X11/96-31

MUMPS Development Committee

Extension to the MDC Standard
Type A Release of the MUMPS Development Committee

OMI Multiple-Transaction Message

March 23, 1996

Produced by the MDC Subcommittee #14
Networking and Communications

Ed de Moel, Chairman
MUMPS Development Committee

Fred Hiltz, Chairman
Subcommittee #14

1. Identification of the Proposed Change

1.1 Title

OMI Multiple-Transaction Message

1.2 MDC Proposer and Sponsor

Proposer
Subcommittee 14
Networking and Communications
Chairman: Frederick L. Hiltz

Sponsor
Frederick L. Hiltz
Brigham and Women's Hospital
10 Vining Street
Boston, MA 02115-6198
617-732-7028
fhiltz@bics.bwh.harvard.edu

1.3 Motion

None. Final version approved by MDC on March 23, 1996. This document supersedes X11/SC14/95–14.

1.4 History

31 Mar 1996	X11/96–31	This document.
23 Mar 1996	X11/SC14/95–14	Accepted by MDC as type A, 21:0:6.
25 Oct 1995	X11/SC14/TG4/95–9	SC14 accepted as Type A, 6:0:2.
03 Jun 1995	X11/SC14/TG4/95–4	Original proposal by John Althouse and Frederick Hiltz. SC14 accepted as Type B, 7:0:0.

1.5 Dependencies

This proposal is dependent upon:
ANSI/MDC X11.2–1995, OMI.

Proposals depending on this proposal: none.

2. Justification of the Proposed Change

2.1 Needs

OMI needs faster performance.

2.2 Existing Practice in Area of the Proposed Change

This change has been implemented by DataTree and Greystone in EMI, an extension to OMI version 1.1.

3. Description of the Proposed Change

3.1 General Description of the Proposed Change

The change permits more than one request in a request message and more than one response in a response message, sharing the overhead of message transmission among several transactions. The OMI major version number changes from 1 to 2.

3.2 Annotated Examples of Use

None. The OMI protocol is invisible to application routines.

3.3 Formalization

Amend ANSI/MDC X11.2–1995, OMI as shown in Appendix 9.

4. Implementation Effects

4.1 Effect on Existing User Practices and Investments

None. Existing routines need not be changed.

4.2 Effect on Existing Vendor Practices and Investments

The extension from OMI version 1 to version 2 requires minor changes in the areas of OMI version negotiation and the buffering of multiple transactions.

4.3 Techniques and Costs for Compliance Verification

A verification facility for OMI compliance must be extended to verify multiple transactions per message.

4.4 Legal Considerations

None.

5. Closely Related Standards Activities

5.1 Other X11 Proposals Under Consideration

5.2 Other Related Standards Efforts

OMI Extended Get and Order, X11/SC14/ 96–1, also improves OMI performance.

5.3 Recommendations for Coordinating Liaison

None.

6. Associated Documents

None

7. Issues, Pros and Cons, and Discussion

7.1 June 1995 MDC meeting

Pro	Con
1 More efficient use of network (0)	

7.2 October 1995 MDC meeting

Pro	Con
1 Better performance of OMI (1)	

7.3 March 1996 MDC meeting

Pro	Con
1 Speeds up OMI (1)	
2 Is implemented (1)	

(Number of citations in the vote.)

8. Glossary

None.

9. Appendix

3 Definitions

3.4 message: a string of 8-bit characters, allowing all character codes from 0 to 255. A request message contains 1 or more requests, and a response message contains the related responses, exactly 1 response for each request.

3.6 client: a process that originates ~~messages (~~requests~~)~~ to be transmitted to a server. Most OMI clients are MUMPS application processes, but a network manager utility program or a non-MUMPS application program could be a client.

3.7 server: a process that responds to clients' requests by performing database functions on their behalf and returning responses~~ messages~~.

3.8 transaction: ~~one~~1 request ~~message~~ and the related response~~ message~~. An OMI transaction is not the same as a transaction processing transaction.

4.3.2 The role of the agent

... The agent is synchronous. A session supports only ~~one transaction~~ 1 request message at a time. (The message may contain several requests.) The agent shall send ~~one~~1 request message and then no more until the server's response message arrives or the circuit fails.

4.3.3 Transactions

An agent shall originate each transaction with exactly ~~one~~1 request~~ message~~. The server shall attempt the requested operation and shall reply with exactly ~~one~~ 1 response~~ message~~, which may indicate failure to perform the operation.

~~NOTE — In anticipation of future versions of the protocol, which may permit many requests and responses in a transaction, messages include sequence numbers and request identifiers.~~

When a request message contains more than 1 request, the requests shall appear in the order of their sequence numbers (see 5.3.1). The server shall process the requests in that order and shall reply with a response message containing the responses, 1 for each request, in that order.

4.5.4 Identification

... Each ~~message~~ request shall identify the user. Together, the agent's name and the user's identification furnish proof of origin, that is, the server "knows" the origin of a request. Completion of the transaction provides the agent with proof of delivery of the request.

4.6 Replication

... – send the same request with replication disabled to each of the destinations, in effect saying, "This is a replication ~~message~~ request. Do not replicate it further."

5.3 Message form

The requests and responses of all transactions except *connect* shall be collected into request messages and response messages respectively. These messages shall be very long strings comprising the following fields:

a *Request count* or *response count:* <LI> the number of requests or responses that follow, not less than 1.

b *Request(s)* or *response(s):* <VS> 1 or more requests or responses, in the order of their sequence numbers.

Requests and responses are themselves very long strings so that their boundaries can be clearly distinguished. Both comprise a header followed by 0 or more of the fields described in 5.2, but the forms of their headers differ.

Connect requests and responses shall not be placed inside messages, but shall be transmitted in the forms shown for requests and responses. These are the forms of OMI Version 1, retained in Version 2 to allow negotiation of the version during session establishment.

All other requests and responses shall be placed inside messages.

~~OMI messages comprise a header followed by 0 or more of the fields described in 5.2. The 2 types of OMI messages — request and response — have different headers. The number and form of the following fields depend on the operation class and type. The overall message is a very long string <VS> so that message boundaries can be clearly distinguished.~~

[Strike Figure 1, "Form of a message" and insert the following four figures in its place.]

	<VS>			
<VI>	<LI>	<VS>	<VS>	
	Request Count	Request	Request	...

Figure 1 – Request message

	<VS>			
<VI>	<LI>	<VS>	<VS>	
	Response Count	Response	Response	...

Figure 2 – Response message

	<VS>	
<VI>	Request Header	Operation Specific Fields

Figure 3 – Form of a request

	<VS>	
<VI>	Response Header	Operation Specific Fields

Figure 4 – Form of a response

Table 1 – Operation types

Type	Name	Description
		Global fetch operations
20	Get	Requests the value of a global variable. Returns the value, or the empty string if the variable is undefined. A status flag indicates if the variable was defined, so that the agent can use this ~~message~~ request for either the MUMPS $Get function or a normal global fetch.

Table 2 – Error conditions

Type	Name	Description
		Database errors
—	Undefined global	Signaled in the Define field of the Get response ~~message~~.
—	Lock not granted	Signaled in the Lock Granted field of the Lock response ~~message~~.
		Session Establishment errors
20	OMI version not supported	The OMI major version requested by the agent in the connect ~~message~~ request is not supported on the OMI server. The agent may try again with another version number.

5.4 Requests and responses

A complete request or response ~~message~~ is a <VS>, whose string comprises the request or response header followed by operation-specific fields defined here.

5.4.1 Connect

(Operation type 1) The fields of the *connect* request and their sequence shall be:

a *Major version:* <SI> the major version of the OMI protocol supported by the agent. This document defines major version ~~1~~2. If the agent supports multiple versions, it shall send *connect* requests for each until the server accepts a major version.

A.1 Implementations

A *conforming implementation* shall

– correctly execute all ~~messages~~ requests and responses conforming to both this standard and the implementation-defined extensions to this standard;

B.q Multiple requests per message

[Insert this clause before "B.7 Registering implementations."]

The provision of many requests and many responses per message spreads the overhead of message transmission across many transactions, especially when the agent can combine requests from many client programs. Even for a single MUMPS client, many *set* and *kill* transactions may profitably be combined, although the MUMPS client must usually await the response to other types of operations before proceeding. Non-MUMPS clients, and perhaps highly optimized MUMPS clients, could, however, usefully combine other operations, for example, to request many items from the database at once.

C.4 Performance enhancements

The *connect* operation (5.4.1) now allows only ~~one~~1 request message outstanding in a session. Multiple request messages are anticipated, necessitating more complex error reporting and recovery mechanisms.

~~The agent could pack many requests into a message and the server could pack many responses, thus spreading the overhead of message transmission among many transactions. This, too, would require more complex error reporting and recovery.~~