FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Agent Management Support for Mobility Specification

Document title	FIPA Agent Management Support for Mobility Specification		
Document number	PC000087A	**Document source**	FIPA Architecture Board
Document status	Preliminary	**Date of this status**	2000/06/30
Supersedes	None		
Contact	fab@fipa.org		
Change history			
2000/06/05	Carried forward from FIPA 1998 Specification 11 V1.0		
2000/06/30	Editorial changes; made `:code-base` parameter optional		

Geneva, Switzerland

Foreword

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

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

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

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

Contents

1 Scope

FIPA is concerned with two types of mobility; mobility in devices such as portable computers and Personal Digital Assistants (PDAs) that can be intermittently connected to the network, and mobility in software such as mobile agents that can move between hosts.

This specification is concerned with specifying the minimum requirements and technologies to allow agents to take advantage of mobility. This specification integrates closely with [FIPA00023] and provides a wrapping mechanism for existing mobile agent systems to promote interoperability. Therefore, the scope of this specification is limited to:

- This specification does not mandate the use of mobility features. Instead, it mandates how agents and APs may support mobility, if mobility is desired.

- This specification does not mandate the use of any explicit technology for supporting mobility. Instead, it provides a wrapping mechanism for mobile agent systems.

- This specification does not define how mobile agents and mobile agent systems operate or are implemented. However, the mobility capabilities defined in this specification rely on their existence.

- Mobile agent security is not currently addressed by this specification. This topic will be addressed in future versions of this specification.

- This specification defines extensions that are necessary to the AMS to support mobility.

- The platform profile can become a standard way for an agent to discover the type of mobility supported by an AP. If an AP does not support mobility, then it should refuse any mobility operation.

2 Agent Management Support for Mobility Reference Model

2.1 Protocols as a Metaphor for Expressing Mobility

It is recognised that there are many ways of expressing mobility within agents, such as code mobility, agent migration and agent cloning. For this reason, FIPA does not mandate a single form of agent mobility but supports a core set of actions that allow flexible and extensible forms of mobility protocols to be supported. Two example protocol abstractions are explained here:

- **Simple Mobility Protocols**
  An agent relies on a high level protocol that uses a single action (for example, *move*) which causes it to be moved to a destination AP. In this case, the AP upon which the agent is executing will have to implement the necessary protocol to realise the entire migration operation. This is illustrated in *Figure 1*, where an agent is delegating its mobility operation to the agent platform.

  The perceived advantages of the simple mobility protocols are that there is a reduced complexity in application agent development since mobility is supported by the AP, they are oriented towards existing mobile agent frameworks (for example, [OMGmaf]) and easy implementation on existing mobile agent platforms via FIPA ACL enhancement, and, there is a reduced number of remote interactions.

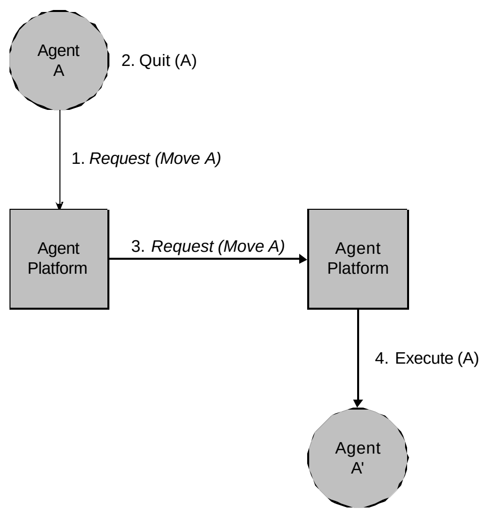

Figure 1: Example Simple Mobility Protocol

- **Full Mobility Protocols**
  An agent directs the mobility protocol itself and does not delegate responsibility to the AP. As shown in *Figure 2*, an agent first *move*s its agent code (and possibly state) to a destination AP and eventually *transfer*s its identity and authority once it is assured that the new agent has been created successfully. Note that the agent mobility operation is not deemed to be completed until both the agent code (and possibly state) and the agent identity have been successfully transferred. Additionally, this protocol also allows the agent to inform its HAP and any other APs that it has moved to a new location.

  The perceived advantages of full mobility protocols are that is a reduced complexity in AP implementation, there are enhanced capabilities for the application agent in controlling the mobility operation, and, it represents a more secure form of mobility.

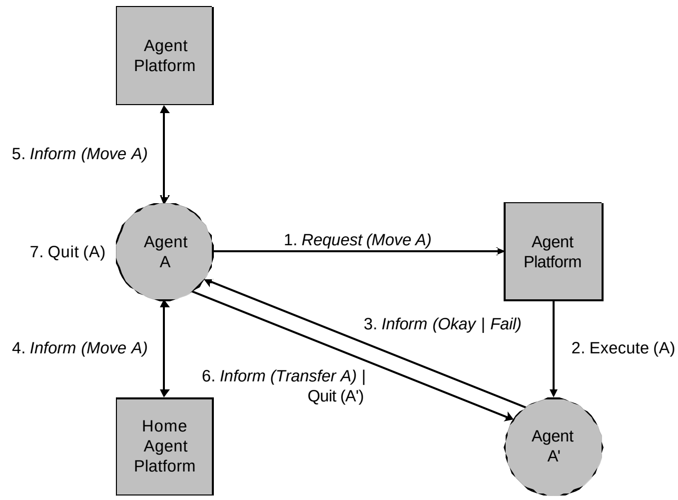

Figure 2: Example Full Mobility Protocol

It is expected that both of these protocols (and others) can be appropriate in different application contexts. Therefore, this specification expects that FIPA AP, that support mobility will implement either low level or high level mobility protocol, or both.

To initiate agent mobility (such as migration, cloning or invocation) with the *move* operation, the sending agent will identify the mobility protocol to be used for that mobility operation (see section *2.3, Mobility Protocols*). Using this information, the involved AMS and agents determine and take subsequent actions to complete the mobility operation which may involve the use of other operations, such as *transfer*.

2.2 Mobility Life Cycle

This specification extends the existing life cycle given in [FIPA00023] by adding a new state (**Transit**) and two new actions to enter and leave that state (**Move** and **Execute**). This allows the current state of the agent to be represented within the AMS. This new life cycle illustrated in *Figure 3*[1].

Only mobile agents can enter the **Transit** state, or to put it another way, stationary agents never 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:

- **Move**
  Puts the agent in a transitory state; this can only be initiated by the agent.

- **Execute**
  Brings the agent out of a transitory state; this can only be initiated by the agent system.

The relationship between the life cycle actions of Move and Execute can be associated with the Agent Management actions of **Move**, **Transfer** and **Execute** in the following way. To enter the **Transit** state, a mobile agent initiates the execution of a mobility protocol which involves sending a **Move** (and possibly a **Transfer** in the case of a full mobility protocol) to an AMS. Correspondingly, a mobile agent is brought out of the **Transit** state by an AMS issuing an execute action upon its code (see section *2.3, Mobility Protocols*).

[1] The **Execute** action is not specified here since it is an implementation issue.

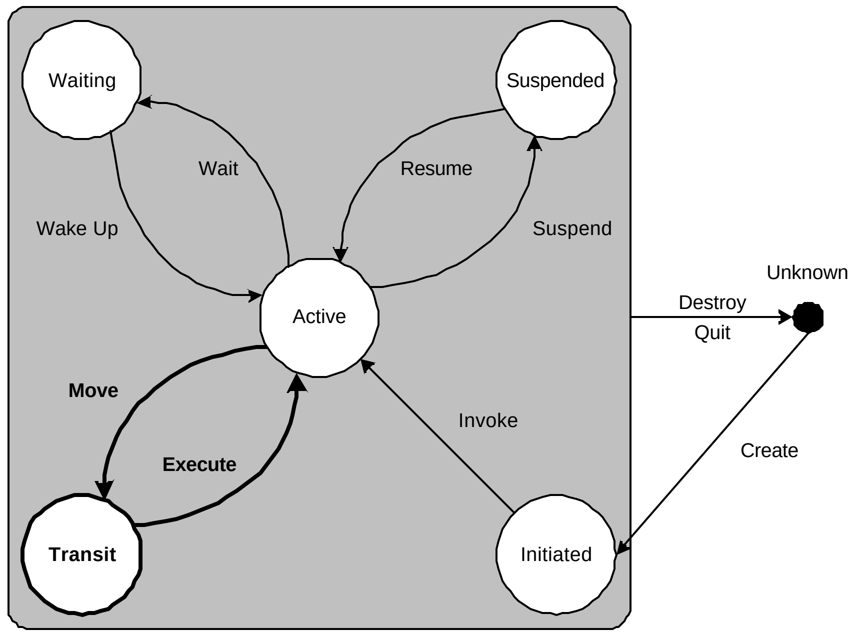

Figure 3: Mobile Agent Life Cycle

2.3 Mobility Protocols

A number of standard protocols have been defined to cover various forms of agent mobility. Specifically, they address:

- Agent migration,
- Agent cloning, and,
- Agent invocation.

As described in section *2.1, Protocols as a Metaphor for Expressing Mobility*, there are essentially two types of protocols; simple and full. The simple protocols base most of the functionality of the mobility operation within the local and remote APs; the full protocols spread the task across the mobile agent and the APs.

Figures 4 to 9 represent the three mobility operations for each type of protocol; when an agent wishes to move to another AP, it can specify one of these as a mobility protocol which describes the actions and reactions of each involved parties. Other protocols can be constructed from the actions given in section *3.2, Function Descriptions* to permit flexible and extensible forms of agent mobility.

2.3.1 Agent Migration

The agent migration protocols are invoked by agents that wish to transport themselves between two APs. The **simple migration protocol** (see *Figure 4*) requires that the migrating agent delegates all responsibility for the migration operation to the APs, who complete the task on its behalf. By comparison, the **full migration protocol** (see *Figure 5*) requires the agent to participate in the migration operation and to control aspects of its completion; the task is not completed until the transfer action has been approved.

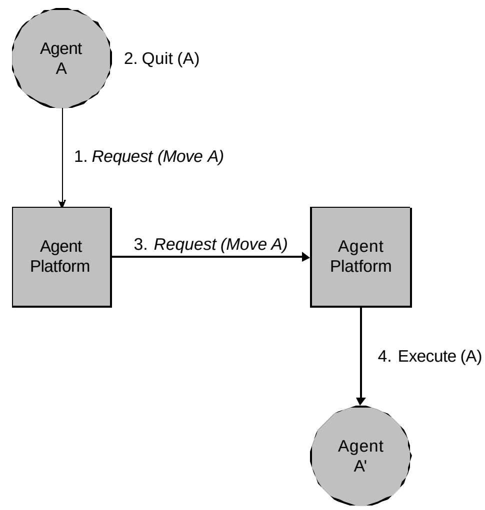

Figure 4: Simple Agent Migration Protocol

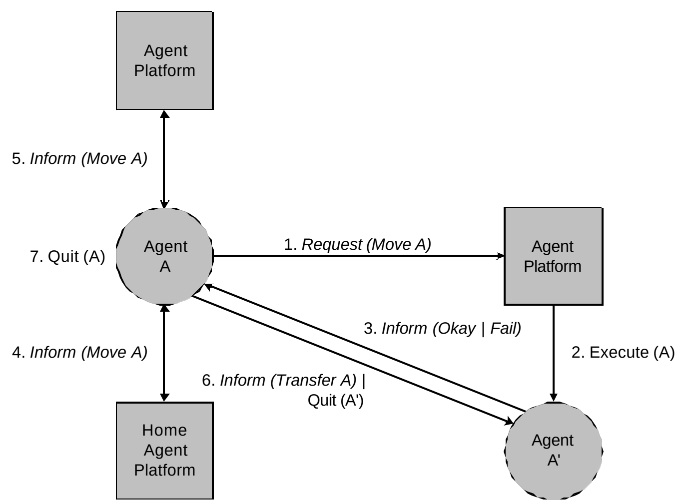

Figure 5: Full Agent Migration Protocol

2.3.2 Agent Cloning

The agent cloning protocols are invoked by agents that wish to create a copy of themselves on an AP. These protocols follow the same principles and responsibilities as agent migration (see *Figure 6* and *Figure 7*).

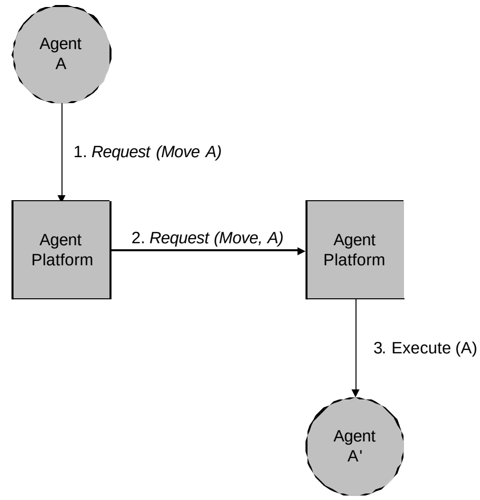

Figure 6: Simple Agent Cloning Protocol

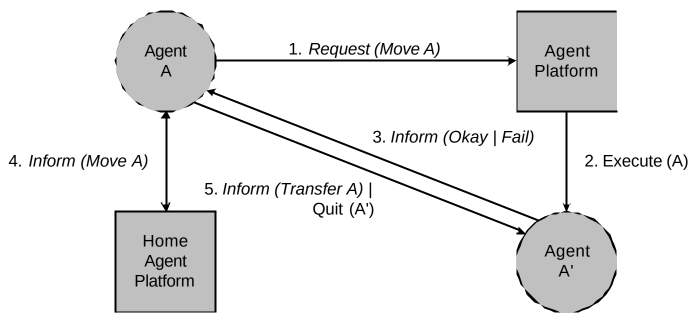

Figure 7: Full Agent Cloning Protocol

2.3.3 Agent Invocation

The agent invocation protocols are invoked by agents that wish to create an agent on an AP. These protocols follow the same principles and responsibilities as agent migration and agent cloning (see *Figure 8* and *Figure 9*).

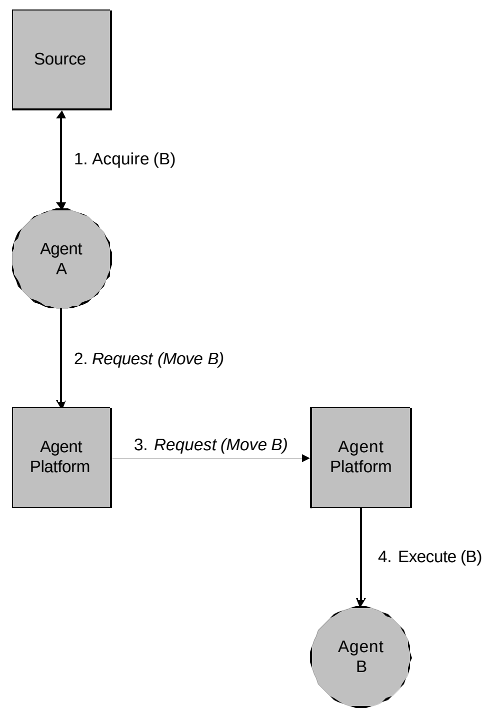

Figure 8: Simple Agent Invocation Protocol

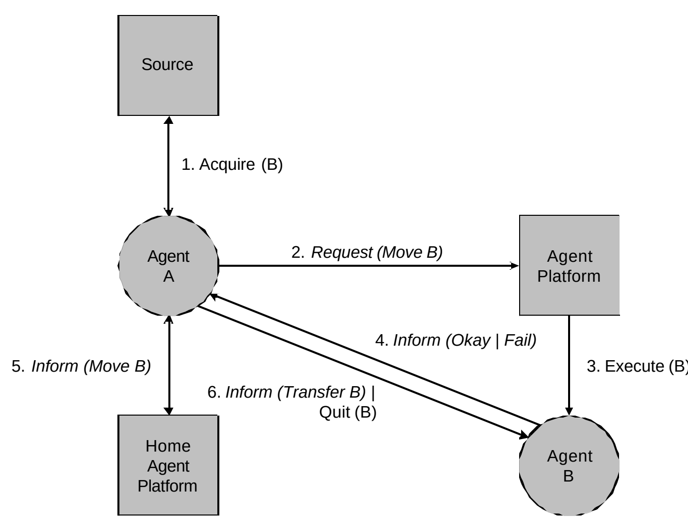

Figure 9: Full Agent Invocation Protocol

2.4 Agent Profiles

Since a mobile agent can be transported between APs in a variety of formats it can make a number of demands upon an AP for a required set of conditions to be met before such an agent can be executed. Some common examples of the form of a mobile agent might be:

- Written in Java (version 1.2) using the Aglets mobile agent toolkit (0.1 beta) represented as serialised byte-code,
- Written in C represented as native code compiled for Linux (version 2.2.15) on i386 hardware, or,
- Written in April (version 4.4) represented as byte-code.

Each of these dependencies can be expressed as part of the meta-information of a mobile agent within the `:profile` parameter (see section *3.1.2, Mobile Agent Profile*). This parameter contains three description sections which allow various characteristics of the mobile agent to be specified:

- `:system`
  Expresses requirements of the mobile agent system which the mobile agent uses (if any), such as Aglets, Mole, AgentTcl or Voyager (see section *3.1.3, Mobile Agent System*).
- `:language`
  Expresses requirements of the language in which the mobile agent is written, such as Java source code, i386 native code or April byte-code (see section *3.1.4, Mobile Agent Language*).
- `:os`
  Expresses requirements of the operating system for which the mobile agent was intended (if any), such as a Solaris SPARC box or a Linux i386 box (see section *3.1.5, Mobile Agent Operating System*).

This permits a great deal of flexibility in stating the execution requirements of a mobile agent and can be used by a receiving AP to determine whether it can support an agent of that type[2]. A particular deficiency in any stated profile description section may cause the agent to be rejected on the grounds of lack of support or for security reasons (`agent-profile-unsupported`).

Extra dependency information can be stated in the `:dependencies` parameter of each profile description section. This is a free-form parameter that may or may not be supported by an AP for that particular class of agent. For example, language dependencies may express additional class libraries required by the mobile agent and operating system dependencies may express additional software that should be installed on the OS (such as Perl, TCL/Tk, etc.).

[2] An AP defines this information in its platform profile as described in [FIPA00023].

3 Agent Mobility Ontology

This ontology represents extensions to the `FIPA-Agent-Management` ontology defined in [FIPA00023] if mobility is supported.

3.1 Object Descriptions

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

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

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

3.1.1 Mobile Agent Description

Frame **Ontology**	`mobile-agent-description` `FIPA-Agent-Management`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`name`	The identifier of the agent.	Mandatory	`agent-identifier`	
`profile`	A list of mobility requirements of the agent.	Optional	Set of `mobile-agent-profile`	
`version`	The version of the agent.	Optional	`String`	
`protocol`	A list of mobility protocols supported by the agent.	Optional	Set of `String`	
`code`	The code-base of the agent	Optional	`Byte-Stream`	
`data`	The dynamic data (state) of the agent.	Optional	`Byte-Stream`	

3.1.2 Mobile Agent Profile

Frame **Ontology**	`mobile-agent-profile` `FIPA-Agent-Management`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`system`	The mobile agent system environment supported by the agent.	Mandatory	`mobile-agent-system`	
`language`	The language environment supported by the agent.	Mandatory	`mobile-agent-language`	
`os`	The operating system environment supported by the agent.	Optional	`mobile-agent-os`	

3.1.3 Mobile Agent System

Frame **Ontology**	`mobile-agent-system` `FIPA-Agent-Management`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`name`	The name of the mobile agent system.	Mandatory	`String`	
`major-version`	The major version of the mobile agent system.	Mandatory	`String`	
`minor-version`	The minor version of the mobile agent system.	Optional	`String`	
`dependencies`	The dependencies required by the mobile agent system.	Optional	Set of `property`	

3.1.4 Mobile Agent Language

Frame **Ontology**	`mobile-agent-language` `FIPA-Agent-Management`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`name`	The name of the mobile agent language.	Mandatory	`String`	
`major-version`	The major version of the mobile agent language.	Mandatory	`String`	
`minor-version`	The minor version of the mobile agent language.	Optional	`String`	
`format`	The format of the code base of the mobile agent.	Mandatory	`String`	
`filter`	The filter that should be executed over the code base before execution.	Optional	`String`	
`dependencies`	The dependencies required by the mobile agent language.	Optional	Set of `property`	

3.1.5 Mobile Agent Operating System

Frame **Ontology**	`mobile-agent-os` `FIPA-Agent-Management`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`name`	The name of the operating system.	Mandatory	`String`	
`major-version`	The major version of the operating system.	Mandatory	`String`	
`minor-version`	The minor version of the operating system.	Optional	`String`	
`hardware`	The hardware of the operating system.	Optional	`String`	
`dependencies`	The dependencies required by the operating system.	Optional	Set of `property`	

3.2 Function Descriptions

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

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

- **Function**. This is the symbol that identifies the function in the ontology.
- **Ontology**. This is the name of the ontology, whose domain of discourse includes the function described in the table.
- **Supported by**. This is the type of agent that supports this function.
- **Description**. This is a natural language description of the semantics of the function.
- **Domain**. This indicates the domain over which the function is defined. The arguments passed to the function must belong to the set identified by the domain.
- **Range**. This indicates the range to which the function maps the symbols of the domain. The result of the function is a symbol belonging to the set identified by the range.
- **Arity**. This indicates the number of arguments that a function takes. If a function can take an arbitrary number of arguments, then its arity is undefined.

3.2.1 Migrate a Mobile Agent

Function	`move`
Ontology	`FIPA-Mobile-Agent-Management`
Supported by	AMS
Description	An agent issues a `move` request to transfer itself to a local/remote AMS. However, the AMS may refuse to accept the `move` request due to lack of agent profile support or other local restrictions.
Domain	`mobile-agent-description`
Range	The execution of this function results in a change of the state but it has no explicit result. Therefore there is no range set.
Arity	1

3.2.2 Transfer the Identity of a Mobile Agent

Function	`transfer`
Ontology	`FIPA-Mobile-Agent-Management`
Supported by	AMS
Description	An agent issues a `transfer` request to send its identity and authority to another agent on a destination AMS. However, the receiving agent may refuse to accept the `transfer` request for security reasons.
Domain	`mobile-agent-description`
Range	The execution of this function results in a change of the state but it has no explicit result. Therefore there is no range set.
Arity	1

3.3 Exceptions

These exceptions extend those defined in [FIPA00023].

3.3.1 Failure Exception Propositions

Communicative Act **Ontology**	`failure` `FIPA-Agent-Management`	
Predicate symbol	**Arguments**	**Description**
`mobility-unsupported`	`String`	The receiving AMS does not support agent mobility.

`profile-unsupported`	`String`	The receiving AMS does not support the specified mobility profile description.
`agent-already-present`	`String`	The receiving AMS already has an agent registered with the same name as the migrating agent.

4 Annex A — Integration of FIPA Agent Mobility and MAF

The intention of the Mobile Agent Facility (MAF - see [OMGmaf]) specification is to achieve a certain degree of interoperability between mobile agent platforms of different manufacturers. A MAF-compliant agent platform can be accessed via two standardised interfaces that are specified by means of the OMG's Interface Definition Language (IDL): `MAFAgentSystem` and `MAFFinder`. These interfaces provide fundamental operations for agent management, agent tracking and agent transport. Note that these interfaces represent the access point to agent systems and registration components; their concrete implementation is not specified.

Several similarities between a FIPA AP that supports agent mobility and a MAF-compliant AP can be drawn regarding their functionality:

- The FIPA AMS can be compared to a MAF agent system, represented by the `MAFAgentSystem` interface; both are responsible for the management of agents,
- The FIPA DF is similar to the MAF registration component, represented by the `MAFFinder` interface; the task of these entities is the maintenance of registration information about agents in a distributed environment,
- The equivalent of the Message Transport System (MTS - see [FIPA00067]) is the Object Request Broker (ORB) in the context of MAF; these entities care for the transfer of messages in a distributed agent environment, and,
- FIPA and MAF provide their specifications in an implementation-independent way.

Beside these similarities, several differences have to be mentioned which are mainly associated with the general design approach of the FIPA specifications and the MAF specification:

- FIPA standards try to cover the set of functionality that is required for the execution and support of mobile agents by means of a high-level speech act language, the ACL, as well as appropriate content languages. ACL allows for the specification of operations and high-level communication protocols, and,
- The MAF specification covers a minimal set of functionality since it is meant as an add-on to existing agent platforms rather than as the basis for completely new systems. The functionality of a MAF-compliant platform is accessible via IDL interfaces. These interfaces provide, among others, methods for the management (that is, creation, suspension, resumption and termination), transport and tracking of agents. In contrast to FIPA, no high-level language is used above the IDL methods. Instead, each IDL method is directly mapped onto a method of the associated, implemented object.

Regarding these characteristics of FIPA and MAF, the two standardisation approaches can be combined to a unified mobile agent framework. One promising way seems to be the integration of the IDL operation(s) defined in FIPA for the transfer of ACL messages into the MAF IDL specifications (see *Figure 10*). To realise an agent platform that is FIPA- and MAF-compliant, the following three possibilities exist:

- The existing MAF interfaces `MAFAgentSystem` and `MAFFinder` can be enhanced by new operations that enable a FIPA-compliant platform access,
- The existing operations of the MAF interfaces can be modified in order to adapt them to the requirements of the FIPA specifications, and,
- Completely new interfaces are specified additionally to the existing MAF interfaces.

While the first two approaches require modification of the existing MAF specification, the third approach can be regarded as a pure extension that does not require any changes.

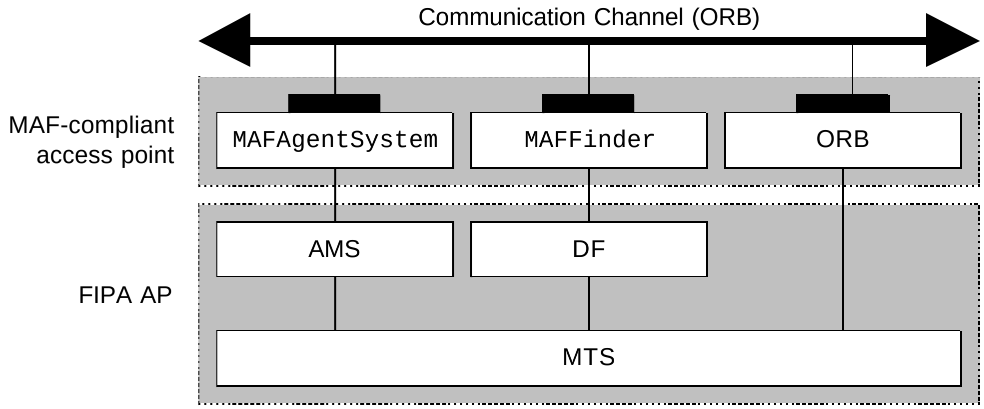

Figure 10: Integration of FIPA and MAF

However, the FIPA specifications could be enhanced by some "specialised" methods as defined in the MAF specification. This could be desirable for methods that have a simple parameter structure and that can be sufficiently represented without using a high-level content language.

5 References

[FIPA00023] FIPA Agent Management Specification. Foundation for Intelligent Physical Agents, 2000.
`http://www.fipa.org/specs/fipa00023/`

[FIPA00067] FIPA Agent Message Transport Service Specification. Foundation for Intelligent Physical Agents, 2000.
`http://www.fipa.org/specs/fipa00067/`

[OMGmaf] OMG Mobile Agent Facility Specification. Object Management Group, 2000.
`http://www.omg.org/cgi-bin/doc?formal/00-01-02.pdf`