

# FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

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

<b>Document title</b>	FIPA AMT Envelope Representation in Bit-Efficient Encoding Specification		
<b>Document number</b>	XC00088C	<b>Document source</b>	FIPA TC Agent Management
<b>Document status</b>	Experimental	<b>Date of this status</b>	2002/11/01
<b>Supersedes</b>	None		
<b>Contact</b>	fab@fipa.org		
<b>Change history</b>	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 **1 Scope**

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

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

55

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

57

## 58 2 Bit-Efficient Envelope Representation

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

62  
63 The message envelope transport syntax is expressed in standard EBNF format<sup>1</sup> (see *Table 1*).  
64

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

65  
66 **Table 1:** EBNF Rules  
67

### 68 2.1 Component Name

69 The name assigned to this component is:

70  
71 `fipa.mts.env.rep.bitefficient.std`  
72

### 73 2.2 ACC Processing of Bit-Efficient Envelope

74 According to [FIPA00067], a FIPA compliant ACC is not allowed to modify any element of the envelope that it receives.  
75 It is however allowed to update a value in any of the envelope's parameters by adding a new `ExtEnvelope` element at  
76 the beginning of the `messageEnvelopes` sequence. This new element is required to have only those parameter values  
77 that the ACC wishes to add or update plus a new `ReceivedObject` element<sup>2</sup>.

78  
79 The following pseudo code algorithm may be used to obtain the latest values for each of the envelope's parameters.  
80

```
81 EnvelopeWithAllParams := new empty Envelope
82 while (not all envelopes processed) {
83   tempEnvelope = getNextEnvelope;
84   foreach parameter in an envelope {
85     if ((this parameter has no value in EnvelopeWithAllParams)
86         AND (this parameter has a value in tempEnvelope))
87       then copy the value of this parameter to EnvelopeWithAllParams
88   }
89 }
90
```

91 `EnvelopeWithAllParams` now contains the latest values for all the parameters set in the envelope.  
92

<sup>1</sup> White space is not allowed between tokens.

<sup>2</sup> The new `ReceivedObject` parameter is forced, syntactically, to be in all envelopes of the `messageEnvelopes` sequence except the first one.

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

```

```

219
220 Len8           = Byte.           /* See comment 6 (Section 2.4) */
221
222 Len16          = Short.          /* See comment 6 (Section 2.4) */
223
224 Len32          = Long.           /* See comment 6 (Section 2.4) */
225
226 Year           = Byte Byte.
227
228 Month          = Byte.
229
230 Day            = Byte.
231
232 Hour           = Byte.
233
234 Minute         = Byte.
235
236 Second         = Byte.
237
238 Millisecond     = Byte Byte.
239
240 String         =                  /* As in [FIPA00070]           */
241
242 CodedNumber    =                  /* See comment 4 (Section 2.4) */
243
244 TypeDesignator =                  /* As in [FIPA00070]           */
245

```

## 246 2.4 Notes on the Grammar Rules

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

251  
 252 The length of the envelope comprises all the parts of the envelope, including the message identifier and the length  
 253 parameter itself. The length of the envelope is expressed in the network byte order.

254  
 255 2. The payload (ACL message) starts at the first byte after the BaseEnvelope. White space is allowed between the  
 256 envelope and the ACL message only if the syntax of ACL allows this. For instance, `fipa.acl.rep.string.std`  
 257 allows white space, but `fipa.acl.rep.bitefficient.std` does not.

258  
 259 3. Dates are coded as numbers, that is, four bits are reserved for each ASCII number (see comment 4 below).  
 260 Information as to whether the type designator is present or not is coded into an identifier byte. These parameters  
 261 always have static length (two bytes for year and milliseconds, one byte for other components).

262  
 263 4. Numbers are coded by reserving four bits for each digit in the number's ASCII representation, that is, two ASCII  
 264 numbers are coded into one byte. *Table 2* shows a 4-bit code for each number and special codes that may appear  
 265 in ASCII coded numbers.

266  
 267 If the ASCII presentation of a number contains an odd number of characters, the last four bits of the coded number  
 268 are set to zero (the `Padding` token), otherwise an additional `0x00` byte is added to the end of the coded number. If  
 269 the number to be coded is either an integer, decimal number, or octal number, the identifier byte `0x12` is used. For  
 270 hexadecimal numbers, the identifier byte `0x13` is used. Hexadecimal numbers are converted to integers before  
 271 coding (the coding scheme does not allow characters from `a` through `f` to appear in number form).



272

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			

273

**Table 2:** Binary Representation of Number Tokens

274

275

276

277

278

279

280

281

282

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).

283

### 3 References

284

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

285

286

[FIPA00069] FIPA ACL Message Representation in Bit-Efficient Encoding Specification. Foundation for Intelligent Physical Agents, 2000.

287

288

<http://www.fipa.org/specs/fipa00069/>

289

[FIPA00070] FIPA ACL Message Representation in String Specification. Foundation for Intelligent Physical Agents, 2000.

290

291

<http://www.fipa.org/specs/fipa00070/>

292

[FIPA00071] FIPA ACL Message Representation in XML Specification. Foundation for Intelligent Physical Agents, 2000.

293

294

<http://www.fipa.org/specs/fipa00071/>

295

## 296 4 Informative Annex A — Examples

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

```

298 <?xml version="1.0"?>
299 <envelope>
300   <params index="1">
301     <to>
302       <agent-identifier>
303         <name>receiver@foo.com</name>
304         <addresses>
305           <url>http://foo.com/acc</url>
306         </addresses>
307       </agent-identifier>
308     </to>
309     <from>
310       <agent-identifier>
311         <name>sender@bar.com</name>
312         <addresses>
313           <url>http://bar.com/acc</url>
314         </addresses>
315       </agent-identifier>
316     </from>
317
318     <acl-representation>fipa.acl.rep.xml.std</acl-representation>
319
320     <date>20000508T042651481</date>
321
322     <received>
323       <received-by value="http://foo.com/acc"/>
324       <received-date value="20000508T042651481"/>
325       <received-id value="123456789"/>
326     </received>
327   </params>
328 </envelope>
329
330

```

331 Using the bit-efficient representation, the envelope becomes:

```

332
333 0xfe 0x00 0x88 0x12 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 0x03 0x02
334 `r' `e' `c' `e' `i' `v' `e' `r' `@' `f' `o' `o' `.` `c' `o' `m' 0x00
335 0x02 `h' `t' `t' `p' `:` `/' `/' `f' `o' `o' `.` `c' `o' `m' `/' `a'
336 `c' `c' 0x00 0x01 0x01 0x02 `s' `e' `n' `d' `e' `r' `@' `b' `a' `r' `.`
337 `c' `o' `m' 0x00 0x02 `h' `t' `t' `p' `:` `/' `/' `b' `a' `r' `.` `c'
338 `o' `m' `/' `a' `c' `c' 0x00 0x01 0x01 0x0a `h' `t' `t' `p' `:` `/' `/'
339 `b' `a' `r' `.` `c' `o' `m' `/' `a' `c' `c' 0x00 0x20 0x31 0x11 0x06 0x19
340 0x15 0x37 0x62 0x59 0x20 0x03 `1' `2' `3' `4' `5' `6' `7' `8' `9' 0x00 0x01
341

```

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

344

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

```

345
346
347 <?xml version="1.0"?>
348 <envelope>
349   <params index="1">
350     <to>
351       <agent-identifier>
352         <name>receiver@foo.com</name>
353         <addresses>
354           <url>http://foo.com/acc</url>
355         </addresses>
356         <resolvers>
357           <agent-identifier>
358             <name>resolver@bar.com</name>
359             <addresses>
360               <url>http://bar.com/acc1</url>
361               <url>http://bar.com/acc2</url>
362               <url>http://bar.com/acc3</url>
363             </addresses>
364           </agent-identifier>
365         </resolvers>
366       </agent-identifier>
367     </to>
368
369     <from>
370       <agent-identifier>
371         <name>sender@bar.com</name>
372         <addresses>
373           <url>http://bar.com/acc</url>
374         </addresses>
375         <resolvers>
376           <agent-identifier>
377             <name>resolver@foobar.com</name>
378             <addresses>
379               <url>http://foobar.com/acc1</url>
380               <url>http://foobar.com/acc2</url>
381               <url>http://foobar.com/acc3</url>
382             </addresses>
383           </agent-identifier>
384         </resolvers>
385       </agent-identifier>
386     </from>
387
388     <comments>No comments!</comments>
389
390     <acl-representation>fipa.acl.rep.xml.std</acl-representation>
391
392     <payload-encoding>US-ASCII</payload-encoding>
393
394     <date>20000508T042651481</date>
395
396     <intended-receiver>
397       <agent-identifier>
398         <name>intendedreceiver@foobar.com</name>
399         <addresses>
400           <url>http://foobar.com/acc1</url>
401           <url>http://foobar.com/acc2</url>
402           <url>http://foobar.com/acc3</url>
403         </addresses>
404         <resolvers>
405           <agent-identifier>
406             <name>resolver@foobar.com</name>
407             <addresses>
408               <url>http://foobar.com/acc1</url>

```

```

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

```

439 Using the bit-efficient representation, the envelope becomes:

```

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

```

471  
472  
473

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

## 474 5 Informative Annex B — ChangeLog

### 475 5.1 2002/11/01 – version C by TC X2S

476 Entire document: Removed *encrypted* field

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

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

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

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

481