

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	XC00088B	Document source	FIPA Agent Management
Document status	Experimental	Date of this status	2001/08/10
Supersedes	None		
Contact	fab@fipa.org		
Change history			
2001/08/10	Approved for Experimental; Line numbering added		

© 2000 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.

20 **Foreword**

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

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

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

36 FIPA is a non-profit association registered in Geneva, Switzerland. As of January 2000, the 56 members of FIPA
37 represented 17 countries worldwide. Further information about FIPA as an organization, membership information, FIPA
38 specifications and upcoming meetings may be found 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	Examples	7
47	4	References.....	11
48			

48 **1 Scope**

49 This document is part of the FIPA specifications and deals with message transportation between inter-operating agents.
50 This document also forms part of the FIPA Agent Management Specification [FIPA00023] and contains specifications
51 for:

52
53 Syntactic representation of a message envelope in bit-efficient form.

54
55 Informative examples of the bit-efficient envelope syntax are given in *Section 3, Examples*.

56
57

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 format (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

N.B. White space is not allowed between tokens.

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 slots by adding a new `ExtEnvelope` element at the beginning of the `messageEnvelopes` sequence. This new element is required to have only those slot values that the ACC wishes to add or update plus a new `ReceivedObject` element¹.

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

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

`EnvelopeWithAllSlots` now contains the latest values for all the slots set in the envelope.

¹ The new `ReceivedObject` is forced, syntactically, to be in all envelopes of the `messageEnvelopes` sequence except the first one.

93 2.3 Concrete Message Envelope Syntax

```

94
95 MessageEnvelope      = (ExtEnvelope)* BaseEnvelope Payload.
96
97 BaseEnvelope        = BaseEnvelopeHeader (Slot)* EndOfEnvelope.
98
99 ExtEnvelope         = ExtEnvelopeHeader (Slot)* EndOfEnvelope.
100
101 BaseEnvelopeHeader  = BaseMsgId EnvLen ACLRepresentation Date.
102
103 ExtEnvelopeHeader   = ExtMsgId EnvLen ReceivedObject.
104
105 EnvLen               = Len16
106                     | JumboEnvelope.          /* See comment 1 (Section 2.4) */
107
108 JumboEnvelope       = EmptyLen16 Len32.
109
110 BaseMsgId            = 0xFE.
111
112 ExtMsgId             = 0xFD.
113
114 EndOfEnvelope       = EndOfCollection.
115
116 Payload              = /* See comment 2 (Section 2.4) */
117
118 Slot                 = PredefinedSlot
119                     | UserDefinedSlot.        /* See comment 5 (Section 2.4) */
120
121 PredefinedSlot      = 0x02 AgentIdentifierSequence /* to
122                     | 0x03 AgentIdentifier          /* from
123                     | 0x04 ACLRepresentation        /* acl-representation
124                     | 0x05 Comments                /* comments
125                     | 0x06 PayloadLength            /* payload-length
126                     | 0x07 PayloadEncoding          /* payload-encoding
127                     | 0x08 Encrypted                /* encrypted
128                     | 0x09 IntendedReceiver         /* intended-receiver
129                     | 0x0a ReceivedObject          /* received
130                     | 0x0b TransportBehaviour.     /* transport-behaviour */
131
132 ACLRepresentation  = UserDefinedACLRepresentation
133                     | 0x10 /* fipa.acl.rep.bitefficient.std [FIPA00069]*/
134                     | 0x11 /* fipa.acl.rep.string.std [FIPA00070] */
135                     | 0x12. /* fipa.acl.rep.xml.std [FIPA00071] */
136
137 Date                = BinDateTimeToken.
138
139 Comments            = NullTerminatedString.
140
141 PayloadLength       = BinNumber.
142
143 PayloadEncoding     = NullTerminatedString.
144
145 Encrypted           = StringSequence.
146
147 IntendedReceiver    = AgentIdentifierSequence.
148
149 TransportBehaviour  = Any.
150
151 UserDefinedACLRepresentation
152                     = 0x00 NullTerminatedString.
153
154 ReceivedObject      = By

```

```

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

```

```

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

```

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 slot (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 slot 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 slots 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).

271

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).

272
 273
 274
 275
 276
 277
 278
 279
 280
 281

3 Examples

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

```

281 <?xml version="1.0"?>
282 <envelope>
283   <params index="1">
284     <to>
285       <agent-identifier>
286         <name>receiver@foo.com</name>
287         <addresses>
288           <url>http://foo.com/acc</url>
289         </addresses>
290       </agent-identifier>
291     </to>
292     <from>
293       <agent-identifier>
294         <name>sender@bar.com</name>
295         <addresses>
296           <url>http://bar.com/acc</url>
297         </addresses>
298       </agent-identifier>
299     </from>
300     <acl-representation>fipa.acl.rep.xml.std</acl-representation>
301     <date>20000508T042651481</date>
302     <encrypted>no encryption</encrypted>
303     <received>
304       <received-by value="http://foo.com/acc" />
305       <received-date value="20000508T042651481" />
306       <received-id value="123456789" />
307     </received>
308   </params>
309 </envelope>

```

Using the bit-efficient representation, the envelope becomes:

```

310 0xfe 0x00 0x97 0x12 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 0x03 0x02
311 'r' 'e' 'c' 'e' 'i' 'v' 'e' 'r' '@' 'f' 'o' 'o' '.' 'c' 'o' 'm' 0x00
312 0x02 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' '.' 'c' 'o' 'm' '/' 'a'
313 'c' 'c' 0x00 0x01 0x01 0x02 's' 'e' 'n' 'd' 'e' 'r' '@' 'b' 'a' 'r' '.'
314 'c' 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'b' 'a' 'r' '.' 'c'
315 'o' 'm' '/' 'a' 'c' 'c' 0x00 0x01 0x01 0x08 'n' 'o' ' ' 'e' 'n' 'c' 'r'
316 'y' 'p' 't' 'e' 'd' ' ' 'n' 'o' 0x00 0x0a 'h' 't' 't' 'p' ':' '/' '/' 'b' 'a'
317 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' 0x00 0x20 0x31 0x11 0x06 0x19 0x15 0x37
318 0x62 0x59 0x20 0x03 '1' '2' '3' '4' '5' '6' '7' '8' '9' 0x00 0x01

```

The length of the original message is about 620 bytes and the encoded result is 151 bytes giving a compression ratio of about 4:1.

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

```

334 <?xml version="1.0"?>
335 <envelope>
336   <params index="1">
337     <to>
338       <agent-identifier>
339         <name>receiver@foo.com</name>
340         <addresses>
341           <url>http://foo.com/acc</url>
342         </addresses>
343       </resolvers>
344     <agent-identifier>
345       <name>resolver@bar.com</name>
346       <addresses>
347         <url>http://bar.com/acc1</url>
348         <url>http://bar.com/acc2</url>
349         <url>http://bar.com/acc3</url>
350       </addresses>
351     </agent-identifier>
352   </resolvers>
353 </agent-identifier>
354 </to>
355 <from>
356   <agent-identifier>
357     <name>sender@bar.com</name>
358     <addresses>
359       <url>http://bar.com/acc</url>
360     </addresses>
361   <resolvers>
362     <agent-identifier>
363       <name>resolver@foobar.com</name>
364       <addresses>
365         <url>http://foobar.com/acc1</url>
366         <url>http://foobar.com/acc2</url>
367         <url>http://foobar.com/acc3</url>
368       </addresses>
369     </agent-identifier>
370   </resolvers>
371 </agent-identifier>
372 </from>
373 <comments>No comments!</comments>
374 <acl-representation>fipa.acl.rep.xml.std</acl-representation>
375 <payload-encoding>US-ASCII</payload-encoding>
376 <date>20000508T042651481</date>
377 <encrypted>no encryption</encrypted>
378 <intended-receiver>
379   <agent-identifier>
380     <name>intendedreceiver@foobar.com</name>
381     <addresses>
382       <url>http://foobar.com/acc1</url>
383       <url>http://foobar.com/acc2</url>
384       <url>http://foobar.com/acc3</url>
385     </addresses>
386   <resolvers>
387     <agent-identifier>
388       <name>resolver@foobar.com</name>
389     </agent-identifier>
390   </resolvers>
391 </intended-receiver>
392 </envelope>
393

```

```

397     <addresses>
398         <url>http://foobar.com/acc1</url>
399         <url>http://foobar.com/acc2</url>
400         <url>http://foobar.com/acc3</url>
401     </addresses>
402     <resolvers>
403         <agent-identifier>
404             <name>resolver@foobar.com</name>
405             <addresses>
406                 <url>http://foobar.com/acc1</url>
407                 <url>http://foobar.com/acc2</url>
408                 <url>http://foobar.com/acc3</url>
409             </addresses>
410         </agent-identifier>
411     </resolvers>
412 </agent-identifier>
413 </resolvers>
414 </agent-identifier>
415 </intended-receiver>
416
417
418 <received>
419     <received-by value="http://foo.com/acc" />
420     <received-from value="http://foobar.com/acc" />
421     <received-date value="20000508T042651481" />
422     <received-id value="123456789" />
423     <received-via value="http://bar.com/acc" />
424 </received>
425
426 </params>
427
428 </envelope>
429

```

Using the bit-efficient representation, the envelope becomes:

```

430
431
432 0xfe 0x01 0xea 0x12 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 0x02 'r'
433 'e' 'c' 'e' 'i' 'v' 'e' 'r' '@' 'f' 'o' 'o' '.' 'c' 'o' 'm' 0x00 0x02
434 'h' 't' 't' 'p' ':' '/' 'f' 'o' 'o' '.' 'c' 'o' 'm' '/' 'a' 'c'
435 'c' 0x00 0x01 0x03 0x02 's' 'e' 'n' 'd' 'e' 'r' '@' 'b' 'a' 'r' '.' 'c'
436 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'b' 'a' 'r' '.' 'c' 'o'
437 'm' '/' 'a' 'c' 'c' 0x00 0x01 0x07 'U' 'S' '-' 'A' 'S' 'C' 'I' 'I' 0x00
438 0x08 'n' 'o' '.' 'e' 'n' 'c' 'r' 'y' 'p' 't' 'i' 'o' 'n' 0x00 0x01 0x09
439 0x02 'i' 'n' 't' 'e' 'n' 'd' 'e' 'd' 'r' 'e' 'c' 'e' 'i' 'v' 'e' 'r'
440 '@' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' 0x00 0x02 'h' 't' 't' 'p'
441 ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c'
442 '1' 0x00 'h' 't' 't' 'p' ':' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c'
443 'o' 'm' '/' 'a' 'c' 'c' '2' 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o'
444 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '3' 0x00 0x01 0x03 0x02
445 'r' 'e' 's' 'o' 'l' 'v' 'e' 'r' '@' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c'
446 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r'
447 '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '1' 0x00 'h' 't' 't' 'p' ':' '/' '/'
448 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '2' 0x00 'h'
449 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/'
450 'a' 'c' 'c' '3' 0x00 0x01 0x03 0x02 'r' 'e' 's' 'o' 'l' 'v' 'e' 'r' '@'
451 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':'
452 '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '1'
453 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o'
454 'm' '/' 'a' 'c' 'c' '2' 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o'
455 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '3' 0x00 0x01 0x01 0x0a 'h'
456 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c'
457 0x00 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 'h' 't' 't' 'p' ':'
458 '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' 0x00
459 0x03 '1' '2' '3' '4' '5' '6' '7' '8' '9' 0x00 0x01 0x01 0x04 'h' 't' 't'
460 'p' ':' '/' '/' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' 0x00 0x01

```

461

462 The length of the original message is about 2400 bytes and the encoded result is 490 bytes giving a compression ratio
463 of about 5:1.

464

464 **4 References**

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

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

470 [FIPA00070] FIPA ACL Message Representation in String Specification. Foundation for Intelligent Physical Agents,
471 2000.
472 <http://www.fipa.org/specs/fipa00070/>

473 [FIPA00071] FIPA ACL Message Representation in XML Specification. Foundation for Intelligent Physical Agents,
474 2000.
475 <http://www.fipa.org/specs/fipa00071/>