FOUNDATION FOR INTELLIGENT PHYSICAL AGENTS

FIPA Message Buffering Service Specification

Document title	FIPA Message Buffering Service Specification		
Document number	PC00092A	**Document source**	FIPA TC Gateways
Document status	Preliminary	**Date of this status**	2001/08/10
Supersedes	None		
Contact	gateways@fipa.org		
Change history			
2001/07/25	Initial version		
2001/08/10	Line numbering added		

Geneva, Switzerland

Foreword

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

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

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

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

Contents

1 Scope

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

Message buffering of FIPA messages.

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

2 Overview

The FIPA Message Buffering Service (FIPA-MBS) provides explicit FIPA-message buffering when a particular agent/agent platform cannot be reached[1]. It allows an agent and/or an agent platform to explicitly apply for message buffering. FIPA-MBS is especially useful in cases where an agent and/or an agent platform is situated on a weakly connected device that does not have a physical connection to the fixed network at all times. Although FIPA-MBS is designed primarily for wireless environments, it also can be used in wireline environments. The FIPA Message Buffer (MB) implements the Message Buffering Service. The MB does not have to be a part of any agent platform, but it may. Application agents do not have to be aware of FIPA-MBS, but the underlying agent platform can take care of the details in order to enable buffering as well as requesting message forwarding.

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

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

2.1 Reference Model

The FIPA Message Buffer is logically situated between two APs (see *Figure 1*). The Message Buffer can be a standalone FIPA-addressable entity (i.e., something that does not necessarily belong to any physical AP), but it also 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 location of the message buffer depends on the environment where it is employed.

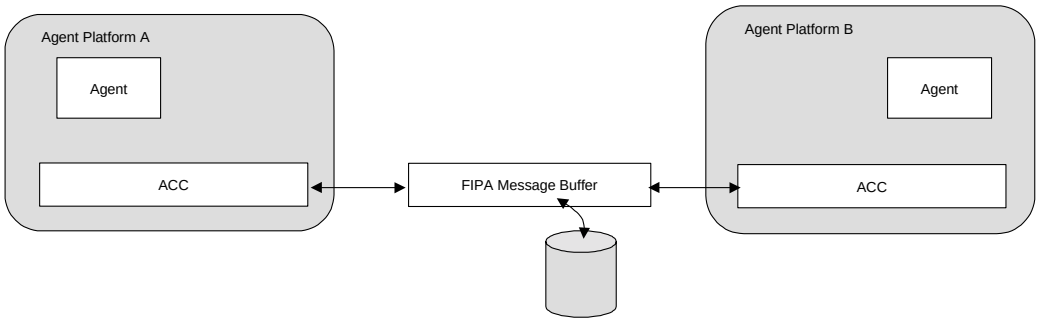

Figure 1: FIPA Message Buffering Service Reference Model

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

3 FIPA Message Buffering Service

3.1 Buffering Messages

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

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

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

3.2 Handling State Expiration Timeout

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

3.3 Handling Forward Timeout

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

3.4 Updating Message Envelope Information

See [FIPA00067].

3.5 Standard Interfaces

See [FIPA00067].

3.6 Proprietary Interfaces

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

3.7 Forwarding Messages

If the buffering is not needed, the MB acts like an ACC (see [FIPA00067]).

3.8 Handling a Single Receiver

If the buffering is not needed, the MB acts like an ACC (see [FIPA00067]).

3.9 Handling Multiple Transport Addresses for a Single Receiver

See [FIPA00067].

3.10 Handling Multiple Receivers

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

3.11 Delivering Messages

See [FIPA00067].

3.12 Using a Name Resolution Services

See [FIPA00067].

3.13 Error Messages

See [FIPA00067].

4 Message Buffering Service Ontology

4.1 Object Descriptions

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

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

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

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

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

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

Presence. This indicates whether each parameter is mandatory or optional.

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

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

4.1.1 Buffer Space Description

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

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 *Section 3.3 Handling Forward Timeout*). This value must not be negative.	Optional	`Integer`	
`keep-time`	Maximum time (in seconds) the messages are buffered (state expiration timeout) (see Section *4.2.2 Handling State Expiration Timeout*). 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`

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

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

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

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

4.1.2 Buffer Space Identifier

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

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`	

The MB implementation determines how the identifiers are constructed.

4.1.3 Destination Description

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

Frame Ontology	`destination` `FIPA-Message-Buffering`			
Parameter	**Description**	**Presence**[2]	**Type**	**Reserved Values**
`address`	Defines the destination address.	Optional	`URL`	
`aid`	Defines the destination agent.	Optional	`agent-identifier` (See [FIPA00023])	

4.2 Function Descriptions

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

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

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

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

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

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

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

[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 `buffer-space-description` 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 `buffer-space-description` can be left empty. In this case, properties of buffer space depend on the service defaults. The argument `destination` specifies the destination address of the messages to be buffered.
Domain	`buffer-space-description, destination`
Range	`buffer-space-identifier`
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	`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

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 `buffer-space-identifier` specifies the buffer space from which messages are to be forwarded and the argument `destination` specifies the destination to where the messages should be forwarded.
Domain	`buffer-space-identifier, destination`
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

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

4.3 Exceptions

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

4.3.1 Not Understood Exception Propositions

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

4.3.2 Refusal Exception Propositions

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

Communicative Act	`refuse`	
Ontology	`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.
`keeptime-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

4.3.3 Failure Exception Propositions

Communicative Act	`failure`	
Ontology	`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.

5 References

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

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

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

6 Annex A — Informative Examples

6.1 Support for Disconnected Mode

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

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`: In order to be sure that no message is lost during a possible disconnection, the agent *dummy* applies to the Message Buffer to buffer the messages if it cannot be reached. The agent *dummy* requests a buffer space for 100 messages, with a state expiration timeout of 120 seconds:

```
(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)
      (reserve-buffer
        (buffer-space-description
            :max-messages 100
            :keep-time 120)
        (destination :address http://helluli.com/acc))))
```

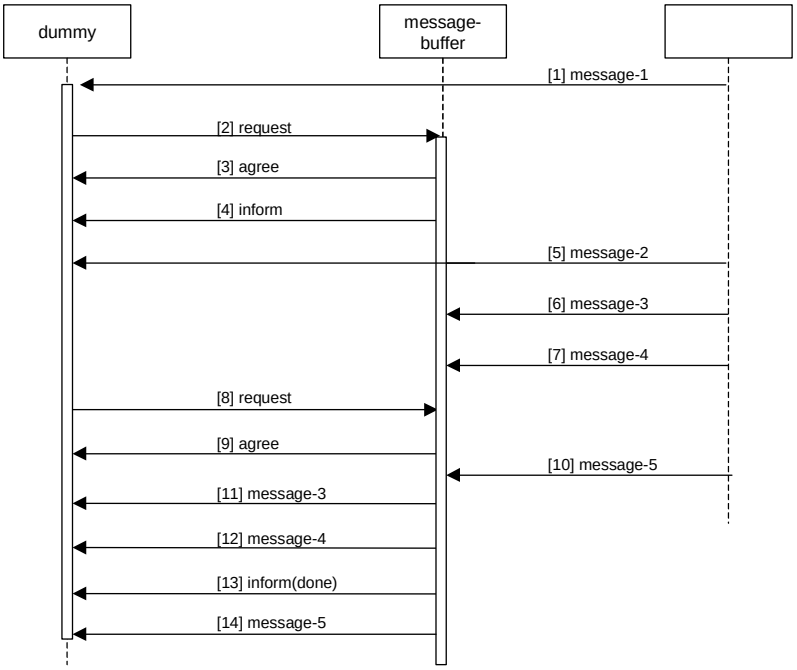

Figure 2: Support for disconnected mode

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

```
(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)
      (reserve-buffer
        (buffer-space-description
            :max-messages 100
            :keep-time 120)
        (destination :address http://helluli.com/acc)))
       true))
```

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

```
(inform
  :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
    (result
      (action (agent-identifier :name message-buffer)
        (reserve-buffer
          (buffer-space-description
              :max-messages 100
              :keep-time 120)
          (destination :address http://helluli.com/acc)))
      (buffer-space-identifier :id buffer-3))
```

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

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

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

```
(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))))
```

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

```
(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))
```

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.

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

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

```
(inform
  :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
    (done (action (agent-identifier :name message-buffer)
      (forward
        (buffer-space-identifier :id buffer-3)
        (destination :address http://helluli.com/acc)))))
```

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

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 MB_1 (located at `helluli.com`) to buffer the messages in the case of disconnection:

```
(request
  :sender
    (agent-identifier
      :name dummy
      :addresses (sequence http://helluli.com/acc))
  :receiver (set
    (agent-identifier
      :name message-buffer1
      :addresses (sequence http://buffer.com/acc)))
  :ontology FIPA-Message-Buffering
  :language fipa-sl0
  :protocol fipa-request
  :content
    (action (agent-identifier :name message-buffer1)
      (reserve-buffer
        (buffer-space-description
            :max-messages 100
            :keep-time 120)
        (destination :address http://helluli.com/acc))))
```

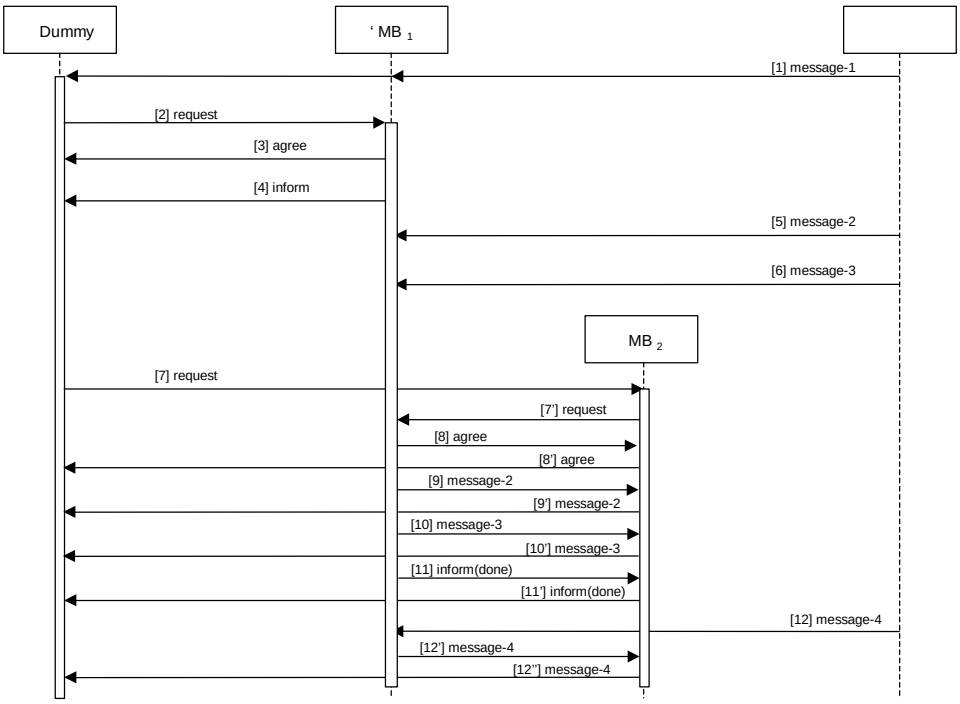

Figure 3: Roaming from one Message Buffer to another

3. Message [3] `agree` and Message [4] `inform`: The MB_1 agrees and informs that buffer space is set up (with `buffer-space-identifier` is `foo1`).

4. Messages [5] and [6]: The MB_1 buffers incoming messages while the agent *dummy* is unreachable.

5. The agent *dummy* establishes a new connection to the fixed network, but using a different access node. At the same time, the agent *dummy* changes its transport address. Let us assume that the new address is `wap://helluli.com/acc`.

6. Messages [7] and [7'] `request`: The agent *dummy* requests the MB_1 to forward all the buffered messages to its new address (the message goes though the MB_2):

```
request
  :sender
    (agent-identifier
      :name dummy
      :addresses (sequence wap://helluli.com/acc))
  :receiver (set
    (agent-identifier
      :name message-buffer1
      :addresses (sequence http://buffer.com/acc)))
  :ontology FIPA-Message-Buffering
  :language fipa-sl0
  :protocol fipa-request
  :content
    (action
      (agent-identifier :name message-buffer1)
      (forward
         (buffer-space-identifier :id foo1)
         (destination :address wap://helluli.com/acc))))
```

7. Messages [8] and [8'] `agree`: The MB_1 agrees (through the MB_2).

8. Messages [9], [9'], [10], [10']: The MB_1 sends the buffered messages (through the MB_2).

9. Messages [11] and [11'] `inform`: The MB_1 informs that all the buffered messages are forwarded.

10. Messages [12], [12'], [12''']: The MB_1 will forward all the messages to the new address until the `forward-time` timeout expires.