FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Agent Message Transport Protocol for IIOP Specification

Document title	FIPA Agent Message Transport Protocol for IIOP Specification		
Document number	SC00075G	**Document source**	FIPA TC Agent Management
Document status	Standard	**Date of this status**	2002/12/03
Supersedes	FIPA00024		
Contact	fab@fipa.org		
Change history	See *Informative Annex B — 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 deals with message transportation between inter-operating agents and also forms part of the FIPA Agent Management Specification [FIPA00023]. It contains specifications for:

- The transport of messages between agents using the Internet Inter-Orb Protocol (IIOP - see [OMGiiop]).

2 Message Transport Protocol for IIOP

This MTP is based on the transfer of an OMG IDL structure containing the message envelope and an octet sequence representing the ACL message body. The envelope and the message body are transferred together within a single IIOP one-way invocation [OMGiiop].

Once the request has been received, the message envelope is used by the ACC to obtain the instructions and information needed to correctly handle the message body.

2.1 Component Name

The name assigned to this component is:

```
fipa.mts.mtp.iiop.std
```

2.2 Interface Definition

The following IDL specifies the message transport interface. This interface contains a single operation `message()` that requires a single argument. This argument has two attributes: a sequence of `Envelope` structures holding the message envelope and the payload, that is a sequence of octets containing the ACL message body.

```
module FIPA {
  typedef sequence<Envelope> Envelopes;
  typedef sequence<octet> Payload;
  struct FipaMessage {
    Envelopes messageEnvelopes;
    Payload   messageBody;
  };

  interface MTS {
    oneway void message(in FipaMessage aFipaMessage);
  };
};
```

2.3 ACC Processing of IDL Envelope

According to [FIPA00067], a FIPA compliant ACC is not allowed to modify any element of the envelope that it receives. It is however allowed to update a value in one of the envelope parameters by adding a new `Envelope` element at the end of the `messageEnvelope`s sequence. This new element is required to have only those parameter values that the ACC wishes to add or update plus a new `ReceivedObject` element as mandated in [FIPA00067].

As a consequence, an ACC that receives a message must implement the procedure described in the following pseudo-code. The procedure recomposes the full envelope structure with its latest values for each parameter. The procedure simply shows that the ACC starts from the last envelope in the sequence and continues until it has all the required values for each parameter of the envelope.

```
EnvelopeWithAllFields := new empty Envelope;

while ((EnvelopeWithAllFields does not contain values for all its fields)
        OR (all Envelopes in the sequence have been processed)) {
  // the ACC gets the next envelope in the sequence starting from the end
  tempEnvelope = getNextEnvelope;

  foreach field in an envelope {
    if ((this field has no value in envelopeWithAllFields)
        AND (this field has a value in tempEnvelope))
   then copy the value of this field from tempEnvelope to envelopeWithAllFields
```

```
  }
}
```

`EnvelopeWithAllFields` now contains the latest values for all its fields.

For example:

```
Envelope(0):
  to = tizio
  from = caio
  aclRepresentation = XML
  received = …

Envelope (1):
  from = caio@molfetta.it
  received = …

Envelope (2):
  intended-receiver = tizio@villardora.it
  received = …

EnvelopeWithAllFields:
 to = tizio                              (from envelope 0)
 from = caio@molfetta.it                 (from envelope 1)
 intended-receiver = tizio@villardora.it (from envelope 2)
 date = 25 May 2000                      (from envelope 0)
```

2.4 Concrete Message Envelope Syntax

The abstract envelope syntax from [FIPA00067] maps into a set of OMG IDL structured types, all of which are enclosed within the FIPA module.

The following standard convention applies for the identification of optional parameters: an empty string and an empty sequence identify the non-presence of a parameter. In the case of the `payload-length` parameter (which is a number) any negative value can be used to identify the non-presence of the parameter.

The complete IDL definition is:

```
module FIPA {
  // No need for an URL struct, since it's only put in the
  // message envelope for informational purposes.
  typedef string URL;


  // this generic type is used to represent user-defined, non FIPA-defined,
  // properties that are added to the message envelope in the form of a
  // keyword and value pair.
  struct Property {
    string keyword;
    any value;
  };

  struct AgentID {  // Agent Identifier
    string name;
    sequence<URL>      addresses;
    sequence<AgentID>  resolvers;
    sequence<Property> userDefinedProperties;
  };

  typedef sequence<AgentID> AgentIDs;  // sequence of Agent Identifiers
```

```
170     // IDL struct to represent a time stamp.
171     // It is based on the ISO8601 format with extension for millisecond durations.
172     // The value of the typeDesignator must be a valid
173     // AlphaCharacter, i.e. ['a'-'z' , 'A'-'Z'], that identifies the timezone.
174     // ISO8601 reports the mapping between typeDesignator and timezone.
175     // The typeDesignator for UTC is the character 'Z'.
176     // If the value of typeDesignator is not an AlphaCharacter, it defaults
177     // to the local timezone.
178     struct DateTime {
179       short year;           // year (e.g. 2000)
180       short month;          // between 1 and 12
181       short day;            // between 1 and 31
182       short hour;           // between 0 and 23
183       short minutes;        // between 0 and 59
184       short seconds;        // between 0 and 59
185       short milliseconds;   // between 0 and 999
186       char  typeDesignator; // see comment above
187     };
188
189     struct ReceivedObject {
190       URL by;
191       URL from;
192       DateTime date;
193       string id;
194       string via;
195     };
196
197     typedef sequence<Property> TransportBehaviourType;
198     typedef sequence<AgentID,1> OptAgentID;
199     typedef sequence<DateTime,1> OptDateTime;
200     typedef sequence<TransportBehaviourType,1> OptTransportBehaviourType;
201     typedef sequence<ReceivedObject,1> OptReceivedObject;
202
203     struct Envelope {
204        AgentIDs                 to;
205        OptAgentID               from;
206        string                   comments;
207        string                   aclRepresentation;
208        long                     payloadLength;
209        string                   payloadEncoding;
210        OptDateTime              date;
211        AgentIDs                 intendedReceiver;
212        OptReceivedObject        received;
213        OptTransportBehaviourType transportBehaviour;
214        sequence<Property>       userDefinedProperties; // user-defined properties
215     };
216
217     typedef sequence<Envelope> Envelopes;
218     typedef sequence<octet> Payload;
219
220     struct FipaMessage {
221       Envelopes messageEnvelopes;
222       Payload   messageBody;
223     };
224
225     interface MTS {
226       oneway void message(in FipaMessage aFipaMessage);
227     };
228   };
229
```

3 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/

[OMGiiop] OMG Internet Inter-ORB Protocol Specification, Common Object Request Broker Architecture 2.2. Object Management Group, 1999.

[OMGint] ORB Interoperability Architecture, CORBA V2.3. Object Management Group, June 1999.

[OMGnam] Common Object Services Specification, Naming Service: v1.1. Object Management Group, 00-08-07, 2000.

4 Informative Annex A — URL Schemes for IIOP Addresses

Section 3.6 of OMG Naming Service specifications [OMGnam] and section 13.6 of OMG ORB Interoperability Architecture [OMGint] describe the Uniform Resource Locator (URL) schemes available to represent a CORBA object or a CORBA object bound in a Naming Service and that can be used within FIPA to represent valid IIOP addresses:

- `IOR`. The string form of an IOR `(IOR:<hex_octets>)` is a valid URL. The scheme name is **IOR** and the text after the `:` is defined in the CORBA 2.3 specification, Section 13.6.6. The IOR URL is robust and insulates the client from the encapsulated transport information and object key used to reference the object. This URL format is independent of Naming Service.

- `corbaloc`. It is difficult for humans to exchange IORs through non-electronic means because of their length and the text encoding of binary information. The `corbaloc` URL scheme provides URLs that are familiar to people and similar to ftp or http URLs. The `corbaloc` URL is described in the CORBA 2.3 Specification, Section 13.6.6. This URL format is independent of the Naming Service.

- `corbaname`. A `corbaname` URL is similar to a `corbaloc` URL. However a `corbaname` URL also contains a stringified name that identifies a binding in a naming context.

Refer to the OMG specs for how to use a CORBA Naming Resolution Service and for the complete syntax of the used URL schemes.

5 Informative Annex B — ChangeLog

5.1 2002/11/01 - version F by TC X2S

Page 3, line 146: **Removed `strings` type definition**
Page 4, line 207: **Removed `encrypted` parameter**

5.2 2002/12/03 - version G by FIPA Architecture Board

Entire document: Promoted to Standard status