

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Agent Message Transport Envelope Representation in Bit-Efficient Encoding Specification

Document title	FIPA AMT Envelope Representation in Bit-Efficient Encoding Specification		
Document number	SC00088D	Document source	FIPA TC Agent Management
Document status	Standard	Date of this status	2002/12/03
Supersedes	None		
Contact	fab@fipa.org		
Change history	See <i>Informative Annex B — ChangeLog</i>		

© 1996-2002 Foundation for Intelligent Physical Agents
<http://www.fipa.org/>
Geneva, Switzerland

Notice

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

21 **Foreword**

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

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

32 The FIPA specifications are developed through direct involvement of the FIPA membership. The status of a
33 specification can be either Preliminary, Experimental, Standard, Deprecated or Obsolete. More detail about the process
34 of specification may be found in the FIPA Document Policy [f-out-00000] and the FIPA Specifications Policy [f-out-
35 00003]. A complete overview of the FIPA specifications and their current status may be found on the FIPA Web site.

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

39 **Contents**

40	1	Scope.....	1
41	2	Bit-Efficient Envelope Representation	2
42	2.1	Component Name	2
43	2.2	ACC Processing of Bit-Efficient Envelope	2
44	2.3	Concrete Message Envelope Syntax	3
45	2.4	Notes on the Grammar Rules.....	5
46	3	References	7
47	4	Informative Annex A — Examples	8
48	5	Informative Annex B — ChangeLog	12
49	5.1	2002/11/01 - version C by TC X2S.....	12
50	5.2	2002/12/03 - version D by FIPA Architecture Board	12

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:

- Syntactic representation of a message envelope in bit-efficient form.

Informative examples of the bit-efficient envelope syntax are given in Section 4.

2 Bit-Efficient Envelope Representation

This section gives the concrete syntax for the message envelope specification that must be used to transport messages over a Message Transport Protocol (MTP - see [FIPA00067]). This concrete syntax is designed to complement [FIPA00069].

The message envelope transport syntax is expressed in standard EBNF format¹ (see *Table 1*).

Grammar rule component	Example
Terminal tokens are enclosed in double quotes	" ("
Non-terminals are written as capitalised identifiers	Expression
Square brackets denote an optional construct	[", " OptionalArg]
Vertical bars denote an alternative between choices	Integer Float
Asterisk denotes zero or more repetitions of the preceding expression	Digit*
Plus denotes one or more repetitions of the preceding expression	Alpha+
Parentheses are used to group expansions	(A B)*
Productions are written with the non-terminal name on the left-hand side, expansion on the right-hand side and terminated by a full stop	ANonTerminal = "terminal".
0x?? is a hexadecimal byte	0x00

Table 1: EBNF Rules

2.1 Component Name

The name assigned to this component is:

fipa.mts.env.rep.bitefficient.std

2.2 ACC Processing of Bit-Efficient 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 any of the envelope's parameters by adding a new `ExtEnvelope` element at the beginning of the `messageEnvelopes` sequence. This new element is required to have only those parameter values that the ACC wishes to add or update plus a new `ReceivedObject` element².

The following pseudo code algorithm may be used to obtain the latest values for each of the envelope's parameters.

```
EnvelopeWithAllParams := new empty Envelope
while (not all envelopes processed) {
  tempEnvelope = getNextEnvelope;
  foreach parameter in an envelope {
    if ((this parameter has no value in EnvelopeWithAllParams)
        AND (this parameter has a value in tempEnvelope))
      then copy the value of this parameter to EnvelopeWithAllParams
  }
}
```

EnvelopeWithAllParams now contains the latest values for all the parameters set in the envelope.

¹ White space is not allowed between tokens.
² The new `ReceivedObject` parameter s forced, syntactically, to be in all envelopes of the `messageEnvelopes` sequence except the first one.

2.3 Concrete Message Envelope Syntax

```

94 MessageEnvelope      = (ExtEnvelope)* BaseEnvelope Payload.
95
96 BaseEnvelope         = BaseEnvelopeHeader (Parameter)* EndOfEnvelope.
97
98 ExtEnvelope          = ExtEnvelopeHeader (Parameter)* EndOfEnvelope.
99
100 BaseEnvelopeHeader   = BaseMsgId EnvLen ACLRepresentation Date.
101
102 ExtEnvelopeHeader    = ExtMsgId EnvLen ReceivedObject.
103
104 EnvLen               = Len16
105                       | JumboEnvelope.          /* See comment 1 (Section 2.4) */
106
107 JumboEnvelope        = EmptyLen16 Len32.
108
109 BaseMsgId             = 0xFE.
110
111 ExtMsgId              = 0xFD.
112
113 EndOfEnvelope        = EndOfCollection.
114
115 Payload              =                               /* See comment 2 (Section 2.4) */
116
117 Parameter             = PredefinedParameter
118                       | UserDefinedParameter. /* See comment 5 (Section 2.4) */
119
120 PredefinedParameter  = 0x02 AgentIdentifierSequence      /* to */
121                       | 0x03 AgentIdentifier             /* from */
122                       | 0x04 ACLRepresentation           /* acl-representation */
123                       | 0x05 Comments                   /* comments */
124                       | 0x06 PayloadLength               /* payload-length */
125                       | 0x07 PayloadEncoding             /* payload-encoding */
126                       | 0x09 IntendedReceiver            /* intended-receiver */
127                       | 0x0a ReceivedObject              /* received */
128                       | 0x0b TransportBehaviour.         /* transport-behaviour */
129
130 ACLRepresentation    = UserDefinedACLRepresentation
131                       | 0x10 /* fipa.acl.rep.bitefficient.std [FIPA00069] */
132                       | 0x11 /* fipa.acl.rep.string.std [FIPA00070] */
133                       | 0x12. /* fipa.acl.rep.xml.std [FIPA00071] */
134
135 Date                 = BinDateTimeToken.
136
137 Comments             = NullTerminatedString.
138
139 PayloadLength        = BinNumber.
140
141 PayloadEncoding       = NullTerminatedString.
142
143 IntendedReceiver     = AgentIdentifierSequence.
144
145 TransportBehaviour   = Any.
146
147 UserDefinedACLRepresentation
148                       = 0x00 NullTerminatedString.
149
150 ReceivedObject       = By
151                       Date
152                       [From]
153                       [Id]
154                       [Via]
155

```

```

156          (UserDefinedParameter)*
157          EndOfCollection.
158
159 By          = URL.
160
161 From        = 0x02 URL.
162
163 Id          = 0x03 NullTerminatedString.
164
165 Via         = 0x04 NullTerminatedString.
166
167 BinNumber   = Digits.          /* See comment 4 (Section 2.4) */
168
169 Digits      = CodedNumber+.
170
171 NullTerminatedString = String 0x00.
172
173 UserDefinedParameter = 0x00 Keyword NullTerminatedString.
174
175 KeyWord     = NullTerminatedString.
176
177 Any         = 0x14 NullTerminatedString
178             | ByteLenEncoded.
179
180 ByteLenEncoded = 0x16 Len8 ByteSequence
181             | 0x17 Len16 ByteSequence
182             | 0x19 Len32 ByteSequence.
183
184 ByteSequence = Byte*.
185
186 AgentIdentifierSequence = (AgentIdentifier)* EndOfCollection.
187
188 AgentIdentifier = 0x02 AgentName
189                 [Addresses]
190                 [Resolvers]
191                 (UserDefinedParameter)*
192                 EndOfCollection.
193
194 AgentName     = NullTerminatedString.
195
196 Addresses     = 0x02 UrlSequence.
197
198 Resolvers     = 0x03 AgentIdentifierSequence.
199
200 UserDefinedParameter = 0x05 NullTerminatedString Any.
201
202 UrlSequence   = (URL)* EndOfCollection.
203
204 URL           = NullTerminatedString.
205
206 StringSequence = (NullTerminatedString)* EndOfCollection.
207
208 BinDateTimeToken = 0x20 BinDate          /* Absolute time */
209                 | 0x21 BinDate          /* Relative time (+) */
210                 | 0x22 BinDate          /* Relative time (-) */
211                 | 0x24 BinDate TypeDesignator /* Absolute time */
212                 | 0x25 BinDate TypeDesignator /* Relative time (+) */
213                 | 0x26 BinDate TypeDesignator /* Relative time (-) */
214
215 BinDate       = Year Month Day Hour Minute Second Millisecond.
216                 /* See comment 3 (Section 2.4) */
217 EndOfCollection = 0x01.
218
219 EmptyLen16     = 0x00 0x00.

```

```

220
221 Len8          = Byte.          /* See comment 6 (Section 2.4) */
222
223 Len16         = Short.         /* See comment 6 (Section 2.4) */
224
225 Len32         = Long.          /* See comment 6 (Section 2.4) */
226
227 Year          = Byte Byte.
228
229 Month         = Byte.
230
231 Day           = Byte.
232
233 Hour          = Byte.
234
235 Minute        = Byte.
236
237 Second        = Byte.
238
239 Millisecond    = Byte Byte.
240
241 String         =                /* As in [FIPA00070]          */
242
243 CodedNumber    =                /* See comment 4 (Section 2.4) */
244
245 TypeDesignator =                /* As in [FIPA00070]          */
246

```

2.4 Notes on the Grammar Rules

1. Normally, the length of an envelope does not exceed 65536 bytes (2^{16}). Therefore, only two bytes are reserved for envelope length (len16). However, the syntax also allows envelopes with greater lengths. In this case, the sender sets the reserved envelope length parameter (two bytes) to length zero and the following four bytes are used to represent the real length (maximum envelope length is therefore 2^{32} bytes).

The length of the envelope comprises all the parts of the envelope, including the message identifier and the length parameter itself. The length of the envelope is expressed in the network byte order.

2. The payload (ACL message) starts at the first byte after the `BaseEnvelope`. White space is allowed between the envelope and the ACL message only if the syntax of ACL allows this. For instance, `fipa.acl.rep.string.std` allows white space, but `fipa.acl.rep.bitefficient.std` does not.

3. Dates are coded as numbers, that is, four bits are reserved for each ASCII number (see comment 4 below). Information as to whether the type designator is present or not is coded into an identifier byte. These parameters always have static length (two bytes for year and milliseconds, one byte for other components).

4. Numbers are coded by reserving four bits for each digit in the number's ASCII representation, that is, two ASCII numbers are coded into one byte. *Table 2* shows a 4-bit code for each number and special codes that may appear in ASCII coded numbers.

If the ASCII presentation of a number contains an odd number of characters, the last four bits of the coded number are set to zero (the `Padding` token), otherwise an additional `0x00` byte is added to the end of the coded number. If the number to be coded is either an integer, decimal number, or octal number, the identifier byte `0x12` is used. For hexadecimal numbers, the identifier byte `0x13` is used. Hexadecimal numbers are converted to integers before coding (the coding scheme does not allow characters from `a` through `f` to appear in number form).

Token	Code		Token	Code
Padding	0000		7	1000
0	0001		8	1001
1	0010		9	1010
2	0011		+	1100
3	0100		E	1101
4	0101		-	1110
5	0110		.	1111
6	0111			

Table 2: Binary Representation of Number Tokens

5. All envelope parameters defined in [FIPA00067] have a predefined code. If an envelope contains a user-defined parameter, an extension mechanism is used (byte 0x00). The names of the user-defined envelope parameters should have the prefix "X-CompanyName-".
6. `Byte` is a one-byte code word, `Short` is a short integer (two bytes, network byte order) and `Long` is a long integer (four bytes, network byte order).

3 References

- [FIPA00067] FIPA Agent Message Transport Service Specification. Foundation for Intelligent Physical Agents, 2000.
<http://www.fipa.org/specs/fipa00067/>
- [FIPA00069] FIPA ACL Message Representation in Bit-Efficient Encoding Specification. Foundation for Intelligent Physical Agents, 2000.
<http://www.fipa.org/specs/fipa00069/>
- [FIPA00070] FIPA ACL Message Representation in String Specification. Foundation for Intelligent Physical Agents, 2000.
<http://www.fipa.org/specs/fipa00070/>
- [FIPA00071] FIPA ACL Message Representation in XML Specification. Foundation for Intelligent Physical Agents, 2000.
<http://www.fipa.org/specs/fipa00071/>

4 Informative Annex A — Examples

- Here is a simple example of an envelope encoded using XML representation:

```

<?xml version="1.0"?>
<envelope>
  <params index="1">
    <to>
      <agent-identifier>
        <name>receiver@foo.com</name>
        <addresses>
          <url>http://foo.com/acc</url>
        </addresses>
      </agent-identifier>
    </to>
    <from>
      <agent-identifier>
        <name>sender@bar.com</name>
        <addresses>
          <url>http://bar.com/acc</url>
        </addresses>
      </agent-identifier>
    </from>

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

    <date>20000508T042651481</date>

    <received>
      <received-by value="http://foo.com/acc"/>
      <received-date value="20000508T042651481"/>
      <received-id value="123456789"/>
    </received>
  </params>
</envelope>

```

Using the bit-efficient representation, the envelope becomes:

```

0xfe 0x00 0x88 0x12 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 0x03 0x02
'r' 'e' 'c' 'e' 'i' 'v' 'e' 'r' '@' 'f' 'o' 'o' '.' 'c' 'o' 'm' 0x00
0x02 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' '.' 'c' 'o' 'm' '/' 'a'
'c' 'c' 0x00 0x01 0x01 0x02 's' 'e' 'n' 'd' 'e' 'r' '@' 'b' 'a' 'r' '.'
'c' 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'b' 'a' 'r' '.' 'c'
'o' 'm' '/' 'a' 'c' 'c' 0x00 0x01 0x01 0x0a 'h' 't' 't' 'p' ':' '/' '/'
'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' 0x00 0x20 0x31 0x11 0x06 0x19
0x15 0x37 0x62 0x59 0x20 0x03 '1' '2' '3' '4' '5' '6' '7' '8' '9' 0x00 0x01

```

The length of the original message is about 584 bytes and the encoded result is 136 bytes giving a compression ratio of about 4:1.

2. Here is an example that covers all aspects of an envelope.

```

346
347
348 <?xml version="1.0"?>
349 <envelope>
350   <params index="1">
351     <to>
352       <agent-identifier>
353         <name>receiver@foo.com</name>
354         <addresses>
355           <url>http://foo.com/acc</url>
356         </addresses>
357         <resolvers>
358           <agent-identifier>
359             <name>resolver@bar.com</name>
360             <addresses>
361               <url>http://bar.com/acc1</url>
362               <url>http://bar.com/acc2</url>
363               <url>http://bar.com/acc3</url>
364             </addresses>
365           </agent-identifier>
366         </resolvers>
367       </agent-identifier>
368     </to>
369
370     <from>
371       <agent-identifier>
372         <name>sender@bar.com</name>
373         <addresses>
374           <url>http://bar.com/acc</url>
375         </addresses>
376         <resolvers>
377           <agent-identifier>
378             <name>resolver@foobar.com</name>
379             <addresses>
380               <url>http://foobar.com/acc1</url>
381               <url>http://foobar.com/acc2</url>
382               <url>http://foobar.com/acc3</url>
383             </addresses>
384           </agent-identifier>
385         </resolvers>
386       </agent-identifier>
387     </from>
388
389     <comments>No comments!</comments>
390
391     <acl-representation>fipa.acl.rep.xml.std</acl-representation>
392
393     <payload-encoding>US-ASCII</payload-encoding>
394
395     <date>20000508T042651481</date>
396
397     <intended-receiver>
398       <agent-identifier>
399         <name>intendedreceiver@foobar.com</name>
400         <addresses>
401           <url>http://foobar.com/acc1</url>
402           <url>http://foobar.com/acc2</url>
403           <url>http://foobar.com/acc3</url>
404         </addresses>
405         <resolvers>
406           <agent-identifier>
407             <name>resolver@foobar.com</name>
408             <addresses>
409               <url>http://foobar.com/acc1</url>

```

```

410         <url>http://foobar.com/acc2</url>
411         <url>http://foobar.com/acc3</url>
412     </addresses>
413     <resolvers>
414         <agent-identifier>
415             <name>resolver@foobar.com</name>
416             <addresses>
417                 <url>http://foobar.com/acc1</url>
418                 <url>http://foobar.com/acc2</url>
419                 <url>http://foobar.com/acc3</url>
420             </addresses>
421         </agent-identifier>
422     </resolvers>
423 </agent-identifier>
424 </resolvers>
425 </agent-identifier>
426 </intended-receiver>
427
428 <received>
429     <received-by value="http://foo.com/acc" />
430     <received-from value="http://foobar.com/acc" />
431     <received-date value="20000508T042651481" />
432     <received-id value="123456789" />
433     <received-via value="http://bar.com/acc" />
434 </received>
435
436 </params>
437
438 </envelope>
439

```

Using the bit-efficient representation, the envelope becomes:

```

440
441
442 0xfe 0x01 0xdb 0x12 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 0x02 'r'
443 'e' 'c' 'e' 'i' 'v' 'e' 'r' '@' 'f' 'o' 'o' '.' 'c' 'o' 'm' 0x00 0x02
444 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' '.' 'c' 'o' 'm' '/' 'a' 'c'
445 'c' 0x00 0x01 0x03 0x02 's' 'e' 'n' 'd' 'e' 'r' '@' 'b' 'a' 'r' '.' 'c'
446 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'b' 'a' 'r' '.' 'c' 'o'
447 'm' '/' 'a' 'c' 'c' 0x00 0x01 0x07 'U' 'S' '-' 'A' 'S' 'C' 'I' 'I' 0x00
448 0x01 0x09 0x02 'i' 'n' 't' 'e' 'r' 'd' 'r' 'e' 'c' 'e' 'i' 'v'
449 'e' 'r' '@' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' 0x00 0x02 'h' 't'
450 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a'
451 'c' 'c' '1' 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r'
452 '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '2' 0x00 'h' 't' 't' 'p' ':' '/' '/'
453 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '3' 0x00 0x01
454 0x03 0x02 'r' 'e' 's' 'o' 'l' 'v' 'e' 'r' '@' 'f' 'o' 'o' 'b' 'a' 'r'
455 '.' 'c' 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b'
456 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '1' 0x00 'h' 't' 't' 'p' ':'
457 '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '2'
458 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o'
459 'm' '/' 'a' 'c' 'c' '3' 0x00 0x01 0x03 0x02 'r' 'e' 's' 'o' 'l' 'v' 'e'
460 'r' '@' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' 0x00 0x02 'h' 't' 't'
461 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c'
462 'c' '1' 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.'
463 'c' 'o' 'm' '/' 'a' 'c' 'c' '2' 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f'
464 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '3' 0x00 0x01 0x01
465 0x0a 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' '.' 'c' 'o' 'm' '/' 'a'
466 'c' 'c' 0x00 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 'h' 't' 't'
467 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c'
468 'c' 0x00 0x03 '1' '2' '3' '4' '5' '6' '7' '8' '9' 0x00 0x01 0x01 0x04 'h'
469 't' 't' 'p' ':' '/' '/' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c'
470 0x00 0x01
471

```

472 The length of the original message is about 2360 bytes and the encoded result is 475 bytes giving a compression
473 ratio of about 5:1.
474

475 5 Informative Annex B — ChangeLog

476 5.1 2002/11/01 - version C by TC X2S

477 Entire document: Removed *encrypted* field

478 Page 4, line 159: Added optional `UserDefinedParameter` to the `ReceivedObject`

479 Page 4, line 202: Changed the identifier byte of the `UserDefinedParameter` from `0x04` to `0x05`

480 Page 4, line 210: Added signs to `BinDateTimeToken`

481 Page 7, lines 281-464: Moved Section 3 to Informative Annex A

482

483 5.2 2002/12/03 - version D by FIPA Architecture Board

484 Entire document: Promoted to Standard status

485