

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	XC00088C	Document source	FIPA Agent Management
Document status	Experimental	Date of this status	2002/10/18
Supersedes	None		
Contact	fab@fipa.org		
Change history			
2001/08/10	Approved for Experimental; Line numbering added		
2002/05/22	See <i>Informative Annex A — 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.

19 **Foreword**

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

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

30 The FIPA specifications are developed through direct involvement of the FIPA membership. The status of a
31 specification can be either Preliminary, Experimental, Standard, Deprecated or Obsolete. More detail about the
32 process of specification may be found in the FIPA Document Policy [f-out-00000] and the FIPA Specifications Policy [f-
33 out-00003]. A complete overview of the FIPA specifications and their current status may be found on the FIPA Web
34 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	Examples	7
47	4	References	11
48	5	Informative Annex A — ChangeLog.....	12
49	5.1	2002/05/22 – version C	12
50			

50 **1 Scope**

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

- 54
- 55 • Syntactic representation of a message envelope in bit-efficient form.
- 56

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

58

59

59 2 Bit-Efficient Envelope Representation

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

63
64 The message envelope transport syntax is expressed in standard EBNF format (see *Table 1*).
65

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

66
67 **Table 1: EBNF Rules**

68
69 N.B. White space is not allowed between tokens.
70

71 2.1 Component Name

72 The name assigned to this component is:

73
74 `fipa.mts.env.rep.bitefficient.std`
75

76 2.2 ACC Processing of Bit-Efficient Envelope

77 According to [FIPA00067], a FIPA compliant ACC is not allowed to modify any element of the envelope that it receives.
78 It is however allowed to update a value in any of the envelope's slots by adding a new `ExtEnvelope` element at the
79 beginning of the `messageEnvelopes` sequence. This new element is required to have only those slot values that the
80 ACC wishes to add or update plus a new `ReceivedObject` element¹.

81
82 The following pseudo code algorithm may be used to obtain the latest values for each of the envelope's slots.
83

```
84 EnvelopeWithAllSlots := new empty Envelope
85 while (not all envelopes processed) {
86     tempEnvelope = getNextEnvelope;
87     foreach slot in an envelope {
88         if ((this slot has no value in EnvelopeWithAllSlots)
89             AND (this slot has a value in tempEnvelope))
90             then copy the value of this slot to EnvelopeWithAllSlots
91     }
92 }
```

93
94 `EnvelopeWithAllSlots` now contains the latest values for all the slots set in the envelope.
95

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

95 2.3 Concrete Message Envelope Syntax

```

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

```

```

156             [Id]
157             [Via]
158             (UserDefinedParameter)*
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 = 0x05 NullTerminatedString Any.
203
204 UrlSequence  = (URL)* EndOfCollection.
205
206 URL          = NullTerminatedString.
207
208 StringSequence = (NullTerminatedString)* EndOfCollection.
209
210 BinDateTimeToken = 0x20 BinDate           /* Absolute time */
211             | 0x21 BinDate           /* Relative time (+) */
212             | 0x22 BinDate           /* Relative time (-) */
213             | 0x24 BinDate TypeDesignator /* Absolute time */
214             | 0x25 BinDate TypeDesignator. /* Relative time (+) */
215             | 0x26 BinDate TypeDesignator. /* Relative time (-) */
216
217 BinDate      = Year Month Day Hour Minute Second Millisecond.
218             /* See comment 3 (Section 2.4) */

```

```

219 EndOfCollection      = 0x01.
220
221 EmptyLen16          = 0x00 0x00.
222
223 Len8                 = Byte.          /* See comment 6 (Section 2.4) */
224
225 Len16                = Short.        /* See comment 6 (Section 2.4) */
226
227 Len32                = Long.         /* See comment 6 (Section 2.4) */
228
229 Year                 = Byte Byte.
230
231 Month                = Byte.
232
233 Day                  = Byte.
234
235 Hour                 = Byte.
236
237 Minute               = Byte.
238
239 Second               = Byte.
240
241 Millisecond          = Byte Byte.
242
243 String                = /* As in [FIPA00070] */
244
245 CodedNumber          = /* See comment 4 (Section 2.4) */
246
247 TypeDesignator      = /* As in [FIPA00070] */
248

```

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).

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).

276
277
278
279
280
281
282
283
284
285

3 Examples

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

```

285
286
287 1. Here is a simple example of an envelope encoded using XML representation:
288
289 <?xml version="1.0"?>
290 <envelope>
291   <params index="1">
292     <to>
293       <agent-identifier>
294         <name>receiver@foo.com</name>
295         <addresses>
296           <url>http://foo.com/acc</url>
297         </addresses>
298       </agent-identifier>
299     </to>
300     <from>
301       <agent-identifier>
302         <name>sender@bar.com</name>
303         <addresses>
304           <url>http://bar.com/acc</url>
305         </addresses>
306       </agent-identifier>
307     </from>
308
309     <acl-representation>fipa.acl.rep.xml.std</acl-representation>
310
311     <date>20000508T042651481</date>
312
313     <received>
314       <received-by value="http://foo.com/acc" />
315       <received-date value="20000508T042651481" />
316       <received-id value="123456789" />
317     </received>
318   </params>
319 </envelope>
320

```

Using the bit-efficient representation, the envelope becomes:

```

321
322
323 0xfe 0x00 0x88 0x12 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 0x03 0x02
324 'r' 'e' 'c' 'e' 'i' 'v' 'e' 'r' '@' 'f' 'o' 'o' '.' 'c' 'o' 'm' 0x00
325 0x02 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' '.' 'c' 'o' 'm' '/' 'a'
326 'c' 'c' 0x00 0x01 0x01 0x02 's' 'e' 'n' 'd' 'e' 'r' '@' 'b' 'a' 'r' '.'
327 'c' 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'b' 'a' 'r' '.' 'c'
328 'o' 'm' '/' 'a' 'c' 'c' 0x00 0x01 0x01 0x0a 'h' 't' 't' 'p' ':' '/' '/'
329 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' 0x00 0x20 0x31 0x11 0x06 0x19
330 0x15 0x37 0x62 0x59 0x20 0x03 '1' '2' '3' '4' '5' '6' '7' '8' '9' 0x00 0x01
331

```

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.

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

```

335 <?xml version="1.0"?>
336 <envelope>
337   <params index="1">
338     <to>
339       <agent-identifier>
340         <name>receiver@foo.com</name>
341         <addresses>
342           <url>http://foo.com/acc</url>
343         </addresses>
344         <resolvers>
345           <agent-identifier>
346             <name>resolver@bar.com</name>
347             <addresses>
348               <url>http://bar.com/acc1</url>
349               <url>http://bar.com/acc2</url>
350               <url>http://bar.com/acc3</url>
351             </addresses>
352           </agent-identifier>
353         </resolvers>
354       </agent-identifier>
355     </to>
356     <from>
357       <agent-identifier>
358         <name>sender@bar.com</name>
359         <addresses>
360           <url>http://bar.com/acc</url>
361         </addresses>
362         <resolvers>
363           <agent-identifier>
364             <name>resolver@foobar.com</name>
365             <addresses>
366               <url>http://foobar.com/acc1</url>
367               <url>http://foobar.com/acc2</url>
368               <url>http://foobar.com/acc3</url>
369             </addresses>
370           </agent-identifier>
371         </resolvers>
372       </agent-identifier>
373     </from>
374   <comments>No comments!</comments>
375   <acl-representation>fipa.acl.rep.xml.std</acl-representation>
376   <payload-encoding>US-ASCII</payload-encoding>
377   <date>20000508T042651481</date>
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           <addresses>
390             <url>http://foobar.com/acc1</url>
391             <url>http://foobar.com/acc2</url>
392             <url>http://foobar.com/acc3</url>
393           </addresses>
394         </resolvers>
395       </agent-identifier>
396     </intended-receiver>
397   </params>
398 </envelope>
399
400
```

```

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

```

Using the bit-efficient representation, the envelope becomes:

```

429
430 0xfe 0x01 0xdb 0x12 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 0x02 'r'
431 'e' 'c' 'e' 'i' 'v' 'e' 'r' '@' 'f' 'o' 'o' '.' 'c' 'o' 'm' 0x00 0x02
432 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' '.' 'c' 'o' 'm' '/' 'a' 'c'
433 'c' 0x00 0x01 0x03 0x02 's' 'e' 'n' 'd' 'e' 'r' '@' 'b' 'a' 'r' '.' 'c'
434 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'b' 'a' 'r' '.' 'c' 'o'
435 'm' '/' 'a' 'c' 'c' 0x00 0x01 0x07 'U' 'S' '-' 'A' 'S' 'C' 'I' 'I' 0x00
436 0x01 0x09 0x02 'i' 'n' 't' 'e' 'n' 'd' 'e' 'd' 'r' 'e' 'c' 'e' 'i' 'v'
437 'e' 'r' '@' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' 0x00 0x02 'h' 't'
438 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a'
439 'c' 'c' '1' 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r'
440 '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '2' 0x00 'h' 't' 't' 'p' ':' '/' '/'
441 'f' 'o' 'o' 'b' 'a' 'r' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '3' 0x00 0x01
442 0x03 0x02 'r' 'e' 's' 'o' 'l' 'v' 'e' 'r' '@' 'f' 'o' 'o' 'b' 'a' 'r'
443 '.' 'c' 'o' 'm' 0x00 0x02 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b'
444 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '1' 0x00 'h' 't' 't' 'p' ':'
445 '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '2'
446 0x00 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o'
447 'm' '/' 'a' 'c' 'c' '3' 0x00 0x01 0x03 0x02 'r' 'e' 's' 'o' 'l' 'v' 'e'
448 'r' '@' 'f' 'o' 'o' 'b' 'a' 'r' 'r' '.' 'c' 'o' 'm' 0x00 0x02 'h' 't' 't'
449 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c'
450 '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' ':' '/' '/' 'f'
452 'o' 'o' 'b' 'a' 'r' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c' '3' 0x00 0x01 0x01
453 0x0a 'h' 't' 't' 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' '.' 'c' 'o' 'm'
454 'c' 'c' 0x00 0x20 0x31 0x11 0x06 0x19 0x15 0x37 0x62 0x59 0x20 0x02 'h' 't' 't'
455 'p' ':' '/' '/' 'f' 'o' 'o' 'b' 'a' 'r' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c'
456 'c' 0x00 0x03 '1' '2' '3' '4' '5' '6' '7' '8' '9' 0x00 0x01 0x01 0x04 'h'
457 't' 't' 'p' ':' '/' '/' 'b' 'a' 'r' 'r' '.' 'c' 'o' 'm' '/' 'a' 'c' 'c'
458 0x00 0x01
459

```

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.

462 **4 References**

- 463 [FIPA00067] FIPA Agent Message Transport Service Specification. Foundation for Intelligent Physical Agents,
464 2000. <http://www.fipa.org/specs/fipa00067/>
- 465 [FIPA00069] FIPA ACL Message Representation in Bit-Efficient Encoding Specification. Foundation for Intelligent
466 Physical Agents, 2000.
467 <http://www.fipa.org/specs/fipa00069/>
- 468 [FIPA00070] FIPA ACL Message Representation in String Specification. Foundation for Intelligent Physical Agents,
469 2000.
470 <http://www.fipa.org/specs/fipa00070/>
- 471 [FIPA00071] FIPA ACL Message Representation in XML Specification. Foundation for Intelligent Physical Agents,
472 2000.
473 <http://www.fipa.org/specs/fipa00071/>
- 474

474 **5 Informative Annex A — ChangeLog**

475 **5.1 2002/05/22 – version C**

476 **Page 3, Line 128:** Removed the “encrypted” field.

477

478 **Page 3, Line 146:** Removed a production related the “encrypted” field.

479

480 **Page 4, Line 159:** Added optional UserDefinedParameter to the ReceivedObject.

481

482 **Page 4, Line 203:** Changed the identifier byte of the UserDefinedParameter from 0x04 to 0x05.

483

484 **Page 4, Lines 210-222:** Added Sign to DateTimeToken.

485

486 **Examples:** Removed the “encrypted” field and updated the bit-efficient versions accordingly.