. In this context, a verb denoting a speech act, is called a *performative*, since saying it makes it so. See FIPA97, part 2 for more details.

Speech Act Theory
A theory of communications which is used as the basis for ACL. Speech act theory is derived from the linguistic analysis of human communication. It is based on the idea that with language the speaker not only makes statements, but also performs actions. A speech act can be put in a stylised form that begins "I hereby request …" or "I hereby declare …". In this form the verb is called the performative, since saying it makes it so. Verbs that cannot be put into this form are not speech acts, for example "I hereby solve this equation" does not actually solve the equation.

Stationary agent
An agent that executes only upon the agent platform where it begins executing and is reliant upon it.

TCP/IP
An internet networking protocol used to establish connections and transmit data between hosts

User Agent
An agent which interacts with a human user.

User Dialog Management Service
An agent service in order for FIPA agents to interact with human users; by converting ACL into media/formats which human users can understand and vice versa, managing the communication channel between agents and users, and identifying users interacting with agents.

User ID
An identifier for a real user.

User Model
A user model contains assumptions about user preferences, capabilities, skills, knowledge, etc, which may be acquired by inductive processing based on observations about the user. User models normally contain knowledge bases which are directly manipulated and administered.

User Personalization Service
An agent service that offers abilities to support personalization, e.g. by maintaining user profiles or forming complex user models by learning from observations of user behavior.

Wrapper Agent
An agent which provides the FIPA-WRAPPER service to an agent domain on the Internet.

4 Symbols (and abbreviated terms)

ACC:	Agent Communication Channel
ACL:	Agent Communication Language
AMS:	Agent Management System
AP:	Agent Platform
API:	Application Programming Interface
APSM:	Agent Platform Security Manager
ARB:	Agent Resource Broker
CA:	Communicative Act
CORBA:	Common Object Request Broker Architecture
DB:	Database
DCOM:	Distributed COM
DF:	Directory Facilitator
FIPA:	Foundation for Intelligent Physical Agents
FP:	Feasibility Precondition
GUID:	Global Unique Identifier
HAP:	Home Agent Platform
HTTP:	Hypertext Transmission Protocol
IDL:	Interface Definition Language
IIOP:	Internet Inter-ORB Protocol
IPMT:	Internal Platform Message Transport
IRE:	Identifying Referring Expression
OMG:	Object Management Group
ORB:	Object Request Broker
P3P:	Platform for Privacy Preferences Project
PICS:	Platform for Internet Content Selection
RE:	Rational Effect
RMI:	Remote Method Invocation, an inter-process communication method embodied in Java
SL:	Semantic Language
SMTP:	Simple Mail Transfer Protocol
SQL:	Structured Query Language
S/W:	Software System
TCP / IP:	Transmission Control Protocol / Internet Protocol
UDMA:	User Dialogue Management Agent
UDMS:	User Dialogue Management Service
UPA:	User Personalization Agent
UPS:	User Personalization Service
XML:	eXtensible Markup Language

5 Overview

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 are logical capability sets (i.e. services) and do not imply any physical configuration. Additionally, the implementation details of individual platforms and agents are the design choices of the individual agent system developers.

Figure 1 is a graphical representation of the agent management reference model.

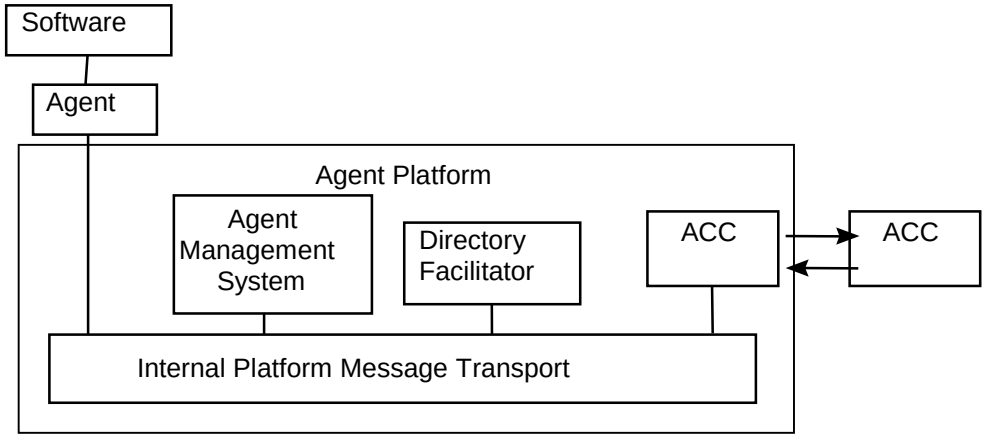

Figure 1 : Agent Management Reference Model

The agent management reference model consists of the following logical components each representing a capability set. These can be combined in physical implementations of agent platforms.

An **Agent** is the fundamental actor on an agent platform which combines one or more service capabilities into a unified and integrated execution model which may include access to external software, human users and communications facilities. An Agent may have certain resource brokering capabilities for accessing software, (see FIPA 97 Part 3 Agent-Software Interaction).

An Agent must have one or more owners, (for example, based on organisational affiliation or human user). An Agent supports several notions of identity. A Globally Unique Identifier (GUID), also known as agent-name, labels an agent over all FIPA domains so that it may be distinguished unambiguously in the agent universe. An agent may be registered at a number of addresses at which it can be contacted. An Agent may have certain resource brokering capabilities for accessing software, (see FIPA 97 Part 3 Agent-Software Interaction).

Directory Facilitator (DF) : The DF provides “yellow pages” services to other agents. The DF is a mandatory, normative agent. Agents may register their services with the DF or query the DF to find out what services are offered by other agents.

Agent Management System (AMS): An AMS is a mandatory component of the AP. It is an agent which 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 logical agent names and their associated transport addresses for an agent platform. The AMS offers “white pages” services to other agents.

Agent Communication Channel (ACC) : All agents have access to at least one ACC. The ACC is the default communication method between agents on different AP's. The message routing service offered by the ACC must be reliable and orderly and will adhere to the requirements specified in section 9, FIPA Baseline Protocol and ACC. (See also Annex B and preferred requirements for ACC and baseline protocol)

An **Agent Platform (AP)** provides the physical infrastructure in which agents can be deployed. The AP consists of the machine(s), operating system, agent support software, FIPA agent management components (DF, AMS, ACC), Internal Platform Message Transport and agents. The Internal Platform Message Transport is outside the scope of FIPA.

The internal design of an AP is an issue for AP developers and is not a subject of standardisation within FIPA. AP's and the agents which are native to those APs, either by creation directly within or migration to the AP may use any proprietary method of intercommunication. For example, an AP could be implemented in Java and message-passing could be equivalent to function calls.

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

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

Internal Platform Message Transport (IPMT) is the proprietary means of exchanging messages within an AP and is outside the scope of FIPA.

An **Agent Domain** is a logical grouping of agents defined by membership of a directory maintained by the DF. Each domain has one and only one DF, which provides a unified, complete and coherent description of the domain. The directory lists all Agents in the DF domain and is used to advertise agent existence, services etc. An agent may be present in one or more domains. As part of its normative life-cycle, an agent must register with a DF in order to be present in a domain. For an agent to exist in the context of this model, it must be registered in at least one domain. Domains may have (for example) organisational, geo-political, contractual, ontological affiliation or physical significance. An AP can support more than one domain.

The entire **Agent Universe** is represented as the set of all domains.

FIPA places minimal restrictions on whatever default intra-AP message routing protocol individual agent-developers wish to support. The minimum baseline protocol a FIPA compliant agent platform will support is the Internet Inter-Orb Protocol (IIOP) from the Object Management Group (OMG). The use of IIOP does not preclude an AP from augmenting this inter-platform messaging protocol with other interoperability protocols, however IIOP must be supported for an AP to be FIPA compliant.

Non-agent software is defined as all non-agent, executable collections of instructions accessible through an agent. Agents may access software (for example) to: add new services, acquire new communications protocols, acquire new security protocols/algorithms, acquire new negotiation protocols, access tools which support migration, etc.

6 Agent Management Services

6.1 Directory Facilitator (DF)

Overview

The DF provides a "yellow-pages" directory service to agents. The DF is a mandatory, normative agent which is the trusted, benign custodian of an agent directory. It is trusted in the sense that it must strive to maintain an accurate, complete and timely list of agents. It is benign in the sense that it must provide the most current information about agents in its directory on a non-discriminatory basis to all authorised agents. At least one DF must be resident on each AP (the default DF). However an AP may support any number of DFs.

The DF may restrict access to information in its directory, and will verify all access permissions for agents which attempt to inform it of Agent state changes. The DF does not control the AP life-cycle of any Agent.

Agents may register their services with the DF or query the DF to find out what services are offered by which agents.

DFs can register with each other. Similarly, the AMS, and ACC can register with a DF.

The DF encompasses a search mechanism which searches first locally, then, if necessary, extends the search to other DFs. The default search mechanism is assumed to be a depth first search. For specific purposes, the following optional constraints can be used : the number of answers `:df-search-resp-req ,` the number of hops `:df-search-depth,` a time-out `:df-search-time-out` and the search protocol `:df-search-algo.`

All DFs have a default name which is `df` appended onto the remainder of a FIPA agent name (see section 8.1 Agent Naming and Addressing), for example:

```
df@iiop://fipa.org:50/acc
```

or

```
df@nmf.org:40/acc
```

Management actions supported by the DF

`register`
`search`
`deregister`
`modify`

6.2 Agent Management System (AMS)

Overview

An AMS is a mandatory component of the AP. Only one AMS will exist in a single AP. The AMS is responsible for managing the operation of an AP. These responsibilities include creation of agents, deletion of agents, deciding whether an agent can dynamically register with the AP (for example, this could be based upon agent ownership) and overseeing the migration of agents to and from the AP. Since different APs have different capabilities, the AMS can be queried to obtain a profile of its AP. A life-cycle is associated with each agent on the AP (see Section 7.1).

The AMS represents the managing authority of an AP. If the AP has multiple machines the AMS represents the authority across all machines. An AMS can request an agent to `quit` (i.e. terminate all execution on its AP). The AMS has authority to forcibly terminate an agent if such a request is ignored.

The AMS maintains an index of all the agents which are currently resident on an AP. The index includes an agent's GUID and their associated transport address for the AP. Residency of an agent on the AP implies that the agent has been registered with the AMS. Access to the `ams-agent-description` in the index is controlled by the AMS.

All AMS have a default name which is `ams` appended onto the remainder of a FIPA agent name (see section 8.1 Agent Naming and Addressing), for example:

```
ams@iiop://fipa.org:50/acc
```

or

```
ams@nmf.org:40/acc
```

Management actions supported by the AMS

Mandatory management actions :

`register-agent`
`deregister-agent`
`modify-agent`
`query-platform-profile`
`authenticate`
`search-agent`

Additional mandatory management action where mobility is supported (see FIPA98 Part 11 v.1.0) :

`move`

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

- suspend agent
- terminate agent
- create agent
- resume agent execution
- invoke agent
- execute agent
- resource management

Management actions supported by the other agents used by the AMS

`quit`

6.3 Agent Communication Channel (ACC)

Overview

The ACC routes messages between agents within an AP to agents resident on other APs. All FIPA agents have access to at least one ACC. Only messages addressed to an agent can be sent to an ACC.

The ACC provides for the routing of messages between agents on different APs. Routing messages between AP's requires agreement on a default interoperability protocol including transport protocol, encoding and addressing scheme. However, if an agent dynamically registers with an AP, then there is always a method available for exchanging messages between that agent and the agents that already reside on the AP.

The ACC is an agent for meta-level control of communication. The ACC has a management interface to the IPMT mechanism which FIPA does not define. The forward action on the ACC should not be understood as the default sending mechanism for agents resident on the same AP.

Management actions supported by the ACC

forward

7. The Agent Platform

7.1 The AP Life-Cycle

FIPA agents exist physically on an AP and utilise the facilities offered by the AP for realising agent functionalities. In this context, an agent, as a physical software process, has a physical life-cycle that has to be managed by the AP. For each agent, this physical life-cycle and the associated states can be different from the external logical life-cycle and states in the domain. The latter are managed by the DF. It should be noted that the implementation of a FIPA conformant AP does not necessitate the use of all the states.

The AP life-cycle of a FIPA agent is :

1) AP bounded : An agent is physically managed within an AP. The life-cycle of a static agent is therefore always bounded to a specific AP.

2) Application independent : The life-cycle model is independent from any application system. It defines only the states and the transitions of the agent service in its life cycle.

3) Instance oriented : The agent described in the life-cycle model is assumed an instance (an agent which has unique name and is executed independently).

4) Uniqueness : Each agent has only one AP life-cycle state at any time and within only one AP.

The agent AP life-cycle is represented by states (circles) and transitions as showed in the figure below.

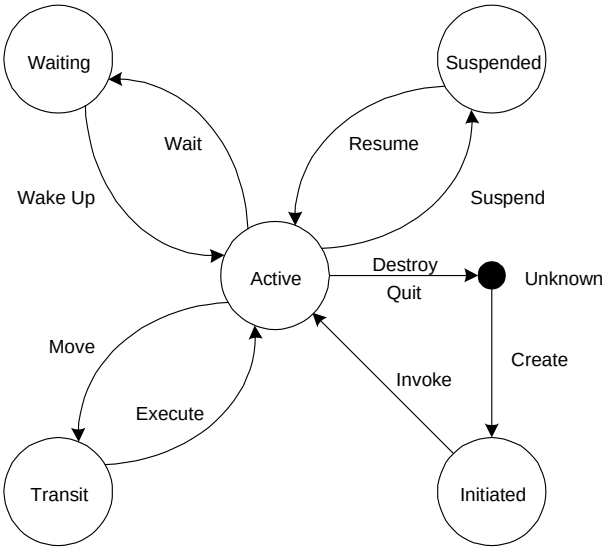

Figure 2: Agent lifecycle

Only mobile agents can enter the transit state. This ensures that a stationary agent executes all of its instructions on the node where it was invoked. The actions of agents can be described as (figure 2):

Create.	The creation (installation) of a new agent.
Invoke.	The invocation of a new agent.
Destroy.	The forceful termination of an agent. This can only be initiated by the AMS and cannot be ignored by the agent.
Quit.	The graceful termination of an agent. This can be ignored by the agent.
Suspend.	Puts an agent in a suspended state. This can be initiated by the agent or the AMS.
Resume.	Brings the agent from a suspended state. This can only be initiated by the AMS.
Wait.	Puts the agent in a waiting state. This can only be initiated by the agent.
Wake Up.	Brings the agent from a waiting state. This can only be initiated by the AMS.

The following two actions are only used by mobile agents (see Part 11 FIPA98).

Move.	Puts the agent in a transitory state. This can only be initiated by the agent.
Execute.	Brings the agent from a transitory state. This can only be initiated by the AMS.

7.2 The Home Agent Platform

The Home Agent Platform (HAP) is the AP on which an agent was created and is responsible for vouching for the agent's identity in its dealings with other agents and APs. This standard requires that every agent has an HAP which vouches for the agent to the rest of the agent community. To enforce this, FIPA requires that the GUID can be analysed to obtain the IIOP-URL of the HAP. FIPA requires that the HAP can authenticate the identity of the agent on that AP. To accomplish this the AMS of the HAP supports the following query:

```
(request
            :sender     ams1-agent@iiop://fipa.org:50/acc
            :receiver   ams2-agent@iiop://agentland.com:90/acc
            :content
                  (action ams2-agent@iiop:// agentland.com:90/acc
                    (authenticate
                      (:ams-description
                         (:agent-name ag@iiop://agentland.com:90/acc)))
                      ...))
```

The AMS on the agent's HAP is responsible for recording an agent's current valid address. It is the agent's responsibility to ensure that the address held by its HAP AMS is valid. An agent must always remain registered with its HAP. An agent will have its name for its entire lifetime.

7.3 Agent Registration on an AP

There are only three ways in which an agent can be registered in the AMS:

1) The agent was created on the AP.

2) The agent migrated to the AP, for those APs which support agent-mobility.

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

Agent registration involves registering the following two items of information with an AMS:

1) The globally unique agent identifier (GUID).

2) The local address of the agent.

When an agent is either created or dynamically registers with an AP, the agent is registered with the AMS for example using the *register-agent* action. In the following example an agent called Peter is registering dynamically with the FIPA AP (located at `fipa.org`). The agent *Peter* was created on the AP (i.e Peter's HAP) at `agentland.com`. and requests that the AMS registers it.

For example :

```
(request
            :sender peter@iiop://agentland.com:50/acc
            :receiver ams@iiop://fipa.org:50/acc
            :ontology   fipa-agent-management
            :language   SL0
            :protocol fipa-request
            :content
             (action ams@iiop://fipa.org:50/acc
                (register-agent
                  (:ams-description
                        (:agent-name peter@iiop://agentland.com:50/acc)
                        (:address    iiop://agentland.com:50/acc)
                              ...)))
```

It should be noted that the address which is supplied to the `register-agent` action is the address the agent would like messages directed to, in effect a forwarding address. This represents an agent's *local AP*, which is the one to which it is attached and represents an ultimate destination for messages directed to that agent. In this example, the agent registers with `fipa.org` and sets it's forwarding address to it's HAP, so any messages which arrive at `fipa.org` for Peter will be forwarded to `agentland.com`[1].

By default, the `forward-agent` parameter is set to the `agent-name`. If however, the agent chooses to change this parameter (using `modify-agent` action on the AMS), then messages will be re-directed to another agent.

[1] When an agent registers with the AMS, the AMS records its local AP which represents a forwarding address. This leads to the natural question of what address does Peter have at its HAP `agentland.com`. FIPA is only concerned with the interoperability between agents and APs. The internal design of an AP is a platform-developer issue and not the subject of standardisation. Since Peter was created on `agentland.com` the address registered with the AMS will only have local significance within the platform, for example, if `agentland.com` were implemented using Java then the address could be a Java Object Reference. Furthermore, it is assumed that platform developers will each specify their own method of enabling agents to contact the ACC.

8. Inter-AP Communication

An agent has two options when it wishes to contact an agent on another AP:

1) It can request that the ACC of the AP on which it currently resides routes the message to the target agent and ACC.

2) It can contact the ACC of the target AP directly - i.e. cause a message to be sent directly to the target ACC. The target ACC is then responsible for routing the message to the agent on the target AP.

To contact another agent, the sender agent must be equipped with the agent name (i.e. GUID) of the receiver agent. In this case the message will be directed to the receiver agent's HAP for delivery to the receiver agent. Alternatively, if the sender wishes to route the message directly to the agent, or to an AP on which the receiver agent has dynamically registered, then the sender can specify an address in the destination field of the envelope in addition to the agent-name in the receiver field of the message.

8.1 Agent Naming and Addressing[2]

All agents have a unique identifier also known as its GUID. An agent name is a concatenation of its HAP communication address and a unique name within that AP.

`<name>@<hostname>:<port>/<target>`[3]

1) where `name` is a unique expression for an agent within the HAP. For example, FipaAgent

2) where `hostname` is the IP address of the host on which an ACC is running or a Domain Name Service (DNS) entry which can be further resolved to an IP address

3) where `port` is the port number of that host on which the ACC is listening; and

4) where `target` is the object key which is used to identify the receiver of the message which the ACC should dispatch the incoming message to. By default, the object key of IIOP messages exchanged between APs will identify the ACC of that AP.

8.2 Agent Messaging

FIPA requires that each AP provide an ACC which will route messages on an agent's behalf where possible. To support this, FIPA requires that each ACC support a baseline protocol as a default method of communication. This does not mean that each agent must also support that protocol. The address an agent provides, for example on registration with the AMS, will determine how a message is routed to that agent. If the address given is the address of an AP (e.g. `iiop://agentland.com/acc`), then the message will be routed to that AP and it is then the responsiblity of the ACC of that AP to route the message to the agent (in an AP-specific manner).

[2] Agent naming is a topic planned for further discussion in FIPA99.

[3] The target address is optional depending on the internal architecture of the AP, for example, direct IIOP may be used.

The payload of the IIOP message will contain an ACL (Agent Communication Language) message wrapped in a *letter* object. A letter object has the following syntax:

```
(letter
   :envelope
     (...)
   :message
      (...))
```

Where `:message` contains a standard ACL message and `:envelope` contains the ultimate recipient of the message (mandatory), security information (optional) and transport preferences (optional). Both the `:envelope` and the `:message` are encapsulated within a letter object. The ACC will read and possibly edit the information in the :`envelope` parameter for message routing purposes ; the ACC is not required to (and indeed for encrypted messages may not be able to) read the contents of the `:message` parameter.

The `:envelope` can contain at least the following parameters:

`:destination` The final destination of the ACL message, composed of:

`:name` GUID of the receiving agent. (Mandatory).

`:address` A list of one or more well formed addresses. (Optional).

`:sender-details`

`:name` GUID of the sending agent. (Mandatory).

`:address` A list of one or more well formed addresses. (Optional).

Other parameters may include requests for delivery receipts, error report handling, message buffering, how the message content has been encoded, priority of the message etc. The use of these extra parameters is not specified by FIPA. The minimal form of an envelope for an agent sending a message includes only the GUID of the sender and the receiver. Note that the receiver name is sufficient to find an address for the receiver (either by looking up the name in the local AMS or forwarding the message to the receivers HAP (whose address can be extracted from the GUID)). The following example is a letter addressed the agent John:

```
 (letter
   :envelope(
      :destination(
         (:name   john@fipa.org:50/acc)
         (:address (iiop://fipa.org:50/acc)))
      :sender-details (
         (:name sally@agentland.com:50/acc))
            )
   :message
         (inform
            :sender      sally@agentland.com:50/acc
            :receiverjohn@fipa.org:50/acc
            :ontologygenealogy
            :languageKIF
            :content (…)))
```

8.3 Sending Messages

Agents can send messages in one of two main ways 1) using the IPMT system or 2) requesting an ACC (either local or remote) to forward it.

8.3.1 Using the IPMT

The IPMT may be able to determine automatically if a message passed to it by an agent is for an agent local to the AP or needs to be sent to a remote AP. In the latter case the IPMT passes the message to the ACC which will then handle the forwarding of the message to the remote destination.

8.3.2 Requesting an ACC to forward a message:

Each ACC must support requests for forwarding letters to agents (this is a forward action). The syntax for such a request is as follows:

```
    (letter
       :envelope(
          :destination (
             (:name    <acc-being-requested>)
             (:address (<acc's address>)))
          :sender-details (
             (:name <requesting-agent>))
                )
       :message
          (request
             :sender   requesting-agent>
             :receiver <acc-being-requested>
             :ontology fipa-agent-management
             :language SL0
             :protocol fipa-request
             :content
                (action     <acc-being-requested>
                   (forward
                       (letter
                            :envelope(…)
                            :message (…)))))))
```

Note that this request is also a letter addressed to the ACC from the agent. An agent can make use of this request mechanism in two contexts:

1) By sending a request to the ACC of the AP the sending agent is currently resident on.

2) By sending a request to a remote ACC. To do this the agent will also need to support IIOP.

The main use for this request mechanism supported by the ACC is for messaging between ACCs which is described below. The more usual way for an agent to send a message is through the IPMT.

8.4 Receiving messages

In general an agent will receive the whole letter object including the envelope. This means the receiving agent has access to information on what happened to the letter during transit. How the agent physically receives a message is

dependent upon the AP implementation and not addressed by FIPA. It is recommended that the AP provide some form of buffering capability to help agents manage their messages.

8.5 Transfer and routing of messages.

If a message is sent between two agents on the same AP this operation remains entirely inside the IPMT and is not specified by FIPA. FIPA only specifies the nature of inter-AP message transfer. This is necessary to guarantee interoperability between FIPA compliant APs. On any given AP it is the ACC that is primarily responsible for message exchange with other APs. Message transport between APs which does not go via an ACC is not covered by this specification.

The standard interface of an ACC for accepting message traffic to handle is the request to perform a forward action. This is also the standard way to transfer messages between APs. One ACC is able to request another to forward a message for it[4]. In the following example the ACC at `somewhere.org` is requesting that the ACC at `agentland.com` forwards a letter to agent Peter (informing Peter of the weather forecast).

```
(letter
 :envelope   (
    :destination   (
       (:name   acc@iiop://agentland.com:50/acc)
       (:address (iiop://agentland.com:50/acc)))
    :sender-details (
       (:name acc@iiop://somewhere.com:50/acc))
          )
 :message
    (request
       :sender     acc@iiop://somewhere.com:50/acc
       :receiver   acc@iiop://agentland.com:50/acc
       :ontology   fipa-agent-management
       :language   SL0
       :protocol   fipa-request
       :content
          (action  acc@iiop://agentland.com:50/acc
             (forward
                (letter
                   :envelope  (
                       :destination (
                            (:name  peter@iiop://agentland.com:50/acc)
                            (:address (iiop://agentland.com:50/acc)))
                       :sender-details (
                            (:name john@iiop://somewhere.org:50/acc))
                            )
                       :message
                            (inform
                            :sender john@iiop://somewhere.org:50/acc
```

[4] FIPA99 wil look into request forward as an interface to the message transport mechanism.

```
                              :receiver peter@iiop://agentland.com:50/acc
                                        :ontology   weather-ontology
                                        :language  a-content-language
                                        :content   (weather-forecast ‘rain)
                              ))))) … )
```

Here the ACC at `somewhere.org` is attempting to forward a message on behalf of the original sender agent John. If the agent John had been able to support IIOP and act as its own ACC it would be able to contact the ACC at `agentland.com` directly to request the forwarding action. However, in this case the agent John could have simply sent the message using the IPMT on its home AP (`somewhere.com`), the IPMT then recognised that the message needed to be sent to another AP and passed it to the ACC at `somewhere.com` which then wraps the message in the appropriate request. One thing to note about communication between ACCs is that ACCs are required to insert the return-address field in the envelope. This facilitates exception reporting.

The ACC receiving the request message will respond according to the FIPA request protocol. The ACC will check with the AP AMS to see if the agent identified by the GUID in the :`destination` parameter of the :`envelope` is registered on the AP. If the destination agent is not registered then the ACC returns a refuse message to the originating ACC (as specified in the request protocol). The following example is a refuse message for the request earlier in the section.

```
(letter
      :envelope (
            :destination(
                  (:name acc@iiop://somewhere.org:50/acc)
                  (:address (iiop://somewhere.org:50/acc)))
            :sender-details (
                  (:name acc@iiop://agentland.com:50/acc)
                  (:address (iiop://agentland.com:50/acc)))
            )
     :message
            (refuse
                :sender acc@iiop://agentland.com:50/acc
                :receiver acc@iiop://somewhere.org:50/acc
                :ontology fipa-agent-management
                :language     SL0
                :context  fipa-request
                :content
                    (refuse unavailable
                        (action acc@iiop://agentland.com:50/acc
                          (forward
                            (letter
                               :envelope(
                                  :destination(
                                       (:name peter@iiop://agentland.com:50/acc)
                                       (:address(iop://agentland.com:50/acc))))
                               :message
                                  (inform
                                       :sender john@iiop://somewhere.org:50/acc
                                       :receiver peter@iiop://agentland.com:50/acc
                                       :ontology      weather-ontology
                                       :language     a-content-language
                                       :content   (weather-forecast ‘rain)
                                            ))))) … )
```

If the agent is registered with the AP the ACC will then attempt to forward the message to the address provided by the AMS. If the translated address is a local AP address then the AP will handle this in an implementation-dependent manner. After forwarding the message the ACC will send an inform message to the originating ACC (as specified in the request protocol) containing the content string `Done(<forward action>).`

If the current address held by the AMS for the destination agent is not a local address the ACC will attempt to forward the message to the specified AP. To forward the message to the agent on another AP the ACC replaces the old address in the `:destination` parameter in the message `:envelope` with the new address obtained from the AMS. A request message containing the letter is then sent to the ACC on the remote AP. Only when the ACC receives confirmation of successful forwarding (an inform message containing the string `Done(<forward action>))` from the ACC at the new address can it send a confirmation to the originating ACC.

8.6 Multiple Addresses

The address parameter of the `:destination` in the `:envelope` of a letter may contain multiple addresses, An ACC uses these in the following way:

1. The ACC should always try to deliver to the first address on the list before trying the others in order.
2. Whenever an ACC is unable to deliver a message to one of the addresses on the list (because the specified AP is unavailable, the agent is not registered etc.) the ACC removes this address from the list and tries the next.
3. If the ACC finds that the specified receiver is registered but has left an off-AP forwarding address (or list of addresses with the AMS this forwarding information is added to the current list of addresses in the `:envelope` parameter. More precisely the new address (or address list) is added into the list on the envelope above the addresses already there. That is the forwarding information left behind by the destination agent takes priority over the information originally given by the sending agent.
4. If delivery is still unsuccessful when all addresses have been tried (the address list is empty) the appropriate error message for the *final* failure is passed back (the errors causing the failures of previous addresses are not returned).

9 FIPA Baseline Protocol and ACC

9.1 Overview

FIPA defines a baseline protocol for AP interoperability, (see figure 3). This means that there is always a well known method for agents on different FIPA-compliant APs to interoperate. Although, FIPA mandates the use of a baseline protocol, it does not preclude the use of other additional common protocols when one can be agreed between agents.

The internal AP message transport mechanism should support the FIPA baseline protocol. It must be able to distinguish between internal and external recipients, and use the most appropriate way of delivering the message, (i.e. internal mechanism, baseline or other appropriate protocol).

The ACC should be considered an agent for meta-level control of communication, (i.e. establishing message forwarding, setting time-outs etc.). It is responsible for routing agent messages between FIPA compliant APs. It has a management interface to the internal AP message transport mechanism which FIPA doesn't define.

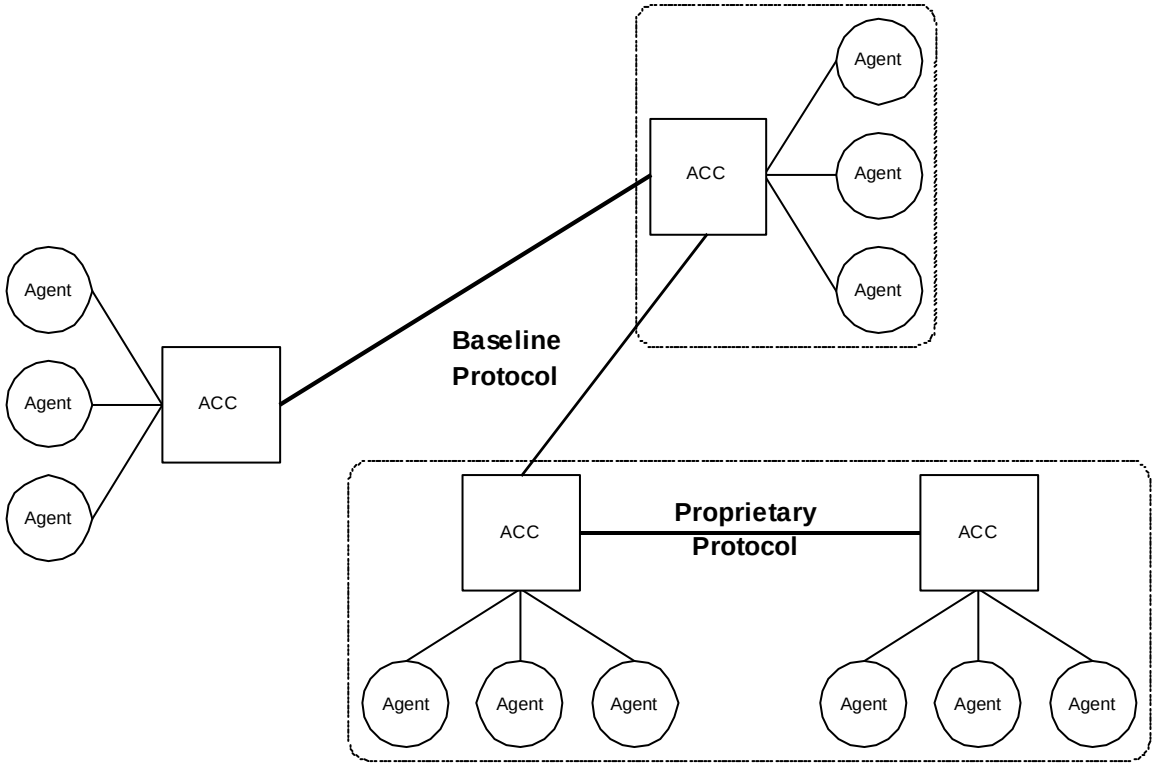

Figure 3 : The FIPA baseline protocol

9.2 IIOP

The FIPA97 baseline protocol is IIOP[5]. FIPA states that in order to be FIPA compliant an AP must minimally support IIOP[1]. The purpose of this requirement is to enable interoperability between APs. As such no requirements are

[5] FIPA is mandating a normative baseline communications protocol in order to guarantee interoperability between independently developed APs. There is little point in mandating a baseline protocol unless such interoperability can be assured.

There are two means of guaranteeing interoperability, one is to develop a tight specification of a communications protocol another is to adopt an available protocol developed specifically for interoperability. In this manner FIPA can guarantee interoperability without having to develop its own specification.

placed upon the communications capabilities of agents themselves or how messages are delivered between agents resident on the same AP. FIPA compliant agents resident on an AP have access to an ACC with IIOP capabilities on that AP through which communication with FIPA compliant agents registered on other AP's is enabled. The minimum requirement for compliance therefore is that every FIPA compliant AP provides an ACC which supports IIOP . If an ACC does not support IIOP then that AP is not FIPA compliant. Any ACC can of course support additional transport protocols, and communication between FIPA agents registered on different APs can occur over any of these protocols when available on both APs, however IIOP must always be available. Therefore, there is always at least one well-known method of communication available between all FIPA compliant APs.

For more information see Annex C.

10. Device Mobility

10.1 Intermittent connectivity and session mobility

Agents may reside on an AP that is on a host which is temporarily disconnected from the network, whether through deliberate act or a failure in part of the network. Typically, device mobility occurs with hardware that can be physically moved, such as laptops, PDAs and mobile phones.

To support these scenarios, an agent can nominate a third party (or a number of third parties) to handle its messages while it is disconnected, migrating or not executing on a particular host. This is achieved by an agent specifying a number of potential recipient addresses as the value of the `:delegate-agent` parameter. The default action of such a list can be summarised as:

1. Upon receipt of a message, the ACC should check to see if the recipient agent is currently executing on this AP. If it is, then the message is delivered as normal.
2. If the agent is not currently executing, then the first message in the `:delegate-agent` parameter is removed and the message is forwarded to the next address in the `:delegate-agent` parameter. By removing addresses that have already been visited, no looping can occur in the forwarding process[6]. An address is only removed if the message arrives there successfully.
3. If there are no more addresses in the `:delegate-agent` parameter, then the ACC buffers the message for subsequent retrieval.

Agents that are executing on intermittently connected devices can embed this information in the `:reply` parameter of their out-going messages. Thus, when an agent that has received one of these messages replies to it, there is a list of potential delivery addresses that the ACC can use to try and re-route the message in the event that it cannot be delivered to the primary address. The following example:

```
(letter
      :envelope     (  :reply iiop://host1/acc iiop://host3/acc)
      :message (request
```

The specification of a small IDL interface is sufficient to guarantee interoperability when combined with the OMGs IIOP specification. The issue of interoperability compliance is greatly simplified as compliance is determined by whether or not the implementation of the FIPA_Agent_97 interface has been implemented in conformance with the OMGs IIOP specification.

IIOP can assure interoperability as it is maintained by a well established and large consortium committed to interoperability. This specification is maintained and both commercial and free implementations are widely available. Furthermore, it is conceivable that by building on this effort that technical advancements will be available in the future, in particular if a message syntax other than string was introduced for FIPA ACL[2].

It is proposed in the FIPA 1999 Call for Proposals that the FIPA baseline protocol be revisited as part of the FIPA99 work plan and so may subsequently be revised.

[6] Unless an address appears in the `:delegate-agent` field more than once

```
                  :sender foo@iiop://host1:40/acc
                  :receiver bar@iiop://host2:40/acc
                  :content (..))
```

would inform the receiving agent bar on host2 that foo on host1 can be contacted primarily at `iiop://host1/acc`; if it is unavailable here, its alternate contact address is `iiop://host3/acc`. When replying to this message, the `:reply` parameter becomes a `:delegate-agent` parameter, thus:

```
(letter
      :envelope (:delegate-agent iiop://host1/acc iiop://host3/acc)
      :message (
            refuse
                  :sender bar@iiop://host2:40/acc
                  :receiver foo@iiop://host1:40/acc
                  :content ..)  )
```

So, if the ACC on host2 cannot deliver the message to the ACC on host1 (because it is disconnected from the network, say), then it will forward the message to the ACC on host3. If the Agent cannot be reached there and host3 supports message buffering, the message is stored there until it is retrieved. If there is no storage capability on host3, an error message is sent back. Upon reconnection, the ACC on host1 should contact the ACC on host 3 and download all of its stored messages.

Due to the fact that the `:reply` and `:delegate-agent` parameters can contain multiple addresses, this method of handling messages can help to deal with negotiating nested firewalls. If an agent is communicating with another agent across a firewall, then the agents can communicate through the firewall machine[7] by specifying the firewall machine as a value in the `:delegate-agent` parameter. When one of these agents tries to contact the other, the ACC will attempt to deliver the message to the AP beyond the firewall (which would be inaccessible), so it forwards it to the firewall machine whose address is subsequently removed from the list. The ACC on the firewall machine will then forward the message to the ACC on the primary address (which will also subsequently be removed from the list), because the agent is not executing on the AP of the firewall machine. Nested firewalls can be negotiated by specifying the address of each firewall machine in the `:delegate-agent` parameter.

10.2 Synchronisation

When several agents share a responsibility, they need some kind of mechanism to synchronise their knowledge. Typically the disconnected PDA needs to update its knowledge as well as its proxy knowledge when reconnecting to the network.

FIPA mandates a minimum mechanism for synchronisation. When an agent re-connects, it has two ways of recovering messages:

- it contacts its HAP ACC to see if messages are stored there, if so they will be communicated in a FIFO order,
- in case it has specified a proxy (or proxies), it is the duty of the agent to contact the proxy to download any stored messages.

These re-connection mechanism allow an agent to control the storage of messages, either on its HAP or on its proxies.

[7] This requires that some FIPA infrastructure is set up on each firewall machine that is to be negotiated (a minimum of an AMS and an ACC).

10.3 Forwarding messages to a proxy agent

Agents may be physically disconnected from one AP rendering them un-contactable until they are re-connected to an AP. Similarly, agents may be disconnected from an AP for prolonged periods of time if they are resident on devices such as laptop computers or mobile phones. In such situations, an agent can request that the AMS forward all messages to another delegated agent.

This delegated authority may have simple functionality such as the ability to buffer messages for later retrieval or more complex ability to act on behalf of the instructing agent.

It is envisaged that this action would be used by an agent prior to it physically being unplugged from an AP or in preparation for its migration to another AP. It is the responsibility of the agent to cancel the forward request once it has re-established itself on an AP.

The ability to delegate authority to another agent is restricted to the instructing agent only. In situations where an attempt is made by a third party agent to delegate responsibility of one agent to another the request action will be refused by the AMS.

The AMS supports the setting-up of an alternate recipient for an agent's messages. Thus Peter could set the AMS to re-direct any messages sent to Peter to Jane. To do this requires modifying the :delegate-agent attribute of the agent entry in the AMS.

11 FIPA Agent Management Grammar and Ontology

11.1 Letter Grammar

```
FIPA-letter =           “(“  “letter” Message-envelope  Message-content “)”

Message-envelope =      “:envelope” “(“ Envelope-parameter+  “)”

Message-content =       “:message” ACL-Message+

Envelope-parameter =    “:destination” “(“ Envelope-value “)”|
                        “:sender-details” “(“ Envelope-value “)”|
                        “:delegate-agent” Agent-name |
                        “:reply” Agent-name

Envelope-value =        “:name” Agent-name |
                        “:address” “(“ Agent-address + “)”

Agent-name =            (see AgentName definition below)

Agent-address =         (see CommAddress definition below)

ACL-Message =           (see Section 6.4.1FIPA97 Part 2)
```

11.2 Agent Management Grammar

This agent management content syntax and grammar should be read as an extension to the Agent Communication Language syntax defined in Part 2 of FIPA97.

This agent management grammar is the definition of terms for Agent Management using SL0, (see Annex B and B3.1 in FIPA97 Part 2).

Agent Management Actions

```
AgentManagementAction = “(“ “register DF-agent-description“)”
                      |“(“ “deregister” DF-agent-description“)”
                      |“(“ “modify” DF-agent-description“)”
                      |“(“ “search” DF-agent-description Constraint+“)”
                      |“(“ “register-agent” AMS-agent-description “)”
                      |“(“ “deregister-agent” AMS-agent-description“)”
                      |“(“ “authenticate” AMS-agent-description “)”
                      |“(“ “modify-agent” AMS-agent-description “)”
                      |“(“ “forward” ACLCommunicationAct “)”
                      |“(“ “search-agent” AMS-agent-description “)”
                      |“(“ “query-platform-profile” AP-description “)”.
```

Agent Management Object Descriptions

```
DF-agent-description=    “(“ “:df-agent-description” FIPA-DF-agent-description+ “)”.

AMS-agent-description =  “(“ “:ams-agent-description” FIPA-AMS-agent-description+“)”.

AP-description =              “(“ “:ap-profile“ FIPA-AP-description “)”.
```

```
FIPA-DF-agent-description =   "(" ":agent-name" AgentName")"
                            |"(" ":address" CommAddress+")"
                            |"(" ":services" FIPA-service-desc+ ")"
                            |"(" ":type" Word")"
                            |"(" ":interaction-protocols" "(" Word+ ")""")"
                            |"(" ":ontology" SL0Term")"
                            |"(" ":language" "(" ContentLanguage+ ")""")"
                            |"(" ":ownership" SL0Term")"
                            |"(" ":df-state" DfLifecycleState")".

FIPA-AMS-agent-description =  "(" ":agent-name" AgentName")"
                            |"(" ":address" CommAddress+ ")"
                            |"(" ":signature" Word")"
                            |"(" ":ap-state" APState")"
                            |"(" ":delegate-agent-name" AgentName")"
                            |"(" ":ownership" Word")".

FIPA-service-desc    = "(" ":service-description" FIPA-service-desc-Item+ ")".

FIPA-service-desc-Item ="(" ":service-name" Word ")"
                            |"(" ":service-type" ServiceTypes ")"
                            |"(" ":service-ontology" SL0Term ")"
                            |"(" ":fixed-properties" FixedProperties")"
                            |"(" ":negotiable-properties" SL0Term ")".

FIPA-AP-description[8] =  "(" ":platform-name" Word")"
                            |"(" ":iiop-url" URL")"
                            |"(" ":dynamic-registration" Boolean")"
                            |"(" ":mobility" Boolean")"
                            |"(" ":ownership" Word")"
                            |"(" ":certification-authority" Word")"
                            |"(" ":fipa-man-compliance" FipaSpecifications+ ")".

DfLifecycleState =      "active"
                            |"suspended"
                            |"retired".

FipaSpecifications =    "fipa97-part1-v1"
                            | "fipa97-part1-v2"
                            | "fipa98-part1-v1".

APState =               "initiated"
                            |"active"
                            |"suspended"
                            |"waiting".

ContentLanguage =       "fipa-sl0"
                            | "fipa-sl1"
                            | "fipa-sl2"
                            | Word.
```

[8] The FIPA-AP-Description contains the characteristics of the AP profile. Additional optional parameters have been added by the FIPA Security Management specification. This is not used in the FIPA97 part 1 specification. However, management operations for querying the AP profile have been incorporated into the FIPA98 part 1 specification.

```
ServiceTypes =          “fipa-df”
                        |“fipa-ams”
                        |“fipa-acc”
                        |“fipa-agent”
                        | Word.

FixedProperties =       “(” “:df-search-def-time-out”Duration “)”
                        | “(” “:df-search-algo”Word “)”
                        | SL0Term.
```

Agent Management Exception Propositions

```
AgentManagementException = “(“ “:fipa-man-exception“FipaException+ “)“.

FipaException =
                         “(“ “no-communication-means” ManOb-description“)”
                        |“(“ “unavailable” ManOb-description“)”
                        |“(“ “agent-not-registered” ManOb-description“)”
                        |“(“ “unrecognised-attribute-value”
                              ManOb-description“)”
                        |“(“ “unrecognised-attribute” ManOb-description“)”
                        |“(“ “unauthorised” “)”
                        |”(“ “failed-management-action” “)”
                        |“(“ “unwilling-to-perform” “)”
                        |“(“ “df-overloaded” “)”
                        |“(“ “ams-overloaded” “)”
                        |“(“ “acc-overloaded” “)”
                        |“(“ “unable-deregister” “)”
                        |“(“ “inconsistency” “)”
                        |“(“ “agent-already-registered” “)”.

Constraint =            |“(“ “:df-search-resp-req” min  Integer“)”
                        |“(“”:df-search-algo” Word
                             (ConstraintFn Integer)+“)””)”
                        |“(“ “:df-search-filter” CommAddressFilter“)”
                        |“(“ “:df-search-time-out” DateTimeToken DateTimeToken “)”.

CommAddressFilter =     (CommProtocol|"*") “://”(IPAddress|DNSName|"*")
                              “:”Integer “/” (ACCObj|"*").

ConstraintFn =          “max”
                        |“min”.

ManOb-description =     FIPA-DF-agent-description
                        | FIPA-AMS-agent-description.

AgentName =             Word “@” CommAddress.

CommAddress =           CommProtocol “://”(IPAddress|DNSName) “:” Integer “/” ACCObj.

CommProtocol =          [“a”-“z”,”A”-“Z”] [“a”-“z”,”A”-“Z”,”0”-“9”,”_”]*.

IPAddress =             Integer “.”Integer “.”Integer “.”Integer.

DNSName =               Word.

ACCObj =                Word.
```

```
DateTimeToken[9] =          "+" ?
                            Year Month Day "T"
                            Hour Minute Second MilliSecond
                            (TypeDesignator ?).

Year =                      Digit Digit Digit Digit.
Month =                     Digit Digit.
Day =                       Digit Digit.
Hour =                      Digit Digit.
Minute =                    Digit Digit.
Second =                    Digit Digit.
MilliSecond =               Digit Digit Digit.
TypeDesignator =            AlphaCharacter.

Digit                       = ["0" – "9"].
```

11.3 Rules for Well Formed Agent Management Messages

The following tables illustrate the mandatory attributes to ensure correct formation for each of the actions defined in this specification. This section further defines the range of permitted expressions in agent management messages. Each table describes the use of a single object. Attributes which are listed as optional can be used to form syntactically correct management actions, however the attribute may have no semantics for that action. The syntax for the actions is given above.

df-agent-description

Attribute	Action			
	register	deregister	modify	search
:agent-name	M	M	M	O
:services	O	O	O	O
:type	M	O	O	O
:interaction-protocols	O	O	O	O
:ontology	O	O	O	O
:language	O	O	O	O
:address	M	O	O	O
:ownership	M	O	O	O
:df-state	M	O	O	O

M = Mandatory O = Optional

[9] See FIPA97 Part 2 Section 6.4.1 and 6.4.2 for this definition and its relation to ISO 8601.

Where services are specified in a `FIPA-DF-agent-description` the following applies :

`service-description`

Attribute	
`:service-name`	M
`:service-type`	M
`:service-ontology`	M
`:fixed-properties`	M
`:negotiable-properties`	O

M = Mandatory O = Optional

`ams-agent-description`

Attribute	Action				
	`modify-agent`	`authenticate`	`register-agent`	`deregister-agent`	`search-agent`
`:agent-name`	M	M	M	M	O
`:address`	O	O	M	O	O
`:ap-state`	O	O	M	O	O
`:delegate-agent-name`	O	O	O	O	O
`:signature`	O	O	O	O	O

M = Mandatory O = Optional

`ap-profile`

Attribute	Action
	`query-platform-profile`
`:platform-name`	M
`:iiop-url`	O
`:dynamic-registration`	O
`:mobility`	O
`:ownership`	O
`:certification-authority`	O
`:fipa-man-compliance`	O

M = Mandatory O = Optional

The management actions search-agent and search do not enforce mandatory attributes, however a well formed message must include at least one attribute.

All management actions using the FIPA-Request protocol will, if successful, yield a inform Done message from the agent which performed the action. The search action is the exception to this rule as it will yield an inform Result when successful.

The semantics of the operators used as constraints for the `search` action is defined as:

Constraint	Operator	Description
`:df-search-resp-req`	max	The search stops as soon as max the number of answers have been found by the DF which initiated the search. When forwarding `search` requests to other DFs, the current DF has to decrement the number of answers from the max value forwarded to other DFs.
`:df-search-algo`	min max	`search` algorithms can be advertised as `:agent-services` when a `DF` registers its services. If the search algorithm is not specified for a particular DF, the default algorithm is a depth first search. The search algorithm constrains the number of hops by specifying integer min and max values, giving relative position to the original DF.
`:df-search-filter`	CommAddressFilter	The CommAddressFilter allows an agent to specify the domain in which the search can be performed. It will filter the addresses of the searched DFs according to their name. The filter supports the * character with its standard meaning.
`:df-search-time-out`	Time & Duration	Each search is initiated with its start time as well as a time-out duration so as to discard any request beyond the time-out. The default time-out can be advertised in the DF service description. The time stamp of the DF is the absolute time.

An agent can access the services a DF offers by issuing a basic search (i.e. without using the above constraints) against that DF. The DF service description can contain the following parameters :

`:df-search-def-time-out`	Specifies the default time-out for a search offered by this DF.
`:df-search-alg`	Lists the search algorithms available from this DF.

11.4 Exceptions

All agent management operations use the fipa-request protocol, (FIPA97 part 2). Exceptions are reported in `refuse` or `failure` communicative acts.

For example: a failure of an agent registration with a DF.

```
(failure
            :sender   a-df-agent@iiop://fipa.org:50/acc
            :receiver agent@iiop://fipa.org:50/acc
            :content
              ((action a-df@iiop://fipa.org:50/acc
                 (register
                  (:df-agent-description
                    (:agent-name an-agent@iiop://fipa.org:50/acc)
                    (:interaction-protocols (fipa-request))
                    (:ontology fipa-agent-management)
                    (:address iiop://fipa.org:50/acc)
                    (:type travel-agent)
                    (:ownership fipa.org)
                    (:df-state  active))))
                (:fipa-exception
                        agent-already-registered))
            :language sl0
            :protocol fipa-request
            :ontology fipa-agent-management)
```

Example 2 : A DF registration refusal due to an ill-formed agent name.

```
(failure
            :sender   a-df-agent@iiop://fipa.org:50/acc
            :receiver agent@iiop://fipa.org:50/acc
            :content
              ((action a-df@iiop://fipa.org:50/acc
                 (register
                  (:df-agent-description
                    (:agent-name an-agent@iiop://fipa.org:50/acc)
                    (:interaction-protocols (fipa-request))
                    (:ontology fipa-agent-management)
                    (:address iiop://fipa.org:50/acc)
                    (:type travel-agent)
                    (:ownership fipa.org)
                    (:df-state  active))))
                (:fipa-exception
                     (unrecognised-attribute-value
                     (:agent-name an-agent@iiop://fipa.org:50/acc))))
            :language sl0
            :protocol fipa-request
           :ontology fipa-agent-management)
```

11.5 Agent Management Actions

11.5.1 register

Supported by	DF	
Description	An agent registers its services in order to publicise some or all of them to other agents. There is *no* intended future commitment or obligation, on the part of the registering agent implied in the act of registering. For example, an agent can refuse a request for a service which is advertised through a DF. There is a commitment on behalf of the DF to honestly broker information it holds. When an agent applies for registration in a domain an agent description must be supplied containing values for all of the mandatory attributes of the agent description. It may also supply optional and private fields, containing non-FIPA standardised information an agent developer might want included in the directory.	
Content	`df-agent-description` (see sections 11.2 and 11.3).	
FIPA Protocol	`fipa-request`	

Example	`(request`
	`:sender an-agent@iiop://fipa.org:50/acc`
	`:receiver a-df@iiop://fipa.org:50/acc`
	`:content`
	`(action a-df@iiop://fipa.org:50/acc`
	`(register`
	`(:df-agent-description`
	`(:agent-name an-agent@iiop://fipa.org:50/acc)`
	`(:services`
	`(:service-description`
	`(:service-type video-on-demand)`
	`(:service-ontology itut-vod)`
	`(:service-name vod-1)`
	`(:fixed-properties (genre sport))))`
	`(:language fipa-sl0)`
	`(:type selling-agent)`
	`(:address iiop://fipa.org:50/acc)`
	`(:ownership fipa.org)`
	`(:df-state active))))`
	`:language sl0`
	`:protocol fipa-request`
	`:ontology fipa-agent-management)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in one of the attribute values.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised by the agent.
	`agent-already-registered`	This exception occurs if the agent to be registered is already in the DF.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the DF is refusing to perform the action.
Failure Reasons	`df-overloaded`	This occurs when the DF fails to complete due to processing resource overload.

11.5.2 search

Supported by	DF
Description	A search action involves a request for information from a DF. The DF does not guarantee the validity of the information provided in response to a search request. A search is satisfied with the DF identifying agent entry in the directory that satisfy the content of the query. This could entail the escalation of the search to other DF's if the query cannot be resolved locally. A search can be defined to constrain the action of the DF. A successful search can return one or more agent descriptions that satisfies the search criteria. A nil return is returned where no agent entries satisfy the search criteria.
Content	`df-agent-description` (see sections 11.2 and 11.3).
FIPA Protocol	`fipa-request` (see FIPA97 Part 2)

Example	`(request`
	`:sender an-agent@iiop://fipa.org:50/acc`
	`:receiver a-df@iiop://fipa.org:50/acc`
	`:content`
	`(action a-df@iiop://fipa.org:50/acc`
	`(search`
	`(:df-agent-description`
	`(:address iiop://fipa.org:50/acc)`
	`(:df-state active))`
	`(df-search-algo depth-first max 1)`
	`(df-search-resp-req max 1)))`
	`:language sl0`
	`:reply-with id2543`
	`:protocol fipa-request`
	`:ontology fipa-agent-management)`

Reply	The above query requests all agent names where the agent is registered as active and has the address `iiop://fipa.org:50/acc.`
	The reply would be a `result,` for example:
	
	`(inform`
	`:sender a-df@iiop://fipa.org :50/acc`
	`:receiver an-agent@iiop://fipa.org:50/acc`
	`:content`
	`(:df-agent-description`
	`(:agent-name an-agent@iiop://fipa.org:50/acc)`
	`(:agent-service`
	`(:service-description`
	`(:service-type video-on-demand)`
	`(:service-ontology itut-vod)`
	`(:service-name vod-1)`
	`(:fixed-properties (genre sport))))`
	`(:interaction-protocols (fipa-request))`
	`(:ontology itu-t))`
	`:language sl0`
	`:in-reply-to id2543`
	`:protocol fipa-request`
	`:ontology fipa-agent-management)`

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in one of the attribute values.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the DF is too busy or overloaded with other operations.
Failure Reasons	`df-overloaded`	This occurs because the DF fails to finish the search operation because of processing resource overload.

1286

11.5.3 modify

Supported by	DF	
Description	Involves the changing of an agent's details in a particular DF directory. The content of a modify message will replace only those attributes which are contained in the `modify df-agent-description.` .	
Content	`df-agent-description` (see sections 11.2 and 11.3).	
FIPA Protocol	`fipa-request` (see FIPA97 Part 2)	

Example	`(request`
	`:sender an-agent@iiop://fipa.org:50/acc`
	`:receiver a-df@iiop://fipa.org:50/acc`
	`:content`
	`(action a-df@iiop://fipa.org:50/acc`
	`(modify`
	`(:df-agent-description`
	`(:agent-name an-agent@iiop://fipa.org:50/acc)`
	`(:df-state suspended))))`
	`:language sl0`
	`:protocol fipa-request`
	`:ontology fipa-agent-management)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in one of the attribute values.
	`inconsistency`	DF rejected the modification because it provides conflicting information.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the DF is too busy or overloaded with other operations.
Failure Reasons	`df-overloaded`	This occurs because the DF fails to finish the modification operation because of processing resource overload.

11.5.4 deregister

Supported by	DF	
Description	An agent de-registers in order to remove any record of its attribute(s) from a domain. The de-register action has the consequence that there is no-longer a commitment on behalf of the DF to broker information relating to that agent.	
Content	`df-agent-description` (see sections 11.2 and 11.3).	
FIPA Protocol	`fipa-request` (see FIPA97 Part 2)	

Example	`(request`
	`:sender an-agent@iiop://fipa.org:50/acc`
	`:receiver a-df@iiop://fipa.org:50/acc`
	`: content`
	`(action a-df@iiop://fipa.org:50/acc`
	`(deregister`
	`(:df-agent-description`
	`(:agent-name an-agent@iiop://fipa.org:50/acc)))`
	`:language sl0`
	`:ontology fipa-agent-management`
	`:protocol fipa-request)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in one of the attribute values.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the DF is too busy or overloaded with other operations.
	`unable-to-deregister`	The agent can not be deregistered because it has still pending contracts, or because the agent is not found in the DF.
Failure Reasons	`df-overloaded`	This occurs because the DF fails to finish the operation because of processing resource overload.

11.5.5 register-agent

Supported by	AMS	
Description	The register-agent action involves the registration of an agent's attributes including its GUID and associated communication address(es) with an AMS.	
Content	`ams-agent-description` (see sections 11.2 and 11.3).2	
FIPA Protocol	`fipa-request` (see FIPA97 Part 2)	

Example	`(request`
	`:sender myagent@iiop://fipa.org:50/acc`
	`:receiver an-ams@iiop://fipa.org:50/acc`
	`:content`
	`(action an-ams@iiop://fipa.org:50/acc`
	`(register-agent`
	`(:ams-agent-description`
	`(:agent-name myagent@iiop://cmp.de:99/acc2-id)`
	`(:address iiop://inf.co.uk:90/acc-id)`
	`(:signature agent-sig))))`
	`:language sl0`
	`:ontology fipa-agent-management`
	`:protocol fipa-request)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in one of the attribute values.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`agent-already-registered`	This exception occurs if the agent to be registered is already in the AMS.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the AMS is too busy or overloaded with other operations.
Failure Reasons	`ams-overloaded`	This occurs because the AMS fails to finish the modification operation because of processing resource overload.

11.5.6 deregister-agent

Supported by	AMS	
Description	An agent de-registers in order to remove any record of its attribute(s) from an AMS. The AMS can be requested to deregister on behalf of another agent during agent migration.	
Content	`ams-agent-description` (see sections 11.2 and 11.3).	
FIPA Protocol	`fipa-request` (see FIPA97 Part 2)	

Example	`(request`
	`:sender an-agent@iiop://fipa.org:50/acc`
	`:receiver ams-agent@iiop://fipa.org:50/acc`
	`:content`
	`(action ams-agent@iiop://fipa.org:50/acc`
	`(deregister-agent`
	`(:ams-agent-description`
	`(:agent-name an-agent@iiop://fipa.org:50/acc)))`
	`:language sl0`
	`:ontology fipa-agent-management`
	`:protocol fipa-request)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in one of the attribute values.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the DF is too busy or overloaded with other operations.
	`unable-to-deregister`	The agent can not be deregistered because it has still pending contracts, or because the agent is not found in the AMS.
Failure Reasons	`ams-overloaded`	This occurs because the AMS fails to finish the operation because of processing resource overload.

11.5.7 search-agent

Supported by	AMS	
Description	A search action involves a request for information from an AMS. A search is satisfied with the AMS identifying an agent entry in its directory that satisfies the content of the query. An ams-agent-description will be returned as the result of a successful `search-agent` operation. An AMS may restrict for confidentiality reasons access to certain attributes in the ams-agent-description, for example, agent-state but will always return an agents address.	
Content	`ams-agent-description` (see sections 11.2 and 11.3).	
FIPA Protocol	`fipa-request` (see FIPA97 Part 2)	

Example	`(request`
	`:sender an-agent@iiop://fipa.org :50/acc`
	`:receiver a-ams@iiop://mmm.org:50/acc`
	`:content`
	`(action a-ams@iiop://mmm.org:50/acc`
	`(search-agent`
	`(:ams-agent-description`
	`(:agent-name an-agent@iiop://fipa.org:50/acc)))`
	`:language sl0`
	`:reply-with id2543`
	`:protocol fipa-request`
	`:ontology fipa-agent-management)`
	`))`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in one of the attribute values.
	`unrecognised-attribute`	This error occurs when one of the attributes in the message does not belong to the AMS object.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the AMS is too busy or overloaded with other operations.
Failure Reasons	`ams-overloaded`	This occurs because the AMS fails to finish the search operation because of processing resource overload.

11.5.8 modify-agent

Supported by	AMS	
Description	The modify-agent action Involves the changing of an agent's details in a particular AMS directory.	
Content	`ams-agent-description` (see sections 11.2 and 11.3).	
FIPA Protocol	`fipa-request` (see FIPA97 Part 2)	

Example	`(request`
	`:sender an-agent@iiop://fipa.org:50/acc`
	`:receiver ams-agent1@iiop://fipa.org:50/acc`
	`:content`
	`(action ams-agent1@iiop://fipa.org:50/acc`
	`(modify-agent`
	`(:ams-agent-description`
	`(:agent-name an-agent@iiop://fipa.org:50/acc)`
	`(:delegate-agent-name`
	`ams-agent2@iiop://fipa.org:50/acc))))`
	`:language sl0`
	`:ontology fipa-agent-management`
	`:protocol fipa-request)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in one of the attribute values.
	`inconsistency`	AMS rejected the modification because it provides conflicting information.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the AMS is too busy or overloaded with other operations.
Failure Reasons	`ams-overloaded`	This occurs because the AMS fails to finish the modification operation because of processing resource overload.

11.5.9 authenticate

Supported by	AMS	
Description	An agent can request that the AMS verifies an agent's identity.	
Content	`ams-agent-description` (see sections 11.2 and 11.3).	
FIPA Protocol	`fipa-request` (see FIPA97 Part 2)	

Example	`(request`
	`:sender an-agent@iiop://fipa.org:50/acc`
	`:receiver ams-agent@iiop://fipa.org:50/acc`
	`:content`
	`(action ams-agent@iiop://fipa.org:50/acc`
	`(authenticate`
	`(:ams-agent-description`
	`(:agent-name`
	`JB234@iiop://fipa.org:50/acc)`
	`(:ownership "John Brown")`
	`(:signature a-sig)))`
	`:language sl0`
	`:ontology fipa-agent-management`
	`:protocol fipa-request)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in the agent name or signature.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`reject-authenticate`	This occurs if the AMS does not authenticate the agent.
	`unwilling-to-perform`	This error occurs if the AMS is too busy or overloaded with other operations.
Failure Reasons	`ams-overloaded`	AMS failed to authenticate the agent due to internal resource problems.

11.5.10 forward

Supported by	ACC	
Description	An agent can ask an ACC to forward a message to a destination agent	
Content	ACLCommunicativeAct (see FIPA97 Part 2)	
FIPA Protocol	fipa-request (see FIPA97 Part 2)	
Example	(request :sender an-agent@iiop://fipa.org:50/acc :receiver an-acc@iiop://fipa.org:50/acc :content (action an-acc@iiop://fipa.org:50/acc (forward (request :sender an-agent@iiop://fipa.org:50/acc :receiver a-df@iiop://agentland.org:50/acc :content (action a-df@iiop://fipa.org:50/acc (modify (:ams-agent-description (:agent-name an-agent@iiop://fipa.org:50/acc) (:ap-state suspended)))) :language sl0 :protocol fipa-request :ontology fipa-agent-management))) :ontology fipa-agent-management :language sl0 :protocol fipa-request)	
Refuse Reasons	unrecognised-attribute-value	This error occurs when an invalid syntax was detected in the agent name or signature.
	unrecognised-attribute	This error occurs when an attribute of the content expression is not recognised.
	unauthorised	This occurs if the requesting agent is not sufficiently authorised.
	unwilling-to-perform	This error occurs if the ACC is too busy or overloaded with other operations.
	agent-not-registered	This error occurs if the destination agent is not registered in the AP.
	no-communications-means	This error occurs if there is no shared communication protocol to reach the destination agent.
Failure Reasons	unavailable	ACC failed to complete the action due to internal resource problems.

11.5.11 query-platform-profile

Supported by	AMS	
Description	An agent can request the profile of an AP from the AMS.	
Content	None	
FIPA Protocol	Fipa-request (see FIPA97 Part 2)	

Example	`(request`
	`:sender an-agent@iiop://fipa.org:50/acc`
	`:receiver an-ams@iiop://fipa.org:50/acc`
	`:content`
	`(action an-ams@iiop://fipa.org:50/acc`
	`query-platform-profile)`
	`:ontology fipa-agent-management`
	`:language sl0`
	`:protocol fipa-request)`	

Reply	The above query requests an AP profile. The reply would be an `inform` for example:
	`(inform`
	`:sender an-ams@iiop://fipa.org:50/acc`
	`:receiver an-agent@iiop://fipa.org:50/acc`
	`:content`
	`(:ap-profile`
	`(:platform-name united-e-commerce-ap)`
	`(:dynamic-registration yes)`
	`(:mobility no)`
	`(:ownership united-incorporated-plc)`
	`(:fipa-man-compliance fipa98-art1-v1))`
	`:language sl0`
	`:in-reply-to id2543`
	`:protocol fipa-request`
	`:ontology fipa-agent-management)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in the content expression.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`unauthorised`	This occurs if the requesting agent is not sufficiently authorised.
	`unwilling-to-perform`	This error occurs if the AMS is too busy or overloaded with other operations.
	`no-communications-means`	This error occurs if there is no shared communication protocol to reach the destination agent.
Failure Reasons	`unavailable`	AMS is unavailable.

11.5.12 quit

Supported by	All FIPA agents	
Description	An AMS can request an agent to terminate all execution on a given AP.	
Content	Agent platform name	
FIPA Protocol	Fipa-request (see FIPA97 Part 2)	

Example	`(request`
	`:sender an-ams@iiop://fipa.org:50/acc`
	`:receiver an-agent@iiop://fipa.org:50/acc`
	`:content`
	`(action an-agent@iiop://fipa.org:50/acc`
	`(quit`
	`(:platform-name a-platform@iiop://fipa.org:50/acc))`
	`)`
	`:ontology fipa-agent-management`
	`:language sl0`
	`:protocol fipa-request)`	

Refuse Reasons	`unrecognised-attribute-value`	This error occurs when an invalid syntax was detected in the content expression.
	`unrecognised-attribute`	This error occurs when an attribute of the content expression is not recognised.
	`unauthorised`	This occurs if the requesting AMS is not sufficiently authorised.
	`no-communications-means`	This error occurs if there is no shared communication protocol to reach the destination agent.
Failure Reasons	`unavailable`	Agent is unavailable.

11.6 Agent Management Objects

11.6.1 `df-agent-description`

Parameter	Description
`:agent-name`	Denotes the globally unique agent identifier.
`:type`	Identifies the type of agent described.
`:services`	Denotes the service(s) the agent can provide. This would include a description of the characteristics of the service description as well as the service description itself. See fipa-man-service-description.
`:interaction-protocols`	Characterises the protocols supported by the agent. This can include both standardised and/or non-standard protocols.
`:ontology`	Denotes the ontology or ontologies the agent can support.
`:language`	Denotes the content language(s) the agent can support.
`:address`	An agent must support at least one communication address and by definition if only one is provided, it must be the IIOP address of the AP on which the agent resides.
`:ownership`	Identifies the owner of the agent.
`:df-state`	Denotes the domain life-cycle state, for example suspended.

11.6.2 ap-profile

Parameter	Description
:platform-name(M)	Denotes a globally unique identifier for the AP
:iiop-url(M)	Denotes the IIOP URL of the AP
:dynamic-registration(M)	Denotes whether the AP supports dynamic registration
:mobility (M)	Denotes whether the AP supports agent mobility.
:ownership (M)	Identifies the owner of the AP.
:certification-authority (M)	Denotes the certification authority for the AP.
:fipa-man-compliance (M)	Denotes the FIPA specification(s) the AP is compliant with.

11.6.3 service-description

Parameter	Description
:service-name	Denotes the service name.
:service-type	Denotes the unique service type.
:service-ontology	Identifies the ontology for the service description.
:fixed-properties	A description of the permanent characteristics of the service. This could be a complex structure using a particular ontology defined in the :service-ontology parameter.
:negotiable-properties	A description of the dynamic properties of the service.

11.6.4 ams-agent-description

Parameter	Description
:agent-name	Denotes the globally unique agent name.
:address	An agent must support at least one communication address and by definition if only one is provided, it must be the IIOP address of the AP on which the agent resides.
:delegate-agent	Denotes the name of an agent, other than the agent that is the subject of the description, (i.e. identified under :agent-name) that has been delegated as recipient of all messages. It identifies an alternative recipient for a message.
:ap-state	Denotes the AP lifecycle state of the agent.
:ownership	Denotes the legal entity (individual or organisation) responsible for the actions of the agent.
:signature	Denotes the agents signature for authentication purposes[10].

[10] FIPA98 Part 10 V.1.0 Agent Security Management contains further information on security issues and mechanisms for multi-agent systems. Agent security management requires further work and is part of the FIPA99 call for proposals.

11.6.5 fipa-man-exception

Parameter	Description
unrecognised-attribute-value	This error occurs when an invalid syntax was detected in the agent name or signature.
unrecognised-attribute	This error occurs when the attribute identifiers which appear in the message are invalid.
unauthorised	This occurs if the requesting agent is not sufficiently authorised.
unwilling-to-perform	This error occurs if the recipient agent is refuses to perform a requested action..
Agent-not-registered	This error occurs if the destination agent is not registered in that AP.
no-communications-means	This error occurs if there is no shared communication protocol to reach the destination agent.
acc-unavailable	ACC failed to complete the action and it is unavailable
unable-to-deregister	The agent can not be deregistered. For example, it might have pending contracts, or because the agent is not found in the DF.
df-overloaded	This occurs because the DF fails to finish the operation because of processing resource overload.
inconsistency	An action is rejected due to some inconsistency in the original request.
agent-already-registered	This failure occurs if the agent to be registered is already in the DF or AMS
unauthorised	This occurs if the requesting agent is not sufficiently authorised.
ams-overloaded	This occurs because the AMS fails to finish the modification operation because of processing resource overload.

Annex A Agent Communication Channel Interface Description Language (Normative)

The following IDL specifies the agent interface which is intentionally minimal. The interface contains a single operation operation *message* which supplies a string containing the ACL message as a parameter. Future versions of FIPA agent specifications reserve the right to extend or modify this interface.

```
interface FIPA_Agent_97 {

      oneway void message(in string acl_message);

      };
```

Annex B ACC & FIPA Baseline Protocol (Informative)

Agent communication channel requirements

The FIPA ACC has the following preferred set of requirements :

1. The ACC must support asynchronous messaging
   - the ACC must not block upon receipt of the message
   - the ACC stores messages until they are received by the destination agent if storage capabilities are provided, if not an error message is sent back.
   - the ACC is required to provide agents with fair access to their messages (i.e. if an agent is able to receive a message it should have access to the messages available for it)
   - the ACC has a minimum policy on message storage which it is able to make known (e.g. time-out period)
   - messages may contain instructions on how it must be handled by the receiving ACC.
   - the sending ACC is able to acknowledge receipt of message from sending agent
2. The ACC must support basic exception reporting such as failure-to-deliver-message, agent-unavailable etc.
3. For a given sender/receiver pair the ACC will forward always messages in order of receipt (i.e. the ACC does not implement its own ordering policy).
4. The ACC will support device mobility where there is intermittent agent connectivity.
5. The ACC can optionally support agent mobility.
6. The ACC supports queries about the transport level protocols it supports.
7. The ACC can optionally support FIPA security services.
8. Where possible the ACC ensure that address fields are well-formed in the letter envelope.
9. In order for the sender to be able to reason about its communication. The ACC must support a `reasonable' semantics for message send.
10. An agent must be able to receive messages from a large number of different sources more-or-less simultaneously. This in turn implies some kind of minimal buffering supported by the ACC.

FIPA baseline protocol requirements

FIPA is committed to specifying a baseline protocol which supports interoperability between FIPA compliant AP's. This baseline protocol should be:

1. Open and widely available, including;
   - a comprehensive specification must be available in the public domain
   - software implementations must be available, preferably free and ideally more than one
   - both specification and software must be maintained
2. Accurate and reliable where;
   - messages are received in the form they were sent
   - best effort will be made to deliver well formed messages
3. Light-weight, minimising ;
   - the weight of encoding/decoding engine
   - the overhead of the protocol
   - development complexity (API)
4. Able to support basic error handling at the protocol level
   - such as time-out, message-undeliverable ...
5. Able to cross firewalls and be able to express firewall policy for the underlying protocol.
6. Extensible
   - enhancements can easily be made to the protocol
7. Able to run on a broad range of networks
   - GSM, IP etc.
8. Able to represent all well formed ACL messages.
9. Should be bit-wise efficient.

Annex C Use of IIOP (Informative)

Addressing the FIPA97 IIOP requirements.

Development of the FIPA97 mandated interoperability mechanism can be supported by a number of methods. These range from direct interaction with IIOP at the TCP/IP level to the use of CORBA support where all interaction with the IIOP protocol is hidden from the developer. These issues are discussed in detail in the FIPA97 Developers Guide.

The easiest way to implement the interoperability mechanism is through the use of a CORBA implementation. One compiles the IDL interface specified by FIPA97 i.e. the FIPA_Agent_97 interface into the implementation language of their choice, incoming messages will arrive as a parameter to the implementation of the message operation of the FIPA_Agent_97 interface. The implementor works at the method invocation level, no understanding of IIOP messages or how they should be handled is required. Applications built upon an IIOP compliant ORB are automatically IIOP compliant

Another way of leveraging available technology in order to implement the FIPA97 interoperability mechanism is to use an IIOP engine. An IIOP engine offers support for sending and receiving IIOP messages yet it is not an ORB, rather it can encode and decode IIOP messages and manage connections but unlike an ORB the implementor decides exactly how each IIOP message is handled. Consequently applications built using an IIOP engine are not necessarily IIOP compliant.

Of course it is possible to interact with IIOP at the TCP/IP protocol level. In this case it is up to the implementor to provide support for sending and receiving IIOP messages. Obviously applications built in this manner are not necessarily IIOP compliant.

Compliance with the IIOP specification is mandated in order to facilitate interworking between interoperability mechanisms built on different technologies for example an ORB and an IIOP engine. The interoperability mechanism built upon an IIOP engine must interwork fully with an interoperability mechanism built upon an ORB, in short all FIPA97 compliant mechanisms should behave as if they were built on an ORB.

Implications of Iiop

A key consideration in enabling the FIPA97 mechanism for inter agent communication is the distribution of IORs so that agents can invoke the ‘message’ method previously described on remote AP ACCs. IORs are often distributed through email, WWW pages, NFS file systems etc, unfortunately such a distribution mechanism is not suitable for FIPA agents because of the attendant overheads and its inherent lack of scalability. Another possibility is through the use of the CORBA naming service, specified by the OMG for exactly this kind of purpose and now available through many CORBA vendors.

It should be noted that IORs are already implicitly distributed through the FIPA agent naming convention. If one examines the FIPA address of an ACC one will note it is of the following form;

iiop://somewhere.com:50/acc

This address is sufficient to construct an agent IOR (there is a slight complication with object keys which will be explained below). The main components of an IOR are the Hostname (‘somewhere.com’), a port number on which the server is listening (‘50’) and an Object Key (‘acc’). These can be combined to form an IOR which can be used to invoke the ‘message’ operation on the necessary ACC.

As mentioned above, using this method of obtaining an IOR leads to a slight complication with the Object Key. This occurs because Object Keys are proprietary and are constructed by various ORB vendors in a proprietary manner, each object key will probably be a combination of Interface name and some sort of Marker or Server name, however these names can be mangled according to vendor policy. To understand the ramifications of this let us examine the server side.

If the ACC has been implemented through the use of an IIOP engine or through direct interaction with the IIOP protocol then there is no problem. This is because the server will be decoding IIOP requests for an object with the object key which has been distributed in its address e.g. 'acc', it merely has to recognise this object key and pass the request on to the required method/function to be handled, in short the server does not care what the object key is as long as it knows in advance what it should be, 'acc' is as good an object key as anything else.

This is not the case if one is using a ORB implementation. In this situation it is not user defined code which is decoding the requests and passing them on the appropriate objects/methods, rather it is the ORB which is doing this, and the ORB is subject to the proprietary Object Key mangling policy of the Vendor. Therefore, if one creates an interface object of Marker (or Server) name 'acc', within an ORBspace there is no reason to believe that its Object Key is going to be 'acc', in fact it is unlikely to be so. How therefore can one trap requests for Object Key 'acc' and forward them to the required Interface Object using an ORB implementation. This can be done by inserting some user defined code at the 'servant' level, that is the level in CORBA which accepts object invocations and forwards them on. In general this will have to be done in a proprietary method for each ORB implementation, luckily it is not difficult, for example using ORBIX one would use the Object Loader to create the required object once an Object Fault is generated. Furthermore, the OMGs new CORBA specification defines a portable method of doing this through the POA (Portable Object Adapter)[3].

The object key distributed in an AP's IIOP URL must be supported regardless of implementation technology constraints.
If you use proprietary technology it is important to understand the potential name conflicts between the IIOP URL distributed by your AP and the actual object key supported by this proprietary technology.

The Object Key interoperability issue is also currently an topic being addressed by the OMG. At the time of writing several proposals have been put forward to the OMG in response to their RFC about an extended Name Service [4]. The extensions include a solution to the issue of generating a IOR for a remote object (i.e. the ACC of a remote AP), and also a URL-like naming convention, which in most of the proposals is very similar (if not identical) to the FIPA iiop://host:port/path format. All of these proposals suggest a modification to the implementation of ORBs so that an extended initial call can be made to return the reference to a number of services without having to know any references to start with. The implementation of the solution will be handled by the ORB and is therefore not something that implementers of the FIPA AP must address themselves. The extensions will most likely make use of a 'special' reserved reference that is always available. More information is available in the individual proposals [5][6][7]. Ultimately, we believe a standard mechanism will be available for resolving URLs to IIOP IORs[11].

References

[1] Foundation for Intelligent Physical Agents, FIPA97 Specification Version 1.0 Part 1
[2] Foundation for Intelligent Physical Agents, FIPA97 Specification Version 1.0 Part 2 (section 5.2)
[3] Ross Mayne, Additions to CORBA on the Horizon - The Portable Object Adapter, Communicate, Volume 4 Issue 1, July 1998, pp 29-32
[4] Interoperability Name Service Enhancements, Draft version 1.2, OMG document orbos/97-12-20, December 1997 - http://www.omg.org/library/schedule/Interoperable_Name_Service_RFP.htm
[5] IONA/Nortel joint Interoperable Name Service RFP Initial Submission (orbos/98-03-03), March 1998
[6] Interoperable Naming Service Joint initial submission (orbos/98-03-04), March 1998
[7] Interoperable Naming Service (orbos/98-03-06), March 1998

[11] FIPA has issued a call for proposals for 1999 covering agent naming services which is likely to resolve these issues.