FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Nomadic Application Support Specification

Document title	FIPA Nomadic Application Support Specification		
Document number	SI00014H	**Document source**	FIPA TC Nomadic Application Support
Document status	Standard	**Date of this status**	2002/12/03
Supersedes	FIPA00062, FIPA00063, FIPA00065, FIPA00066		
Contact	fab@fipa.org		
Change history	See *Informative Annex A — ChangeLog*		

http://www.fipa.org/
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 where appropriate.

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 Document Policy [f-out-00000] and the FIPA Specifications Policy [f-out-00003]. A complete overview of the FIPA specifications and their current status may be found on the FIPA Web site.

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

Contents

1 Scope

This document is part of the FIPA specifications and deals with agent middleware to support applications in nomadic environment. The environment of mobile computing is very different compared to today's environment of traditional distributed systems in many respects. Bandwidth, latency, delay, error rate, interference, interoperability, computing power, quality of display, among other things may change dramatically as a nomadic end-user moves from one location to another. All these cause new demands for adaptability of data services.

Adaptability to the changes in the environment of nomadic end-users is an important issue. A nomadic end-user confronted with these circumstances would benefit from having the following functionality provided by the infrastructure: information about expected performance, agents controlling over the transfer operations, a condition-based control policy, capability provided by agents to work in a disconnected mode, advanced error recovery methods, and adaptability.

This specification gives an overview of the nomadic application support area and contains informative specifications for:

- Monitor Agent (MA) functionality, and

- Control Agent (CA) functionality.

In addition, three other FIPA specifications are related to nomadic application support: [FIPA00069], [FIPA00088] and [FIPA00094].

2 General Analysis

2.1 Overview

The results of current developments in both wireless data communications and mobile computers are being combined to facilitate a new trend: *nomadic computing*. Compared to today's traditional distributed systems, the nomadic computing environment is very different in many respects. Bandwidth, latency, delay, error rate, quality of display and other non-functional parameters may change dramatically when a nomadic end-user moves from one location to another and thus from one computing environment to another, for example, from a wire line LAN to a UMTS network. The variety of mobile workstations, handheld devices and smart phones, which allow nomadic end-users to access Internet services, is increasing rapidly. The capabilities of mobile devices range from very low performance equipment (such as PDAs) up to high performance laptop PCs. All these devices create new demands for adaptability of Internet services. For example, PDAs cannot display properly high quality images and as nomadic end-users may be charged based on the amount of data transmitted over the GPRS-UMTS network, they may have to pay for bits that are totally useless to them.

Confronted with these circumstances, the nomadic end-user would benefit from having the following functionality provided by the infrastructure: information about expected performance, agent monitoring and controlling the transfer operations, and adaptability.

The ability to automatically adjust to changes in a transparent and integrated fashion is essential for *nomadicity*; nomadic end-users are usually professionals in areas other than computing. Furthermore, today's mobile computer systems are already very complex to use as productivity tools. Thus, nomadic end-users need all the support that a FIPA agent-based distributed system can deliver and adaptability to the changes in the environment of nomadic end-users is an important issue.

The adaptation of applications to various nomadic computing environments is an important area. There are several tasks that agents need to carry out during application adaptation:

1. Selection of Message Transport Protocol (MTP) and Message Transport Connection (MTC) to be used for agent communication.

2. Selection of an ACL and content language representation to be used for agent communication.

3. Provision of support for application agents to carry out adaptation of application data, such as still images, video and audio, XML, etc. Today's Internet application data (such as multimedia content) are designed with high performance desktop PCs and high quality displays in mind. Therefore, the application data is frequently unsuitable for nomadic computing using wireless wide-area networks and low performance mobile devices, and hence requires modification.

4. Communication between agents performing adaptation.

The FIPA Nomadic Application Support specifications define agent middleware to monitor and control an MTP and the underlying MTC. In addition, this specification gives examples of the use of the above scenarios.

2.2 Monitoring and Controlling Quality of Service

The functions required to carry out monitoring and controlling for Quality of Service (QoS) can be split into several specific tasks:

1. Observing the QoS of MTPs and MTCs,

2. Measuring (if there are no other means to obtain the required information) the QoS of an MTP and MTC,

3. Collecting information from the observing and measuring sources,

4. Analysing the information, and,

5. Controlling an MTC and selecting an MTP.

Based on this division, the agent middleware consists of the following logical agents (see *Figure 1*):

- A MA which carries out tasks 1 through 4, and,

- A CA which carries out task 5.

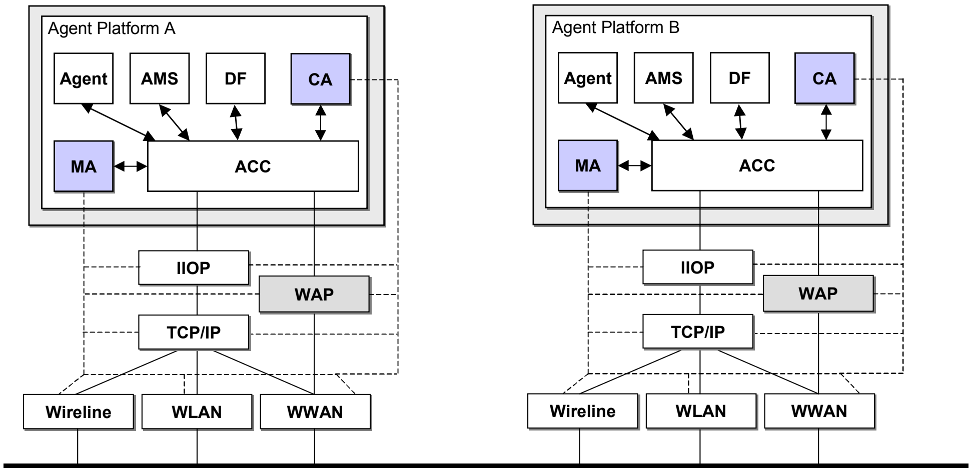

Figure 1: Reference Model of Agent based Adaptation

The most appropriate configuration of MAs and CAs is that there is at least one pair in each AP involving adaptation. The MA may measure the actual QoS of an MTC, if the network running an MTC does not provide users with required performance data[1].

An MA may:

- Consist of network-service-specific components that collect raw performance data at fixed intervals,

- Provide a repository for the measurement data collected,

- Perform first level analysis of the collected data, and,

- Send the results of the analysis to CA, if requested to do so.

A CA may:

- Manage (establish, close, suspend, activate, etc.) an MTC[2].

In some cases there is a need for MAs and CAs in heterogeneous APs to communicate with each other; therefore, interaction protocols and ontologies to achieve this are specified in this document.

---

[1] The way this actual measurement is performed is not a subject of standardisation within FIPA.

[2] The way that management actions are executed is not a subject of standardisation within FIPA.

2.3 Negotiation of Message Transport Requirements

There are several mechanisms that can determine the MTP, message representation and content language to use between communicating entities:

- Communicating entities know a peer entity's preferences beforehand and use them.

- The activating entity tries to use a method and if the peer entity is not capable of using the suggested method, then the activating entity may try another one (and so on).

- The communicating entities negotiate about a method to be used.

2.3.1 Negotiation about Message Transport Protocols

Previous FIPA specifications have implicitly assumed that the MTC is operational all the time (meaning that the MTC has been established before the agent message exchange and that it is reliable). However, this is not always the case within a nomadic environment.

A CA can activate the selection of an MTP or an agent can propose an MTP to a CA and it is the responsibility of the CA to either accept or reject the proposal based on whether it is possible to use the proposed MTP. CAs negotiate with peer CAs to use proposed MTPs which is illustrated in *Figure 2*.

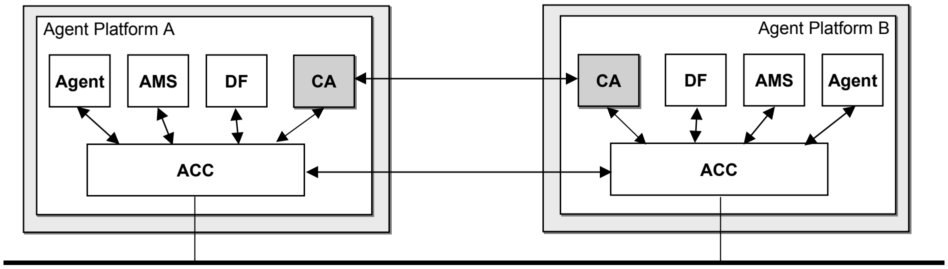

Figure 2: Control Agents Negotiating About a Message Transport Protocol

CAs use the `fipa-propose` interaction protocol [FIPA00036] and the `use` action to negotiate about an MTP. An example negotiation is given in Section 5.2.

2.3.2 Negotiation about Message Representation

In the environment of nomadic applications, it may be necessary to switch from one ACL representation to another; for example, when a mobile host roams from a wire line network to a wireless network. Application agents may use the `fipa-propose` interaction protocol and the `use` action to negotiate about the representation of ACL. Examples of this negotiation are given in Section 5.3.

3 Nomadic Application Support Ontology

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-nas` ontology. The `fipa-nas` ontology extends the `fipa-qos` ontology defined in [FIPA00094].

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 Transport Protocol Selection

This type of object represents a selection of transport protocol.

Frame	`transports`
Ontology	`fipa-nas`

Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`send`	A list of transport protocols supported for sending messages.	Mandatory	Sequence of `transport-protocol`[3]	
`recv`	A list of transport protocols supported for receiving messages.	Mandatory	Sequence of `transport-protocol`	

3.1.2 Message Representation Description

This type of object represents an ACL message representation.

Frame	`msg-representation`
Ontology	`fipa-nas`

Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`name`	The name of the message representation.	Mandatory	`word`	
`options`	A list of parameters for the message representation.	Optional	Set of `property`[4]	

---

[3] See [FIPA00094].

[4] See [FIPA00023].

3.1.3 Message Representation Selection

This type of object represents a selection of message representations.

Frame **Ontology**	`msg-encoding` `fipa-nas`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`send`	A list of message representations supported for sending messages.	Mandatory	Sequence of `msg-representation`	
`recv`	A list of message representations supported for receiving messages.	Mandatory	Sequence of `msg-representation`	

3.2 Function and Predicate Descriptions

The following tables define usage and semantics of the functions and the predicates that are part of the `fipa-nas` ontology.

The following terms are used to describe the functions of the `fipa-nas` domain:

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

- **Predicate**. This is the symbol that identifies the predicate in the ontology.

- **Ontology**. This is the name of the ontology, whose domain of discourse includes the function or the predicate described in the table.

- **Supported by**. This is the type of agent that supports this function or predicate.

- **Description**. This is a natural language description of the semantics of the function or the predicate.

- **Domain**. This indicates the domain over which the function predicate is defined. The arguments passed to the function or predicate 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 or a predicate takes. If a function or a predicate can take an arbitrary number of arguments, then its arity is undefined.

3.2.1 Transport Selection

Predicate	`transport-selection`
Ontology	`fipa-nas`
Supported by	CA
Description	An agent specifies the transport protocols that it is willing to use. The predicate is true, when the values of the `transports` parameter contain the transport protocol descriptions that the agent is willing to use. Otherwise, the predicate is false
Domain	`transports`
Arity	1

3.2.2 Message Encoding Selection

Predicate	`msg-encoding-selection`
Ontology	`fipa-nas`
Supported by	CA
Description	An agent specifies the message encoding choices that it is willing to use. The predicate is true, when the values of the `msg-encoding` parameter contain the message encoding choices that the agent is willing to use. Otherwise, the predicate is false
Domain	`msg-encoding`
Arity	1

3.2.3 Open Communication Channel

Function	`open-comm-channel`

Ontology	`fipa-nas`
Supported by	CA
Description	An agent can request that a CA open a communication channel. The communication channel description should contain enough information for a CA to be able to choose the right communication channel, that is, either the `name` parameter or the `target-addr` parameter must be present. The agent may also supply additional communication channel information by using the `options` parameter.
Domain	`comm-channel`
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

3.2.4 Close Communication Channel

Function	`close-comm-channel`
Ontology	`fipa-nas`
Supported by	CA
Description	An agent can request that a CA close a communication channel. The communication channel description should contain enough information for a CA to be able to choose the right communication channel, that is, either the `name` parameter or the `target-addr` parameter must be present.
Domain	`comm-channel`
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

3.2.5 Activate a Message Transport Protocol

Function	`activate`
Ontology	`fipa-nas`
Supported by	CA
Description	An agent can request that a CA activate a Message Transport Protocol (MTP). The transport protocol description should contain enough information to allow the CA to identify the correct transport protocol. Additionally, the agent may supply address information to where the transport protocol connection should be opened. It is possible to give the address of the gateway and/or the address of the destination AP. If the action is successful, the CA will return the object description of activated MTP.
Domain	Sequence of `transport-protocol`
Range	`transport-protocol`
Arity	1

3.2.6 Deactivate a Message Transport Protocol

Function	`deactivate`
Ontology	`fipa-nas`
Supported by	CA
Description	An agent can request that a CA deactivate an MTP.
Domain	`transport-protocol`
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

3.2.7 Select a Message Transport Protocol

Function	`use`
Ontology	`fipa-nas`
Supported by	CA
Description	An CA can request another CA to select an MTP or message encoding for use between Agent Communication Channels (ACCs). The requesting CA shall provide enough information to establish a working MTP connection or message encoding. The direction of communication (either send, receive or both) and the list of choices must be present. The list of choices is an ordered list where the highest priority is the first item and the lowest priority is the last item in the list. The receiving CA shall select at most one choice for the proposed direction of communication (either send, receive or both)
Domain	`transports` /[5] `msg-encoding`
Range	`transport-selection` /[6] `msg-encoding-selection`
Arity	1

3.3 Exceptions

The exceptions for the `fipa-nas` ontology follow the same form and rules as specified in [FIPA00023].

3.3.1 Not Understood Exception Predicates

Communicative Act	`not-understood`
Ontology	`fipa-nas`

Predicate Symbol	**Arguments**	**Description**
`unsupported-act`	`string`	The receiving agent does not support the specific communicative act; the string identifies the unsupported communicative act.
`unexpected-act`	`string`	The receiving agent supports the specified communicative act, but it is out of context; the string identifies the unexpected communicative act.
`unsupported-value`	`string`	The receiving agent does not support the value of a message parameter; the string identifies the message parameter name.
`unrecognised-value`	`string`	The receiving agent cannot recognise the value of a message parameter; the string identifies the message parameter name.

3.3.2 Refusal Exception Predicates

Communicative Act	`refuse`
Ontology	`fipa-nas`

Predicate symbol	**Arguments**	**Description**
`unauthorised`		The sending agent is not authorised to perform the function.
`unsupported-function`	`string`	The receiving agent does not support the function; the string identifies the unsupported function name.
`missing-argument`	`string`	A mandatory function argument is missing; the string identifies the missing function argument name.

---

[5] Where '/' is "exclusive or".

[6] Where '/' is "exclusive or".

`unexpected-argument`	`string`	A mandatory function argument is present which is not required; the string identifies the function argument that is not expected.
`unexpected-argument-count`		The number of function arguments is incorrect.
`missing-parameter`	`string string`	A mandatory parameter is missing; the first string represents the object name and the second string represents the missing parameter name.
`unexpected-parameter`	`string string`	The receiving agent does not support the parameter; the first string represents the function name and the second string represents the unsupported parameter name.
`unrecognised-parameter-value`	`string string`	The receiving agent cannot recognise the value of a parameter; the first string represents the object name and the second string represents the parameter name of the unrecognised parameter value.
`already-open`	`string`	The specified communication channel is already open; the string identifies the communication channel.
`not-open`	`string`	The specified communication channel is not open; the string identifies the communication channel.
`already-activated`	`string`	The specified transport protocol is already activated; the string identifies the transport protocol.
`not-active`	`string`	The specified transport protocol is not active; the string identifies the transport protocol.
`unrecognised-comm-channel`	`string`	The specified communication channel is not recognised; the string identifies the communication channel.
`unsupported-protocol`	`string`	The specified transport protocol is not supported; the string identifies the transport protocol.

3.3.3 Failure Exception Propositions

Communicative Act	`failure`
Ontology	`fipa-nas`

Predicate symbol	**Arguments**	**Description**
`internal-error`	`string`	An internal error occurred; the string identifies the internal error.
`open-failed`	`string`	The opening of a communication channel failed; the string identifies the failure reason.
`transient-failed`	`string`	The opening/closing of a communication channel or the activation/deactivation of a transport protocol failed; the string identifies the failure reason.
`close-failed`	`string`	The closing of a communication channel failed; the string identifies the failure reason.
`activation-failed`	`string`	The activation of a transport protocol failed; the string identifies the failure reason.
`deactivation-failed`	`string`	The deactivation of a transport protocol failed; the string identifies the failure reason.

4 Registration with the Directory Facilitator

In order for a CA and MA to advertise its willingness to provide its services to an agent domain, it must register with a DF (as described in [FIPA00023]. As part of this registration process, the following of constant values are introduced that universally identify the services the agent provides:

- The `name` parameter in `service-description` frame of a CA must be declared as a constant `fipa-mts-control`.

- The `type` parameter in `service-description` frame of a CA must be declared as a constant `fipa-ca`.

- The `ontology` parameter in `service-description` frame of a CA should be declared as a constant `fipa-nas`.

- The `type` parameter in `service-description` frame of a MA must be declared as a constant `fipa-mts-monitor`.

- The `type` parameter in `service-description` frame of a MA must be declared as a constant `fipa-ma`.

- The `ontology` parameter in `service-description` frame of a MA should be declared as a constant `fipa-qos`.

Below is given an example content of a `df-agent-description` frame which provides both MA and CA functionality:

```
(df-agent-description
  :name
    (agent-identifier
      :name monitor&control_agent@iiop://foo.com/acc
      :addresses (sequence iiop://foo.com/acc))
  :protocols (set fipa-request fipa-propose)
  :ontology (set fipa-nas)
  :language (set fipa-sl)
  :services (set
    (service-description
      :name fipa-mts-control
      :type fipa-ca
      :ontology fipa-nas)
    (service-description
      :name fipa-mts-monitor
      :type fipa-ma
      :ontology fipa-qos))
  :ownership (set Sonera)))))
```

5 Examples

5.1 Registration with a Directory Facilitator

1. A CA registers with a DF (see [FIPA00023]):

```
(request
  :sender
    (agent-identifier
      :name ca@foo.com
      :addresses (sequence http://foo.com/acc))
  :receiver (set
    (agent-identifier
      :name df@foo.com
      :addresses (sequence http://foo.com/acc)))
  :language fipa-sl
  :protocol fipa-request
  :ontology fipa-agent-management
  :content "(
    (action
      (agent-identifier
        :name df@foo.com
        :addresses (sequence http://foo.com/acc))
      (register
        (df-agent-description
          :name
            (agent-identifier
              :name ca@foo.com
              :addresses (sequence http://foo.com/acc))
          :services (set
            (service-description
              :name fipa-mts-control
              :type fipa-ca
              :ontology (set fipa-nas))))))))")
```

2. An MA registers with a DF.

```
(request
  :sender
    (agent-identifier
      :name ma@foo.com
      :addresses (sequence http://foo.com/acc))
  :receiver (set
    (agent-identifier
      :name df@foo.com
      :addresses (sequence http://foo.com/acc)))
  :language fipa-sl
  :protocol fipa-request
  :ontology fipa-agent-management
  :content " (
    (action
      (agent-identifier
        :name df@foo.com
        :addresses (sequence http://foo.com/acc))
      (register
        (df-agent-description
          :name
            (agent-identifier
              :name ma@foo.com
              :addresses (sequence http://foo.com/acc))
          :services (set
            (service-description
              :name fipa-mts-monitor
              :type fipa-ma
              :ontology (set fipa-nas))))))))")
```

5.2 Negotiating Message Transport Protocols

This example shows a scenario, where an application agent requests the use of either the WAP MTP [FIPA00076] or a proprietary MTP (for example, `x.uh.mdcp`). The message flow of a successful negotiation is illustrated in *Figure 3*.

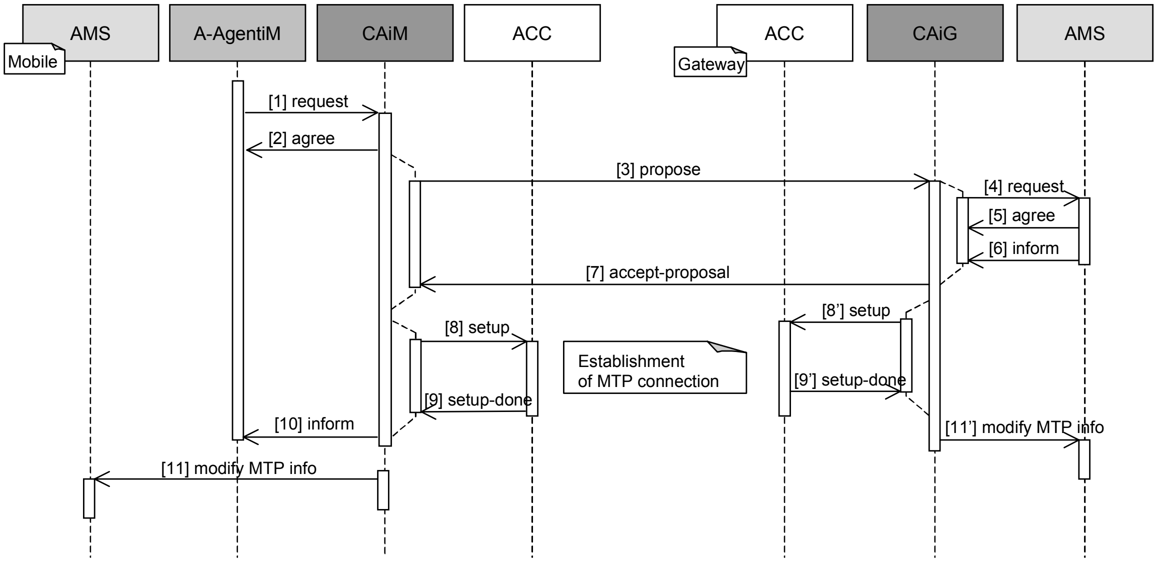

Figure 3: Flow of Message Transport Protocol Negotiation

1. Message 1 `request`: An application agent issues a request to the CA to activate either the `fipa.mts.mtp.wap.std` or `x.uh.mdcp` MTPs.

   ```
   (request
     :sender
       (agent-identifier
         :name A-AgentiM@mobile.com[7])
     :receiver (set
       (agent-identifier
          :name CaiM@mobile.com))
     :ontology fipa-nas
     :language fipa-sl
     :protocol fipa-request
     :content "(
       (action
         (agent-identifier
           :name CAiM@mobile.com)
         (activate (sequence
           (transport-protocol
             :name x.uh.mdcp)
           (transport-protocol
             :name fipa.mts.mtp.wap.std
             :dest-addr wap://gateway.com:1234/acc)))))")
   ```

---

[7] In all of the examples in this specification, the suffix of `iM` in an agent's name represents a mobile host, that is, an agent that is located on a mobile AP. Similarly, the suffix `iG` represents a gateway host and the suffix `iF` represents a fixed network host.

2. Message 2 `agree`: The CA agrees to activate an MTP. The decision to agree or disagree to activate an MTP might be based on the internal state of the CA (that is, the CA knows whether a requested MTP can be activated or not) or the CA might ask for an AP description from an AMS.

```
(agree
  :sender
    (agent-identifier
      :name CAiM@mobile.com)
  :receiver (set
    (agent-identifier
      :name A-AgentiM@mobile.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-request
  :content "(
    (action
      (agent-identifier
        :name CAiM@mobile.com)
     (activate (sequence
       (transport-protocol
         :name x.uh.mdcp)
       (transport-protocol
         :name fipa.mts.mtp.wap.std
         :dest-addr wap://gateway.com:1234/acc))))
    true))")
```

3. Message 3 `propose`: The CA in the mobile host proposes to its peer CA in the gateway host that either the `fipa.mts.mtp.wap.std` or `x.uh.mdcp` MTPs should be used in communication between the APs.

```
<?xml version="1.0"?>[8]

<envelope>

  <params index="1">

    <to>
      <agent-identifier>
        <name>CAiG@gateway.com</name>
      </agent-identifier>
    </to>
    <from>
      <agent-identifier>
        <name>CAiM@mobile.com</name>
      </agent-identifier>
    </from>

    <acl-representation>fipa.acl.rep.string.std</acl-representation>

    <date>20000606T100900000</date>

  </params>

</envelope>

(propose
  :sender
    (agent-identifier
      :name CAiM@mobile.com)
  :receiver (set
    (agent-identifier
```

---

[8] In most of the examples, the `envelope` part has been omitted for clarity.

```
          :name CAiG@gateway.com))
      :ontology fipa-nas
      :language fipa-sl
      :protocol fipa-propose
      :content "(
        (action
          (agent-identifier
            :name CAiM@mobile.com)
          (use
            (transports
              :send (sequence
                (transport-protocol
                  :name x.uh.mdcp)
                (transport-protocol
                  :name fipa.mts.mtp.wap.std))
              :recv (sequence
                (transport-protocol
                  :name x.uh.mdcp)
                (transport-protocol
                  :name fipa.mts.mtp.wap.std)))))
        true)")
```

4. Message 4 `request`, message 5 `agree` and message 6 `inform`: The CA in the gateway host requests the AP description from the local AMS (see [FIPA00023]) to determine whether the `x.uh.mdcp` or `fipa.mts.mtp.wap.std` MTPs are supported. The AMS informs the CA that both MTPs are supported and the CA decides to use `fipa.mts.mtp.wap.std` MTP based on the current QoS requirements of the MTC.

```
(request
  :sender
    (agent-identifier
      :name CAiG@gateway.com)
  :receiver (set
    (agent-identifier
      :name ams@gateway.com))
  :ontology fipa-agent-management
  :language fipa-sl
  :protocol fipa-request
  :content "(
    (action
      (agent-identifier
        :name ams@gateway.com)
      get-description))")

(agree
  :sender
    (agent-identifier
      :name ams@gateway.com)
  :receiver (set
    (agent-identifier
      :name CAiG@gateway.com))
  :ontology fipa-agent-management
  :language fipa-sl
  :protocol fipa-request
  :content "(
    (action
      (agent-identifier
        :name ams@gateway.com)
      get-description)
    true)")

(inform
  :sender
    (agent-identifier
      :name ams@gateway.com
      :addresses (sequence http://gateway.com/acc))
  :receiver (set
    (agent-identifier
      :name CAiG@gateway.com
      :addresses (sequence http://gateway.com/acc)))
  :ontology fipa-agent-management
  :language fipa-sl
  :protocol fipa-request
  :content "(
    (result
      (action
        (agent-identifier :name ams@gateway.com)
        get-description)
      (ap-description
        :name sonera-platform
        :transport-profile
       (ap-transport-description
         :available-mtps
           (set
             (mtp-description
                 :profile fipa.profile.mts.alpha
                 :mtp-name fipa.mts.mtp.iiop.std
                 :addresses (sequence iiop://gateway.com/acc))
             (mtp-description
                 :profile fipa.profile.mts.beta
                 :mtp-name fipa.mts.mtp.wap.std
                 :addresses (sequence wap://gateway.com:1234/acc))
             (mtp-description
                 :profile x.uh.profile
                 :mtp-name x.uh.mdcp
                 :addresses (set mdcp://gateway.com/acc)))))))")
```

5. Message 7 `accept-proposal`: The CA in the gateway host accepts the proposal to use the `fipa.mts.mtp.wap.std` MTP and sends the response to the CA in the mobile host informing it about the preferred MTP.

```
(accept-proposal
  :sender
    (agent-identifier
      :name CAiG@gateway.com)
  :receiver (set
    (agent-identifier
      :name CAiM@mobile.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-propose
  :content "(
    (action
      (agent-identifier
        :name CAiM@mobile.com)
      (use
        (transports
          :send (sequence
            (transport-protocol
              :name x.uh.mdcp)
            (transport-protocol
              :name fipa.mts.mtp.wap.std))
          :recv (sequence
            (transport-protocol
              :name x.uh.mdcp)
            (transport-protocol
              :name fipa.mts.mtp.wap.std)))))
    (transport-selection
      (transports
        :send (sequence
          (transport-protocol
            :name fipa.mts.mtp.wap.std))
        :recv (sequence
          (transport-protocol
            :name fipa.mts.mtp.wap.std)))))")
```

6. Messages 8 and 8' `setup`: The CAs request their respective ACCs to setup the `fipa.mts.mtp.wap.std` MTP. This is an implementation issue.

7. Message 9 and 9' `setup-done`: The ACCs inform their respective CAs that the `fipa.mts.mtp.wap.std` MTP has been established between the mobile host and the gateway host.

8. Message 10 `inform`: The CA informs the application agent that the MTC is established.

```
(inform
  :sender
    (agent-identifier
      :name CAiM@mobile.com)
  :receiver (set
    (agent-identifier
      :name A-AgentiM@mobile.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-request
  :content "(
    (result
      (action
        (agent-identifier
          :name CaiM@mobile.com)
      (activate (sequence
        (transport-protocol
          :name x.uh.mdcp)
        (transport-protocol
          :name fipa.mts.mtp.wap.std
          :dest-addr wap://gateway.com:1234/acc))))
    (transport-protocol
      :name fipa.mts.mtp.wap.std)))")
```

9. Message 11 and 11' `set-description`: `CAiM` (/`CAiG`) modifies the AP description to show that the `fipa.mts.mtp.wap.std` is now active.

5.3 Negotiating Message Representations

This example shows a scenario where an application agent in a mobile host proposes to its peer application agent in a fixed host the use of the `fipa.acl.rep.bitefficient.std` representation of ACL [FIPA00069] for their communication. The message flow is illustrated in *Figure 4*.

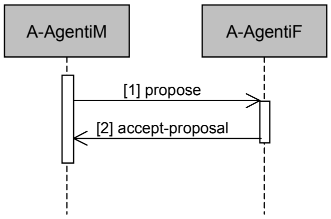

Figure 4: Flow of Message Representation Negotiation

1. Message 1 `propose`: The agent in the mobile host proposes the use of the `fipa.acl.rep.bitefficient.std` representation of ACL.

```
(propose
  :sender
    (agent-identifier
      :name A-AgentiM@mobile.com)
  :receiver (set
    (agent-identifier
      :name A-AgentiF@fixed.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-propose
  :content "(
    (action
      (agent-identifier
        :name A-AgentiM@mobile.com)
      (use
        (msg-rep-selection
          :send (sequence
            (msg-representation
              :name fipa.acl.rep.bitefficient.std))
          :recv (sequence
            (msg-representation
              :name fipa.acl.rep.bitefficient.std)))))
      true)")
```

2. Message 2 `accept-proposal`: The agent in the fixed host accepts the proposal.

```
(accept-proposal
  :sender
    (agent-identifier
      :name A-AgentiF@fixed.com)
  :receiver (set
    (agent-identifier
      :name A-AgentiM@mobile.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-propose
  :content "(
    (action
      (agent-identifier
         :name A-AgentiM@mobile.com)
      (use
        (msg-encoding
          :send (sequence
            (msg-representation :name fipa.acl.rep.bitefficient.std))
          :recv (sequence
            (msg-representation :name fipa.acl.rep.bitefficient.std)))))
    (msg-encoding-selection
       (msg-encoding
          :send (sequence
            (msg-representation :name fipa.acl.rep.bitefficient.std))
          :recv
            (sequence
              (msg-representation :name fipa.acl.rep.bitefficient.std))))))")
```

6 Paramedic Scenario

This section illustrates some of the important issues of nomadic application support, using a paramedic application as an example.

6.1 Overview

A paramedic team has several working environments:

- An emergency dispatch centre, which is covered by the hospital ATM network,

- A geographical area, which is wireless wide-area network (for example, GPRS), and,

- One or more hospitals, which are provided with a wireless local-area network.

When in transit, the paramedic computers are attached to docking stations residing in ambulances. At the dispatch centre, the docking stations are connected to the ATM network. The paramedic application comprises the following services:

- Retrieval of a patient's personal information, such as name, address, phone, and relatives,

- Retrieval of the patient's medical histories,

- Support for paramedic workers, and,

- Informing the hospital receiving the patient about the patient's current injury or illness and medical care given so far.

There are several application agents: Paramedic Support Agents (PSAs) working in the paramedic computers, Dispatching Support Agent (DSA) working at the dispatch centre system, and the Hospital First Aid Support Agent (HFASA) working at the hospital system.

The dispatch centre receives a call regarding a man who has severe chest pain; the symptom of an acute myocardial infarct. The caller identifies the man and gives his personal identification number to the dispatcher. The dispatcher alerts the paramedic team and informs the DSA about the address where the patient is located and his personal identification number. The DSA simultaneously informs the PSA about the address of the attack (and possibly some additional information about the environment of the heart attack) and queries the patient's medical history. Since the results of the query to a local hospital are received before the paramedic unit is dispatched, the DSA (in co-operation with the PSA) begins to load the patient's personal information and medical history into the paramedic computers. The medical history includes several items of text-based information. The transmission time to load the information via the ATM network to the paramedic computers (which are currently docked at the dispatch centre) is less than a second. Before the ambulance leaves the dispatch centre, the docking station is detached from the ATM network and is connected to the wireless wide-area network.

While the ambulance is approaching the location of the incident, the DSA receives more relevant results of the query of the medical histories such as the latest heart operation of the patient. The medical history comprises several parts of textual information and several images and the DSA begins loading the information. As the loading takes place when the ambulance is in motion, the DSA finds out that the quality of transport service is too low for loading some textual parts and any of the images of the medical history. It would take at least 40 minutes to download the images. Therefore, the DSA informs the PSA that images are not required for the paramedic unit. During downloading, the ambulance drives into a tunnel that causes the wireless link to be disconnected. After the tunnel, a CA re-establishes the connection and downloading continues.

At the scene, the ambulance is stationary and the quality of transmission service increases to a level at which the DSA is able to load the most relevant images (the ECGs) using an efficient compression method which is negotiated between the DSA and the PSA to the paramedic computer. The paramedic team detaches the computers from the docking station and carries them to the patient.

The paramedic team realises that they need the assistance of a medical expert located at the university hospital to stabilise the patient's condition. Therefore, they attach electrodes to the patient and the PSA starts transmitting the data of measurement such as SpO2 (oxygen saturation), cardiac rhythm, ECG, end tidal CO2 and temperature to the hospital. After successfully stabilising the patient's condition, the paramedic team moves the patient to the ambulance and sets off for the hospital. As the quality of the transport service decreases because of the motion, the PSA finds out that not all the on-going measurement data can be transmitted on-line to the hospital. Therefore, the PSA decides to transmit the most relevant data (SpO2 and cardiac rhythm). The PSA stores the rest of the data (ECG, end tidal CO2 and temperature) into a cache of the paramedic computer.

After the ambulance arrives at the hospital, the patient is transferred immediately to an operating room. Simultaneously, the paramedic team connects their paramedic computer to the wireless LAN of the hospital and the PSA transmits (in co-operation with the HFASA) all the measurement data to the hospital's system. A surgeon retrieves and analyses the measurement data before surgery.

This example illustrates a future agent-based distributed system that offers its services at the best obtainable QoS in a wide variety of environments. A possible agent architecture is illustrated in *Figure 5* which refers to three separate APs: *Dispatch*, *Gateway* and *Paracom*. In addition, there are several hospital APs which are not illustrated.

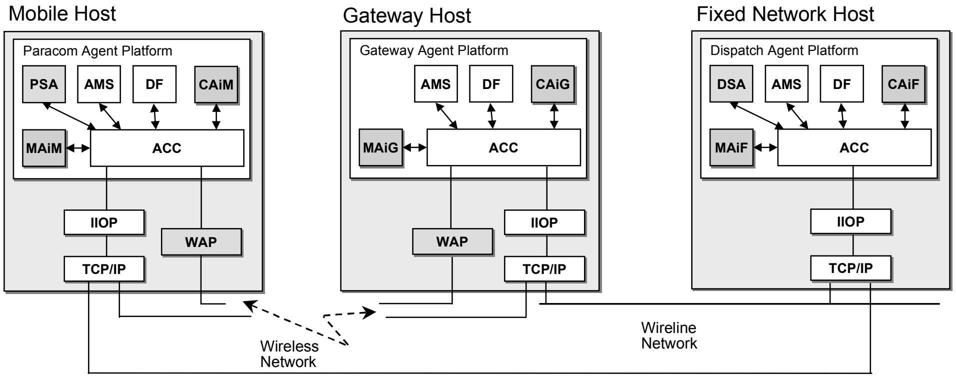

Figure 5: Paramedic Scenario Architecture

The agents in the scenario are:

- `MAiM`, `MAiG` and `MAiF` are MAs which monitor the quality of the communication service, and,

- `CAiM`, `CAiG` and `CAiF` are CAs which manage the establishment, teardown, suspension, activation, etc. of the connection between the PAs. The MA informs application agents about the status and changes of the network services.

When the mobile host is connected either to the ATM network or to the wireless LAN, the `fipa.mts.mtp.iiop.std` MTP is used directly between the *Paracom* AP and the *Dispatch* AP. When the mobile host is connected to the wireless WAN, all agent message communication takes place through the gateway host. The `fipa.mts.mtp.wap.std` MTP is primarily used between the *Paracom* AP and the *Gateway* AP. The `fipa.mts.mtp.iiop.std` MTP is used between the *Gateway* AP and the *Dispatch* AP.

6.2 Seamless Roaming

The Seamless Roaming scenario describes the process, when the paramedic computer roams from the ATM network to the UMTS network. The scenario is split into following events:

- Disconnection and reconnection of MTCs,
- Negotiation of MTPs, and,
- Negotiation of message representations.

6.2.1 Disconnection and Reconnection of an Message Transport Connection

The message exchange between the agents is illustrated in *Figure 6*.

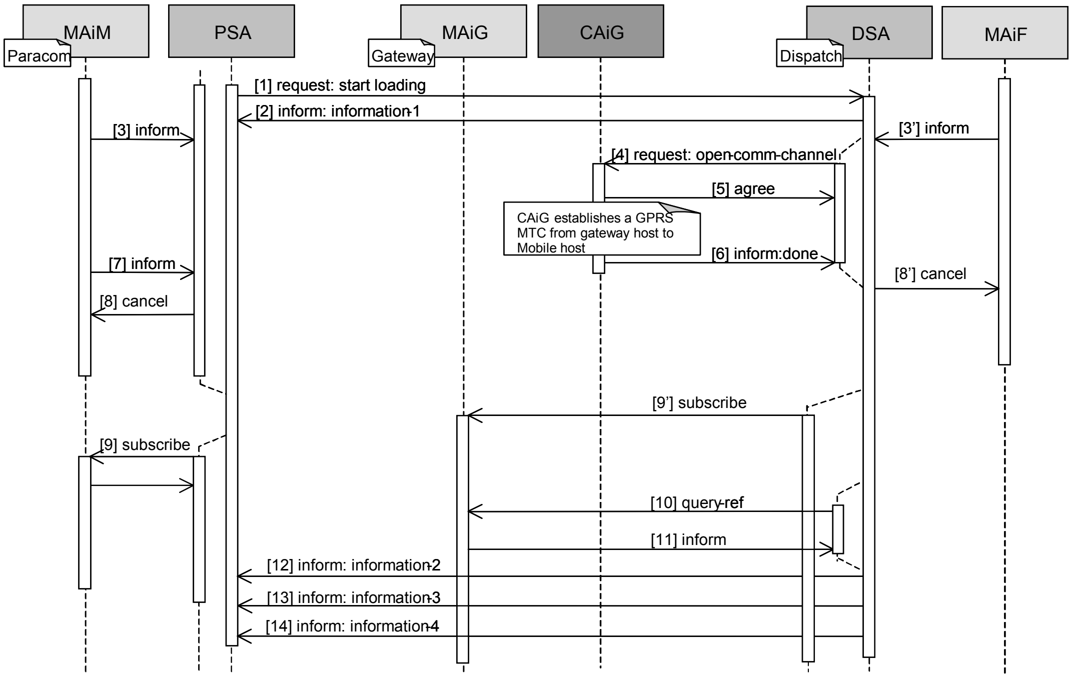
MAiM
Paracom
PSA
MAiG
Gateway
CAiG
DSA
Dispatch
MAiF
[1] request: start loading
[2] inform: information-1
[3] inform
[3'] inform
[4] request: open-comm-channel
[5] agree
CAiG establishes a GPRS MTC from gateway host to Mobile host
[6] inform:done
[7] inform
[8] cancel
[8'] cancel
[9'] subscribe
[9] subscribe
[10] query-ref
[11] inform
[12] inform: information-2
[13] inform: information-3
[14] inform: information-4

Figure 6: Disconnection and Reconnection of a Message Transport Connection

1. Message 1 `request`: The `PSA` starts loading data from the `DSA` by sending a `request` message. This message is application specific and thus not shown here.

2. Message 2 `inform`: The `DSA` starts sending information by first sending an inform message.

3. Messages 3 and 3' `inform`: `MAiM` (/`MAiF`) informs the `PSA` (/`DSA`) that the ATM connection has broken.

```
(inform
  :sender
    (agent-identifier
      :name MAiM@paracom.com)
  :receiver (set
    (agent-identifier
      :name PSA@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-subscribe
  :conversation-id subscription-3105
```

```
    :content "(
      (qos-information
        (comm-channel
          :name ATM
          :target-addr iiop://dispatch.com/acc)
        (qos
          :status disconnected)))")
```

4. Message 4 `request`: The `DSA` requests `CAiG` to open a wireless wide-area MTC.

```
  (request
    :sender
      (agent-identifier
        :name DSA@dispatch.com)
    :receiver (set
      (agent-identifier
        :name CAiG@gateway.com))
    :ontology fipa-nas
    :language fipa-sl
    :protocol fipa-request
    :content "(
      (action
        (agent-identifier
          :name CAiG@gateway.com)
        (open-comm-channel
          (comm-channel
            :name GPRS
            :target-addr iiop://paramedic.com/acc))))")
```

5. Message 5 `agree`: `CAiG` agrees that it will try to open the GPRS connection.

```
  (agree
    :sender
      (agent-identifier
        :name CAiG@gateway.com)
    :receiver (set
      (agent-identifier
        :name DSA@dispatch.com))
    :ontology fipa-nas
    :language fipa-sl
    :protocol fipa-request
    :content "(
      (action
        (agent-identifier
          :name CAiG@gateway.com)
        (open-comm-channel
          (comm-channel
            :name GPRS
            :target-addr iiop://paramedic.com/acc))))
      true)")
```

Next `CAiG` establishes a GPRS MTC from the gateway host to the mobile host (his is an implementation issue).

6. Message 6 `inform`: After successful establishment, `CAiG` informs the `DSA`.

```
  (inform
    :sender
      (agent-identifier
        :name CAiG@gateway.com)
    :receiver (set
      (agent-identifier
        :name DSA@dispatch.com))
    :ontology fipa-nas
    :language fipa-sl
    :protocol fipa-request
    :content "(
      (done
        (action
          (agent-identifier :name CAiG@gateway.com)
          (open-comm-channel
            (comm-channel :name gprs :target-addr iiop://paramedic.com/acc)))))")
```

7. Message 7 `inform`: `MAiM` informs the `PSA` that a new MTC has been established

```
(inform
  :sender
    (agent-identifier
      :name MAiM@paracom.com)
  :receiver (set
    (agent-identifier
      :name PSA@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-subscribe
  :conversation-id subscription-3105
  :content "(
      (qos-information
        (comm-channel
          :name GPRS
          :target-addr wap://paramedic.com:1234/acc)
      (qos
        :status disconnected)))")
```

8. Message 8 and 8’ `cancel`: The `PSA` (/`DSA`) cancels subscription notifications about the changes in the ATM MTC.

```
(cancel
  :sender
    (agent-identifier
      :name PSA@paracom.com)
  :receiver (set
    (agent-identifier
      :name MAiM@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-subscribe
  :content "(
    (iota ?x
      (exists ?y
        (and
          (qos-matches ?x
            (qos-information
              (comm-channel
                :name gprs
                :target-addr wap://paramedic.com:1234/acc)
              (qos :status ?y)))
          (or (= ?y connected) (= ?y disconnected))))))")
```

9. Message 9 and 9’ `subscribe`: The `DSA` (/`PSA`) subscribes to `MAiG` (/`MAiM`) for notifications about the changes in the GPRS MTC.

```
(subscribe
  :sender
    (agent-identifier
      :name DSA@dispatch.com)
  :receiver (set
    (agent-identifier
      :name MAiG@gateway.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-request
  :content "(
    (iota ?x
      (and
        (time-constraint (time-type :value every) (time-value :value 10 :unit s))
        (qos-matches ?x
          (qos-information
            (comm-channel
              :name gprs
              :target-addr iiop://paramedic.comm/acc))))))")
```

10. Message 10 `query-ref`: The `DSA` requests current QoS of the GPRS MTC from `MaiG`.

```
  (query-ref
    :sender
      (agent-identifier
        :name DSA@dispatch.com)
    :receiver (set
      (agent-identifier
        :name MAiG@gateway.com))
    :ontology fipa-nas
    :language fipa-sl
    :protocol fipa-query
    :content "(
      (iota ?x
        (qos-information
          (comm-channel
            :name gprs)
          (qos
            :throughput ?x))))")
```

11. Message 11 `inform`: `MAiG` informs the `DSA` the current QoS of the GPRS MTC.

```
  (inform
    :sender
      (agent-identifier
        :name MAiG@gateway.com)
    :receiver (set
      (agent-identifier
        :name DSA@dispatch.com))
    :ontology fipa-nas
    :language fipa-sl
    :protocol fipa-query
    :content "(
      (= (iota ?x
        (qos-information
          (comm-channel
            :name gprs)
          (qos
            :throughput ?x)))
      (rate-value
        :direction outbound
        :unit kbits/s
        :value 20))))")
```

12. Messages 12, 13 and 14 `inform`: The `DSA` sends the rest of the requested information to the `PSA`.

6.2.2 Example Negotiation of a Message Transport Protocol

When the mobile host roams from the ATM network to the GPRS network – after the reconnection – the `PSA` receives the information from `MAiM` that the *Paracom* AP is now connected to the GPRS MTC. The `PSA` reasons that the `fipa.mts.mtp.wap.std` MTP is better in that environment and it requests the `CAiM` to establish this MTP between `ACCiM` and `ACCiG`. Also, `CAiM` proposes the establishment of this MTP to `CAiG`, which accepts the proposal, and they command their respective ACCs to set it up. As a last action, both `CAiF` and `CAiG` modify the AP descriptions of their APs. The message flow is illustrated in *Figure 7*.

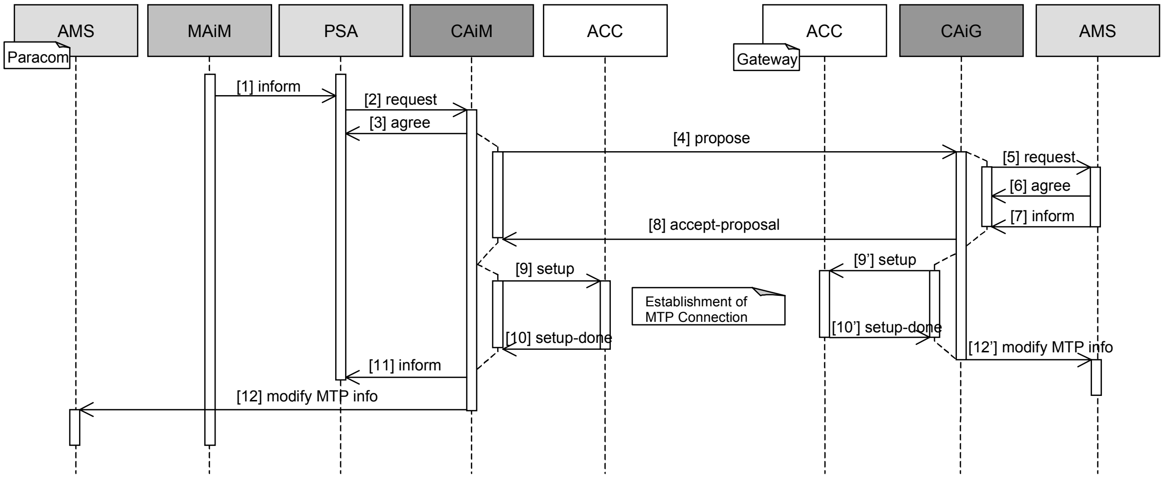

Figure 7: Example Negotiation of a Message Transport Protocol

1. Message 1 `inform`: `MAiM` informs the `PSA` that the *Paracom* AP is now connected to the GPRS network.

```
(inform
  :sender
    (agent-identifier
      :name MAiM@paracom.com)
  :receiver (set
    (agent-identifier
      :name PSA@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-subscribe
  :conversation-id subscription-3106
  :content "(
      (qos-information
        (comm-channel
          :name gprs
          :target-addr wap://paramedic.com:1234/acc)
        (qos
          :status connected)))")
```

2. Message 2 `request` and message 3 `agree`: The `PSA` requests `CAiM` to establish the `fipa.mts.mtp.wap.std` MTP between `ACCiM` and `ACCiG`.

```
(request
  :sender
    (agent-identifier
      :name PSA@paracom.com)
  :receiver (set
    (agent-identifier
      :name CAiM@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-request
  :content "(
    (action
      (agent-identifier
        :name CAiM@paracom.com)
      (activate (sequence
        (transport-protocol
          :name fipa.mts.mtp.wap.std
          :gw-addr wap://gateway.com:1234/acc)))))")
```

3. Message 4 `propose`: `CAiM` sends a `propose` message to the `CAiG`.

```
(propose
  :sender
    (agent-identifier
      :name CAiM@paracom.com)
  :receiver (set
    (agent-identifier
      :name CAiG@gateway.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-propose
  :content "(
    (action
      (agent-identifier
        :name CAiM@paracom.com)
      (use
        (transports
          :send (sequence
            (transport-protocol
              :name fipa.mts.mtp.wap.std))
          :recv (sequence
            (transport-protocol
              :name fipa.mts.mtp.wap.std)))))
    true)")
```

4. Message 5 `request`, message 6 `agree` and message 7 `inform`: `CAiG` requests the local AP description to find out if the `fipa.mts.mtp.wap.std` MTP is supported (see [FIPA00023]).

5. Message (8) `accept-proposal`: `CAiG` accepts `CAiM`'s proposal to use the `fipa.mts.mtp.wap.std` MTP.

```
(accept-proposal
  :sender
    (agent-identifier
      :name CAiG@gateway.com)
  :receiver (set
    (agent-identifier
      :name CAiM@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
```

```
    :protocol fipa-propose
    :content "(
      (action
        (agent-identifier :name CAiM@paracom.com)
        (use
          (transports
            :send (sequence (transport-protocol :name fipa.mts.mtp.wap.std))
            :recv (sequence (transport-protocol :name fipa.mts.mtp.wap.std)))))
      (transport-selection
        (transports
          :send (sequence (transport-protocol :name fipa.mts.mtp.wap.std))
          :recv (sequence (transport-protocol :name fipa.mts.mtp.wap.std)))))")
```

6. Messages 9 and 9’ `setup` and messages 10 and 10’ `setup-done`: `CAiM` (`CAiG`) commands `ACCiM` (`ACCiG`) to setup the `fipa.mts.mtp.wap.std` MTP. As this is intra-platform communication between `CAiM` (`CAiG`) and `ACCiM` (`ACCiG`), this is an implementation issue.

7. Message 11 `inform`: `CAiM` returns the result to the PSA.

```
(inform
  :sender
    (agent-identifier
      :name CAiM@paracom.com)
  :receiver (set
    (agent-identifier
      :name PSA@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-request
  :content "(
    (result
      (action
        (agent-identifier :name CAiM@paracom.com)
        (activate
          (sequence
            (transport-protocol
               :name fipa.mts.mtp.wap.std
               :gw-addr wap://gateway.com:1234/acc))))
    (transport-protocol
      :name fipa.mts.mtp.wap.std :gw-addr wap://gateway.com:1234/acc)))")
```

8. Message 12 and 12’ `set-description`: `CAiM` (`CAiG`) modifies the AP description to show that the `fipa.mts.mtp.wap.std` is now active.

6.2.3 Example Negotiation of a Message Representation

`MAiM` informs the `PSA` that the quality of the message transport connection has dropped significantly. The `PSA` reasons that the ACL representation needs to be changed to `fipa.acl.rep.bitefficient.std` and it proposes that to the `DSA`. The `DSA` accepts the PSA’s proposal. The message flow is illustrated in *Figure 11*.

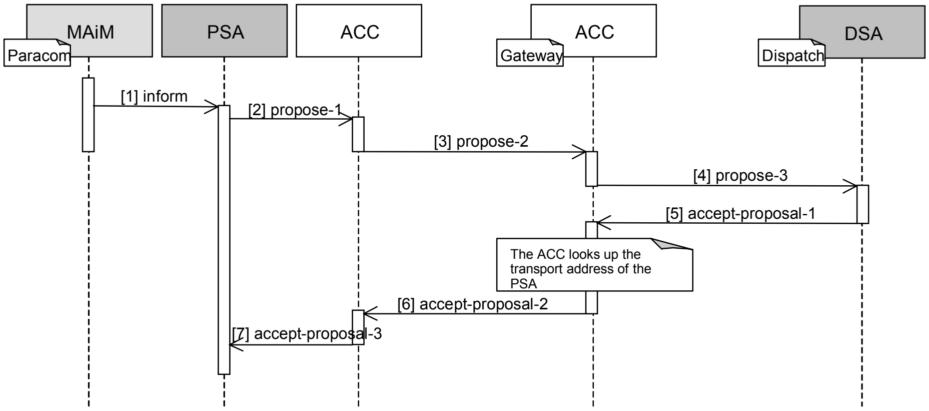

Figure 11: Example Negotiation of a Message Representation

1. Message 1 `inform`: The MA informs the `PSA` that the outbound throughput has changed.

```
(inform
  :sender
    (agent-identifier
      :name MAiM@paracom.com)
  :receiver (set
    (agent-identifier
      :name PSA@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-subscribe
  :conversation-id subscription-3106
  :content "(
    (qos-information
      (comm-channel name gprs)
      (qos :throughput
        (rate-value :unit Kbits/s :direction Outbound :value 0.96))))")
```

2. Message 2 `propose-1`: Based on the new throughput value, the `PSA` decides to change to the message representation.

```
(propose
  :sender
    (agent-identifier
      :name PSA@paracom.com)
  :receiver (set
    (agent-identifier
      :name DSA@dispatch.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-propose
  :content "(
    (action
      (agent-identifier
        :name PSA@paracom.com)
      (use
        (msg-encoding
          :send (sequence
            (msg-representation
              :name fipa.acl.rep.bitefficient.std))
          :recv (sequence
            (msg-representation
              :name fipa.acl.rep.bitefficient.std)))))
    true)")
```

3. Message 3 `propose-2`: The ACC at the mobile host forwards the same message to the ACC at the gateway host.

4. Message 4 `propose-3`: The ACC at the gateway host forwards the same message to the `PSA`.

5. Message 5 `accept-proposal-1`: The `PSA` accepts the proposal and sends a message back to the `DSA`.

```
(accept-proposal
  :sender
    (agent-identifier
      :name DSA@dispatch.com)
  :receiver (set
    (agent-identifier
      :name PSA@paracom.com))
  :ontology fipa-nas
  :language fipa-sl
  :protocol fipa-propose
  :content "(
    (action
      (agent-identifier :name PSA@paracom.com)
      (use
        (msg-encoding
          :send (sequence
            (msg-representation :name fipa.acl.rep.bitefficient.std))
          :recv (sequence
            (msg-representation :name fipa.acl.rep.bitefficient.std)))))
    (msg-encoding-selection
      (msg-encoding
        :send (sequence
          (msg-representation :name fipa.acl.rep.bitefficient.std))
        :recv (sequence
          (msg-representation :name fipa.acl.rep.bitefficient.std)))))")
```

6. Message 6 `accept-proposal-2`: The ACC at the gateway host forwards same message to the ACC at the mobile host.

7. Message 7 `accept-proposal-3`: The ACC at the mobile host delivers the same message to the `PSA`.

7 References

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

[FIPA00036] FIPA Propose Interaction Protocol Specification. Foundation for Intelligent Physical Agents, 2000.
`http://www.fipa.org/specs/fipa00036/`

[FIPA00069] FIPA ACL Message Representation in Bit-Efficient Encoding Specification. Foundation for Intelligent Physical Agents, 2000.
`http://www.fipa.org/specs/fipa00069/`

[FIPA00075] FIPA Agent Message Transport Protocol for IIOP Specification. Foundation for Intelligent Physical Agents, 2000.
`http://www.fipa.org/specs/fipa00075/`

[FIPA00076] FIPA Agent Message Transport Protocol for WAP Specification. Foundation for Intelligent Physical Agents, 2000.

[FIPA00094] FIPA Quality of Service Specification. Foundation for Intelligent Physical Agents, 2000.
`http://www.fipa.org/specs/fipa00094/`

[ITUE800] Recommendation E.800 - Telephone Network and ISDN, Quality of Service, Network Management and Traffic Engineering, Terms and Definitions Related to Quality of Service and Network Performance Including Dependability. International Telecommunication Union, International Telecommunication Union, 1995.

[ITUX135] Recommendation X.135 - Speed of Service (delay and throughput), Performance Values for Public Data Networks when Providing Packet-Switched Services. International Telegraph and Telephone Consultative Committee, 1993.

[WAP99] Wireless Application Protocol Specification Version 1.2. WAP Forum, 1999.
`http://www.wapforum.org/what/technical.htm`

8 Informative Annex A — ChangeLog

8.1 2001/10/17 - version E by TC Gateways

Page 8, lines 290-291:	Added a new frame `subscription-identifier` which is used to map subscriptions and subsequent cancel by the `subscribe-notification` and `cancel-notification` functions
Page 12, lines 340-341:	Replaced predicate `qos-notification` with function `subscribe-notification`; the `qos-notification` predicate was used as content for `subscribe` act, which is not used in this specification anymore, thus there is no need for this predicate, and, the `subscribe-notification` function replaces the subscribe act (in this spec), that is, it is used to subscribe changes in QoS
Page 12, lines 341-342:	Added new function `cancel-notification` which replaces the `cancel` act (in this spec), that is, it is used to cancel previously subscribed notification(s)
Page 13, lines 346-347:	Added sentence describing the return value of the function
Page 14, lines 364-365:	Added a new refuse reason which is needed by the `cancel-notification` function
Page 15, line 398:	Removed `fipa-subscribe` protocol from advertised protocols
Pages 22-27, lines 799-1014:	“Message Exchange over WAP MTP” section removed because: (1) the example uses dynamic registration, and, (2) the functionality can be better implemented using FIPA messaging interoperability specification and FIPA message buffering specification
Page 30, lines 1117-1119:	Figure 9 updated
Page 30, lines 1127-1145:	Example ACL message updated to follow new subscription method
Page 32, line 1216-1234:	Example ACL message updated to follow new subscription method
Page 32, lines 1236-1268:	The `cancel` method replaced with the new one which includes replacing the `cancel` ACL message with `request`, `agree` and `inform` messages (`fipa-request`)
Page 34: lines 1268-1290:	The `subscribe` method replaced with the new one which includes replacing the `subscribe` ACL message with `request`, `agree` and `inform` messages (`fipa-request`)
Page 34, line 1290:	Updated message number
Page 34, line 1310:	Updated message number
Pages 34-35, lines 1312-1332:	Example ACL message updated to follow new subscription method
Page 35, line 1334:	Updated message numbers
Page 35, lines 1350-1368:	Example ACL message updated to follow new subscription method
Page 38, lines 1496-1534:	Example ACL message updated to follow new subscription method
Page 41, lines 1599-1600:	Removed reference to `fipa-subscribe` [FIPA00035]

8.2 2002/09/13 - version F by TC X2S

Entire document:	Changed all ontology terms to lowercase
Entire document:	Ontology name changed from `FIPA-Nomadic-Application` to `fipa-nas`
Entire document:	Examples updated according to other modifications
Page 1, lines 102–103:	Removed reference to QoS ontology from the list of specification contents
Page 1, lines 105–107:	Removed reference to WAP MTP and added references to bit-efficient message envelope and to QoS ontology specifications
Page 2, lines 133–139:	Removed paragraph about WAP MTP
Page 2, lines 160–161:	Removed reference to QoS ontology
Page 5, lines 266–268:	**Removed the `qos` frame (moved to [FIPA00094])**
Page 6, lines 269–272:	**Removed the `rate-value` frame (moved to [FIPA00094])**
Page 7, lines 273–276:	**Removed the `time-value` frame (moved to [FIPA00094])**
Page 7, lines 277–280:	**Removed the `probability-value` frame (moved to [FIPA00094])**
Page 8, lines 281–284:	**Removed the `change-constraint` frame (moved to [FIPA00094])**
Page 8, lines 285–288:	**Removed the `time-constraint` frame (moved to [FIPA00094])**
Page 8, lines 289–292:	**Removed the `subscription-id` frame (moved to [FIPA00094])**
Page 8, lines 293–297:	**Removed the `comm-channel` frame (moved to [FIPA00094])**

Page 9, lines 297–300:	**Removed the `transport-protocol` frame (moved to [FIPA00094])**
Page 11, lines 340–341:	**Removed the `qos-information` predicate (moved to [FIPA00094])**
Page 11, line 340:	**Added a `transport-selection` predicate**
Page 11, line 340:	**Added an `msg-encoding-selection` predicate**
Page 12, lines 343–344:	**Removed the `subscribe-notification` function (moved to [FIPA00094])**
Page 13, lines 345–346:	**Removed the `cancel-notification` function (moved to [FIPA00094])**
Page 14, lines 362–364:	Replaced the reference to the `fipa-agent-management not-understood` exception predicates with actual predicates
Page 15, lines 366–368:	Replaced the reference to the `fipa-agent-management refusal` exception propositions with the actual propositions

8.3 2002/11/01 - version G by TC X2S

Entire document:	Updated subscription examples to use `fipa-subscribe` protocol

8.4 2002/12/03 - version H by FIPA Architecture Board

Entire document:	Promoted to Standard status