

1
2
3
4

FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

5
6
7

FIPA Message Buffering Service Specification

Document title	FIPA Message Buffering Service Specification		
Document number	XC00092B	Document source	FIPA TC Gateways
Document status	Experimental	Date of this status	2002/05/10
Supersedes	None		
Contact	fab@fipa.org		
Change history			
2002/05/10	Approved for Experimental		

8
9
10
11
12
13
14
15
16
17 © 2001 Foundation for Intelligent Physical Agents - <http://www.fipa.org/>

18 *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.

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 Procedures for Technical Work. A complete overview of the FIPA
33 specifications and their current status may be found in the FIPA List of Specifications. A list of terms and abbreviations
34 used in the FIPA specifications may be found in the FIPA Glossary.

35 FIPA is a non-profit association registered in Geneva, Switzerland. As of January 2000, the 56 members of FIPA
36 represented 17 countries worldwide. Further information about FIPA as an organization, membership information, FIPA
37 specifications and upcoming meetings may be found at <http://www.fipa.org/>.

38 Contents

39	1	Scope	1
40	2	Overview.....	2
41	2.1	Reference Model.....	2
42	3	FIPA Message Buffering Service	3
43	3.1	Buffering Messages	3
44	3.2	Handling State Expiration Timeout	3
45	3.3	Handling Forward Timeout.....	3
46	3.4	Updating Message Envelope Information.....	3
47	3.5	Standard Interfaces.....	3
48	3.6	Proprietary Interfaces.....	3
49	3.7	Forwarding Messages.....	3
50	3.8	Handling a Single Receiver.....	3
51	3.9	Handling Multiple Transport Addresses for a Single Receiver	4
52	3.10	Handling Multiple Receivers.....	4
53	3.11	Delivering Messages	4
54	3.12	Using a Name Resolution Services.....	4
55	3.13	Error Messages	4
56	4	Message Buffering Service Ontology.....	5
57	4.1	Object Descriptions	5
58	4.1.1	Buffer Space Description	5
59	4.1.2	Buffer Space Identifier	6
60	4.1.3	Destination Description	6
61	4.2	Function Descriptions.....	6
62	4.2.1	Reserve Buffer Space.....	7
63	4.2.2	Delete Buffer Space.....	7
64	4.2.3	Forward Message	7
65	4.2.4	Delete Messages	8
66	4.3	Exceptions.....	8
67	4.3.1	Not Understood Exception Propositions.....	8
68	4.3.2	Refusal Exception Propositions	8
69	4.3.3	Failure Exception Propositions	8
70	5	References	9
71	6	Annex A — Informative Examples.....	10
72	6.1	Support for Disconnected Mode	10
73	6.2	Support for Roaming	14

74 **1 Scope**

75 This document is part of the FIPA specifications and deals with message buffering between inter-operating agents.
76 This document also forms part of the FIPA Message Transport Service Specification [FIPA00067] and contains
77 specification for:

78
79 Message buffering of FIPA messages.
80

81 The document provides a series of examples to illustrate the agent management functions defined.
82

83 2 Overview

84 The FIPA Message Buffering Service (FIPA-MBS) provides explicit FIPA-message buffering when a particular
 85 agent/agent platform cannot be reached¹. It allows an agent and/or an agent platform to explicitly apply for message
 86 buffering. FIPA-MBS is especially useful in cases where an agent and/or an agent platform is situated on a weakly
 87 connected device that does not have a physical connection to the fixed network at all times. Although FIPA-MBS is
 88 designed primarily for wireless environments, it also can be used in wireline environments. The FIPA Message Buffer
 89 (MB) implements the Message Buffering Service. The MB does not have to be a part of any agent platform, but it may.
 90 Application agents do not have to be aware of FIPA-MBS, but the underlying agent platform can take care of the
 91 details in order to enable buffering as well as requesting message forwarding.

92
 93 The FIPA-MBS allows roaming between Message Buffers. This allows, for example, the usage of dynamic addresses
 94 for the agents situated in the weakly connected devices.

95
 96 The specification contains features that may weaken the messaging security. These issues, however, are not explicitly
 97 discussed in the specification.
 98

99 2.1 Reference Model

100 The FIPA Message Buffer is logically situated between two APs (see *Figure 1*). The Message Buffer can be a
 101 standalone FIPA-addressable entity, that is, something that does not necessarily belong to any physical AP, but it also
 102 can be part of an AP. Especially, the Message Buffer can be a part of either platform (A or B) in *Figure 1*. The actual
 103 location of the message buffer depends on the environment where it is employed.
 104

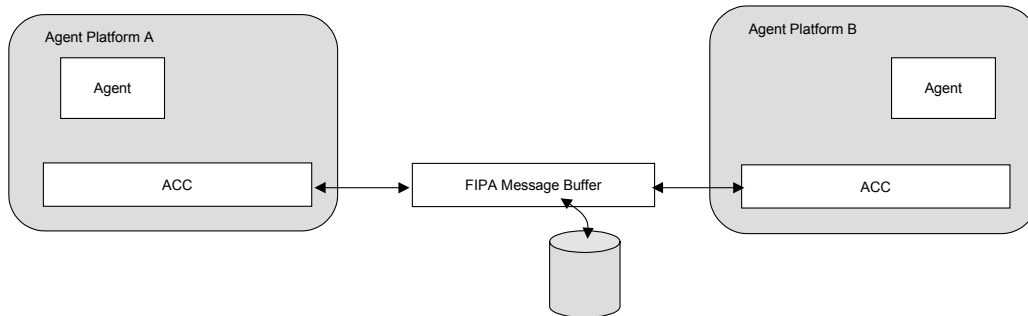


Figure 1: FIPA Message Buffering Service Reference Model

105

¹ Detecting connectivity is an implementation issue. One possibility is using the Monitor Agent as specified in [FIPA00014].

106 3 FIPA Message Buffering Service

107 3.1 Buffering Messages

108 The Message Buffer will buffer messages when requested to do so. Buffering will happen whenever there is no
109 connection to the next destination, and the address of the next destination is the one for which buffering has been
110 previously requested. If the message contains multiple receivers, the message is forwarded normally to those
111 receivers that can be reached.

112
113 Messages will be buffered even if the envelope contains reachable addresses to where the message could be
114 forwarded. If the destination that has requested the message buffering does not request the MB to forward the buffered
115 messages before the timeout expires (*keep-time*), messages are either forwarded to the next address in the
116 message envelope (if there is such an address) or an error message is sent as specified in [FIPA00067].

117
118 If the buffer space reserved for a given destination become full, the MB raises an error for each incoming message
119 destined to this address.
120

121 3.2 Handling State Expiration Timeout

122 A *buffer-space* object (see *Section 4.1*) may contain a state expiration timeout (*keep-time*) for buffered
123 messages. When this timeout expires, the MB acts like an ACC, that is, it either forwards the message to the next
124 address defined in the message envelope or it raises an error. The state expiration timer is started whenever the MB
125 buffers a message.
126

127 3.3 Handling Forward Timeout

128 A *buffer-space* object may contain a timeout (*forward-time*) for how long the MB will forward messages after it
129 has been requested to do so. The forwarding timer is started when the MB receives a forwarding request. After the
130 timer expires, the MB acts like a Message Transport Service, that is, it either forwards the messages to the next
131 address defined in the message envelope or it raises an error.
132

133 3.4 Updating Message Envelope Information

134 See [FIPA00067].
135

136 3.5 Standard Interfaces

137 See [FIPA00067].
138

139 3.6 Proprietary Interfaces

140 FIPA does not specify how agents communicate with the MB using proprietary interfaces.
141

142 3.7 Forwarding Messages

143 If the buffering is not needed, the MB acts like a Message Transport Service (see [FIPA00067]).
144

145 3.8 Handling a Single Receiver

146 If the buffering is not needed, the MB acts like a Message Transport Service (see [FIPA00067]).
147

148 **3.9 Handling Multiple Transport Addresses for a Single Receiver**

149 See [FIPA00067].

150

151 **3.10 Handling Multiple Receivers**

152 If the buffering is not needed for any of the receivers, the MB acts like a Message Transport Service (see
153 [FIPA00067]). If the buffering is needed for some destinations, the message(s) destined to these addresses are
154 buffered. For other destinations, the MB acts like an ACC.

155

156 **3.11 Delivering Messages**

157 See [FIPA00067].

158

159 **3.12 Using a Name Resolution Services**

160 See [FIPA00067].

161

162 **3.13 Error Messages**

163 See [FIPA00067].

164 4 Message Buffering Service Ontology

165 4.1 Object Descriptions

166 This section describes a set of frames that represent the classes of objects in the domain of discourse within the
167 framework of the FIPA-Message-Buffering ontology.

168
169 The following terms are used to describe the objects of the domain:

171 **Frame.** This is the mandatory name of this entity that must be used to represent each instance of this class.

172
173 **Ontology.** This is the name of the ontology, whose domain of discourse includes the parameters described in the
174 table.

175
176 **Parameter.** This is the mandatory name of a parameter of this frame.

177
178 **Description.** This is a natural language description of the semantics of each parameter.

179
180 **Presence.** This indicates whether each parameter is mandatory or optional.

181
182 **Type.** This is the type of the values of the parameter: Integer, Word, String, URL, Term, Set or Sequence.

183
184 **Reserved Values.** This is a list of FIPA-defined constants that can assume values for this parameter.

185

186 4.1.1 Buffer Space Description

187 This type of object represents the properties of a buffer space.

188

Frame	buffer-space-description			
Ontology	FIPA-Message-Buffering			
Parameter	Description	Presence	Type	Reserved Values
max-messages	Maximum number of messages that MB can buffer. This value must be positive.	Optional	Integer	
max-size	Maximum number of bytes that MB can buffer. This value must be positive.	Optional	Integer	
forward-time	Timeout (in seconds) the MB will forward messages after forward request (see <i>Section 3.3 Handling Forward Timeout</i>). This value must not be negative.	Optional	Integer	
keep-time	Maximum time (in seconds) the messages are buffered (state expiration timeout) (see <i>Section 4.2.2 Handling State Expiration Timeout</i>). This value must be positive.	Optional	Integer	
force-buffering	Forces message buffering even if there is a connection between the MB and the message destination.	Optional	Boolean	true false

189

190 If the `buffer-space-description` object does not contain the `max-messages` or the `max-size` parameter, the
191 size of the buffer depends on service defaults. However, because of physical limits, it may happen that the buffer
192 overflows, and the MB must raise an error (i.e., the entity that requested buffering, may still have to be prepared for
193 lost messages because of possible buffer overflow). If both parameters—`max-messages` and `max-size`—are
194 defined, then the actual buffer space is the minimum of these two. For example, if the value of the `max-messages`

195 parameter is 2 and the value of the `max-size` parameter is 1024, the buffer space cannot hold even one message, if
 196 the message size is more than 1024 bytes.

197
 198 If either the `keep-time` or the `forward-time` parameter is missing from the `buffer-space` object, corresponding
 199 timeout depends on service defaults.

200
 201 The `force-buffering` parameter defines whether the messages must be buffered even if the message destination
 202 is reachable. By default, messages are not buffered if the destination is reachable.

203

204 **4.1.2 Buffer Space Identifier**

205 This type of object represents the identification of the buffer space.

206

Frame Ontology	buffer-space-identifier FIPA-Message-Buffering			
Parameter	Description	Presence	Type	Reserved Values
id	A unique identifier for the buffer space. The identifier is unique only in one MB.	Mandatory	String	

207
 208 The MB implementation determines how the identifiers are constructed.

209

210 **4.1.3 Destination Description**

211 This type of object represents the identification of a message destination.

212

Frame Ontology	destination FIPA-Message-Buffering			
Parameter	Description	Presence²	Type	Reserved Values
address	Defines the destination address.	Optional	URL	
aid	Defines the destination agent.	Optional	agent-identifier (see [FIPA00023])	

213

214 **4.2 Function Descriptions**

215 The following tables define usage and semantics of the functions that are part of the `FIPA-Message-Buffering`
 216 ontology.

217

218 The following terms are used to describe the functions of the `FIPA-Message-Buffering` domain:

219

220 **Function.** This is the symbol that identifies the function in the ontology.

221

222 **Ontology.** This is the name of the ontology, whose domain of discourse includes the function described in the
 223 table.

224

225 **Supported by.** This is the type of agent that supports this function.

226

227 **Description.** This is a natural language description of the semantics of the function.

228

229 **Domain.** This indicates the domain over which the function is defined. The arguments passed to the function must
 230 belong to the set identified by the domain.

231

² While both of these parameters are optional, a valid `destination` object should contain at least one parameter

Range. This indicates the range to which the function maps the symbols of the domain. The result of the function is a symbol belonging to the set identified by the range.

Arity. This indicates the number of arguments that a function takes. If a function can take an arbitrary number of arguments, then its arity is undefined.

4.2.1 Reserve Buffer Space

Function	reserve-buffer
Ontology	FIPA-Message-Buffering
Supported by	FIPA-MB
Description	An agent can request a buffer space for messages that might be destined to it while the agent cannot be reached (for example, because of a disconnection). The argument <code>buffer-space-description</code> defines the requirements for buffer space (e.g., how much buffer space is needed and for how long time). If the sender does not want to specify requirements for buffer space, the <code>buffer-space-description</code> can be left empty. In this case, properties of buffer space depend on the service defaults. The argument <code>destination</code> specifies the destination address of the messages to be buffered.
Domain	<code>buffer-space-description</code> , <code>destination</code>
Range	<code>buffer-space-identifier</code>
Arity	2

4.2.2 Delete Buffer Space

Function	delete-buffer
Ontology	FIPA-Message-Buffering
Supported by	FIPA-MB
Description	An agent can request a Message Buffer to discard all the messages buffered in a given buffer space and also that new messages should not be buffered. An error message is sent to the originator agent of each discarded message.
Domain	<code>buffer-space-identifier</code>
Range	The execution of this function results in a change of the state, but it has no explicit result. Therefore there is no range set.
Arity	1

4.2.3 Forward Message

Function	forward
Ontology	FIPA-Message-Buffering
Supported by	FIPA-MB
Description	An agent can request a Message Buffer to forward all or some of the buffered messages to the given destination. The argument <code>buffer-space-identifier</code> specifies the buffer space from which messages are to be forwarded and the argument <code>destination</code> specifies the destination to where the messages should be forwarded.
Domain	<code>buffer-space-identifier</code> , <code>destination</code>
Range	The execution of this function results in a change of the state, but it has no explicit result. Therefore there is no range set.
Arity	2

244 **4.2.4 Delete Messages**

Function	delete
Ontology	FIPA-Message-Buffering
Supported by	FIPA-MB
Description	An agent can request a Message Buffer to delete all of the buffered messages. An error message is sent to the original sender of each deleted message.
Domain	buffer-space-identifier
Range	The execution of this function results in a change of the state, but it has no explicit result. Therefore there is no range set.
Arity	1

245

246 **4.3 Exceptions**

247 The exceptions for the FIPA-Message-Buffering ontology follow the same form and rules as specified in
 248 [FIPA00023].

249

250 **4.3.1 Not Understood Exception Propositions**

251 The same set of “*Not Understood Exception Propositions*” as in the FIPA-Agent-Management ontology is used in
 252 the FIPA-Message-Buffering ontology (see [FIPA00023]).

253

254 **4.3.2 Refusal Exception Propositions**

255 The same set of “*Refusal Exception Propositions*” as defined in the FIPA-Agent-Management ontology is used in
 256 FIPA-Message-Buffering ontology (see [FIPA00023]). In addition, the FIPA-Message-Buffering ontology
 257 defines the propositions given below.

258

Communicative Act Ontology	refuse FIPA-Message-Buffering	
Predicate symbol	Arguments	Description
size-value-too-large	String	The agent has requested more buffer space than the MB allows for one agent
forward-time-too-long		The agent has requested too long forward-time timeout.
keep-time-too-long		The agent has requested too long keep-time timeout.
force-buffering-not-supported		The agent has requested message buffering even if it is reachable, but the MB does not support this functionality

259

260 **4.3.3 Failure Exception Propositions**

Communicative Act Ontology	failure FIPA-Message-Buffering	
Predicate symbol	Arguments	Description
internal-error	String	See [FIPA00023].
allocation-failed	String	The allocating a buffer space failed; the string identifies failure reason.
forwarding-failed	String	The forwarding a message failed; the string identifies failure reason.
unknown-identifier		The buffer-space-identifier is not known.

261 **5 References**

262 [FIPA00014] FIPA Nomadic Application Support Specification. Foundation for Intelligent Physical Agents, 2000.
263 <http://www.fipa.org/specs/fipa00014/>

264 [FIPA00023] FIPA Agent Management Specification. Foundation for Intelligent Physical Agents, 2000.
265 <http://www.fipa.org/specs/fipa00023/>

266 [FIPA00067] FIPA Agent Message Transport Service Specification. Foundation for Intelligent Physical Agents,
267 2000. <http://www.fipa.org/specs/fipa00067/>
268

269 6 Annex A — Informative Examples

270 6.1 Support for Disconnected Mode

271 This example shows how the Message Buffering Service may support the disconnected mode of operation. The
 272 message flow is illustrated in the *Figure 4*.

- 273
- 274 1. Message [1]: The agent *dummy* (located at a mobile device) is receiving messages from agents located at the
 275 fixed network.
 - 276
 - 277 2. Message [2] *request*: In order to be sure that no message is lost during a possible disconnection, the agent
 278 *dummy* applies to the Message Buffer to buffer the messages if it cannot be reached. The agent *dummy* requests
 279 a buffer space for 100 messages, with a state expiration timeout of 120 seconds:

```

280
281 (request
282   :sender
283     (agent-identifier
284      :name dummy
285      :addresses (sequence http://helluli.com/acc))
286   :receiver (set
287     (agent-identifier
288      :name message-buffer
289      :addresses (sequence http://buffer.com/acc)))
290   :ontology FIPA-Message-Buffering
291   :language fipa-s10
292   :protocol fipa-request
293   :content
294     (action (agent-identifier :name message-buffer)
295      (reserve-buffer
296       (buffer-space-description
297        :max-messages 100
298        :keep-time 120)
299       (destination :address http://helluli.com/acc))))
300
    
```

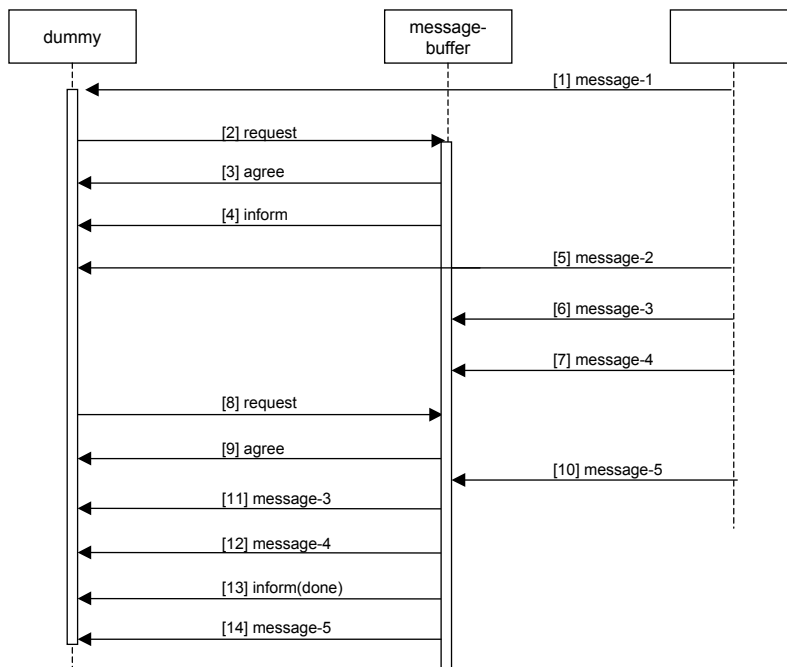


Figure 2: Support for Disconnected Mode

301 3. Message [3] agree: The Message Buffer agrees to reserve a buffer space:

```
302
303 (agree
304   :sender
305     (agent-identifier
306      :name message-buffer
307      :addresses (sequence http://buffer.com/acc))
308   :receiver (set
309     (agent-identifier
310      :name dummy
311      :addresses (sequence http://helluli.com/acc)))
312   :ontology FIPA-Message-Buffering
313   :language fipa-s10
314   :protocol fipa-request
315   :content
316     ((action (agent-identifier :name message-buffer)
317      (reserve-buffer
318       (buffer-space-description
319        :max-messages 100
320        :keep-time 120)
321       (destination :address http://helluli.com/acc))))
322     true))
323
```

324 4. Message [4] inform: The Message Buffer informs the agent *dummy* that buffer space is reserved with an
325 identifier *buffer-3*:

```
326
327 (inform
328   :sender
329     (agent-identifier
330      :name message-buffer
331      :addresses (sequence http://buffer.com/acc))
332   :receiver (set
333     (agent-identifier
334      :name dummy
335      :addresses (sequence http://helluli.com/acc)))
336   :ontology FIPA-Message-Buffering
337   :language fipa-s10
338   :protocol fipa-request
339   :content
340     (result
341      (action (agent-identifier :name message-buffer)
342       (reserve-buffer
343        (buffer-space-description
344         :max-messages 100
345         :keep-time 120)
346        (destination :address http://helluli.com/acc))))
347      (buffer-space-identifier :id buffer-3))
348
```

349 5. Message [5]: Messages coming from the fixed network are still forwarded to the agent *dummy*:

350

351 6. Messages [6] and [7]: During a disconnection (when the agent *dummy* cannot be reached anymore) the messages
352 are buffered.

353

354

355

7. Message [8] request: The agent *dummy* requests the Message Buffer to forward all the buffered messages:

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

```
(request
  :sender
    (agent-identifier
      :name dummy
      :addresses (sequence http://helluli.com/acc))
  :receiver (set
    (agent-identifier
      :name message-buffer
      :addresses (sequence http://buffer.com/acc)))
  :ontology FIPA-Message-Buffering
  :language fipa-sl0
  :protocol fipa-request
  :content
    (action (agent-identifier :name message-buffer)
      (forward
        (buffer-space-identifier :id buffer-3)
        (destination :address http://helluli.com/acc))))))
```

375

8. Message [9] agree: The Message Buffer agrees:

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

```
(agree
  :sender
    (agent-identifier
      :name message-buffer
      :addresses (sequence http://buffer.com/acc))
  :receiver (set
    (agent-identifier
      :name dummy
      :addresses (sequence http://helluli.com/acc)))
  :ontology FIPA-Message-Buffering
  :language fipa-sl0
  :protocol fipa-request
  :content
    ((action (agent-identifier :name message-buffer)
      (forward
        (buffer-space-identifier :id buffer-3)
        (destination :address http://helluli.com/acc)))
      true))
```

396

397

398

9. Message [10]: A new message arrives from the fixed network. The Message Buffer does not forward this message until all the messages are forwarded from the buffer in order to preserve message ordering.

399

400

401

10. Messages [11] and [12]: The Message Buffer forwards the messages ([6] and [7]) that were buffered while the agent *dummy* was unreachable.

402 11. Message [13] inform: The Message Buffer informs the agent *dummy* that all messages are now forwarded:

```
403 (inform
404   :sender
405     (agent-identifier
406      :name message-buffer
407      :addresses (sequence http://buffer.com/acc))
408   :receiver (set
409     (agent-identifier
410      :name dummy
411      :addresses (sequence http://helluli.com/acc)))
412   :ontology FIPA-Message-Buffering
413   :language fipa-sl0
414   :protocol fipa-request
415   :content
416     (done (action (agent-identifier :name message-buffer)
417      (forward
418       (buffer-space-identifier :id buffer-3)
419       (destination :address http://helluli.com/acc))))))
```

421
422 12. Message [14]: Finally, the Message Buffer forwards the message [10] to the agent *dummy*.
423

6.2 Support for Roaming

This example shows how the Message Buffer may support roaming from one Message Buffer to another. In this example, the agent changes its transport address. The message flow is illustrated in *Figure 5*.

1. Message [1]: The agent *dummy* (located at a mobile device) is receiving messages from agents located at the fixed network.
2. Message [2] *request*: The agent *dummy* applies to the MB₁ (located at *helluli.com*) to buffer the messages in the case of disconnection:

```

434 (request
435   :sender
436     (agent-identifier
437       :name dummy
438       :addresses (sequence http://helluli.com/acc))
439   :receiver (set
440     (agent-identifier
441       :name message-buffer1
442       :addresses (sequence http://buffer.com/acc)))
443   :ontology FIPA-Message-Buffering
444   :language fipa-sl0
445   :protocol fipa-request
446   :content
447     (action (agent-identifier :name message-buffer1)
448       (reserve-buffer
449         (buffer-space-description
450           :max-messages 100
451           :keep-time 120)
452         (destination :address http://helluli.com/acc))))

```

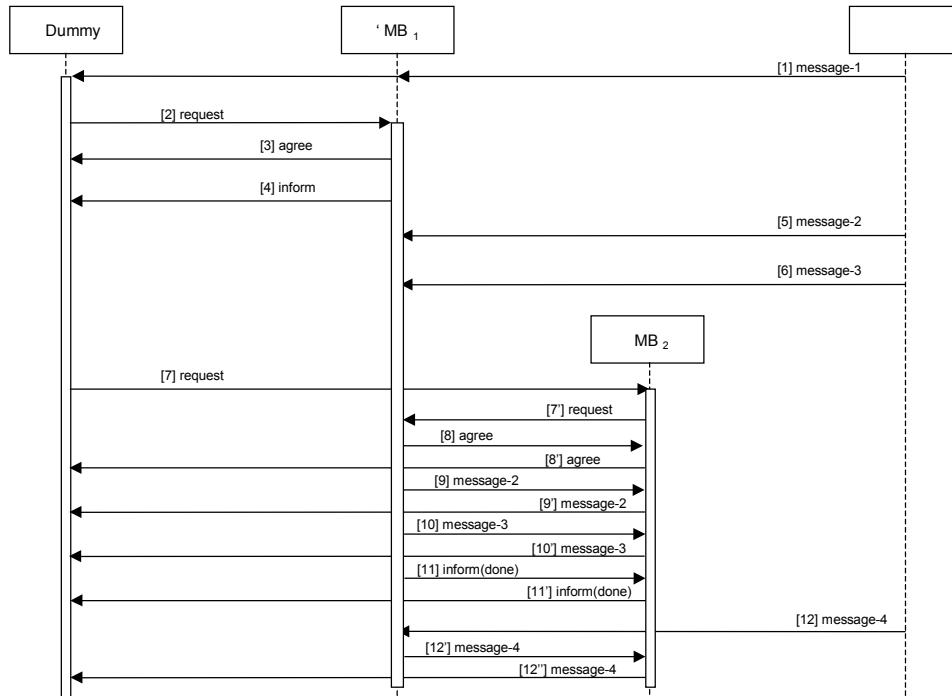


Figure 3: Roaming from one Message Buffer to Another

- 454 3. Message [3] agree and Message [4] inform: The MB₁ agrees and informs that buffer space is set up (with
 455 buffer-space-identifier is fool).
 456
- 457 4. Messages [5] and [6]: The MB₁ buffers incoming messages while the agent *dummy* is unreachable.
 458
- 459 5. The agent *dummy* establishes a new connection to the fixed network, but using a different access node. At the
 460 same time, the agent *dummy* changes its transport address. Let us assume that the new address is
 461 wap://helluli.com/acc.
 462
- 463 6. Messages [7] and [7'] request: The agent *dummy* requests the MB₁ to forward all the buffered messages to its
 464 new address (the message goes though the MB₂):
 465
 466 (request
 467 :sender
 468 (agent-identifier
 469 :name dummy
 470 :addresses (sequence wap://helluli.com/acc))
 471 :receiver (set
 472 (agent-identifier
 473 :name message-buffer1
 474 :addresses (sequence http://buffer.com/acc)))
 475 :ontology FIPA-Message-Buffering
 476 :language fipa-sl0
 477 :protocol fipa-request
 478 :content
 479 (action
 480 (agent-identifier :name message-buffer1)
 481 (forward
 482 (buffer-space-identifier :id fool)
 483 (destination :address wap://helluli.com/acc))))
 484
- 485 7. Messages [8] and [8'] agree: The MB₁ agrees (through the MB₂).
 486
- 487 8. Messages [9], [9'], [10], [10']: The MB₁ sends the buffered messages (through the MB₂).
 488
- 489 9. Messages [11] and [11'] inform: The MB₁ informs that all the buffered messages are forwarded.
 490
- 491 10. Messages [12], [12'], [12'']: The MB₁ will forward all the messages to the new address until the forward-time
 492 timeout expires.
 493