MUMPS Development Committee

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

OMI JOB

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 JOB

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/SC14/95–13.

1.4 History

31 Mar 1996	X11/96–30	This document.
23 Mar 1996	X11/SC14/95–13	Accepted by MDC as type A, 21:0:6.
25 Oct 1995	X11/SC14/TG4/95–7	SC14 accepted as Type A, 5:0:3.
03 Jun 1995	X11/SC14/TG4/95–2	SC14 accepted as Type B, 7:0:0.
28 Jan 1995	X11/SC14/TG4/94–7	Original proposal by Peter Cudhea-Pierce and Frederick Hiltz. SC14 accepted as Type B, 7:0:0.

1.5 Dependencies

This proposal is dependent upon:
ANSI/MDC X11.1–1995, M;
ANSI/MDC X11.2–1995, OMI;
X11/95–2 Execution Environment.

Proposals depending on this proposal:
X11/SC14/ 96–2 OMI International.

2. Justification of the Proposed Change

2.1 Needs

Programs need to initiate processes on other nodes in a network.

2.2 Existing Practice in Area of the Proposed Change

DataTree M (InterSystems) permits processparameters to specify a node.

3. Description of the Proposed Change

3.1 General Description of the Proposed Change

Four requests extend the OMI protocol to initiate an M process on another node. The *job* request conveys the data of a JOB command, the *job status* request asks for the status of the last *job* request, the *cancel job* request cancels the last *job* request, and the *cancel all jobs* request cancels all incomplete job requests from a client node.

Because process initiation can take some time, the server does not wait, but responds immediately to a *job* request. Subsequent *job status* requests determine the success or failure of process initiation.

3.2 Annotated Examples of Use

None. The OMI JOB service is invisible to application routines.

3.3 Formalization

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

4.1 OMI and MUMPS

– *Job:* 8.2.10 initiates an M process.

– *processparameters:* 8.2.10 qualifies a JOB command.

– *label:* 6.2.3 identifies a line in a MUMPS routine.

– *routinename:* 6.1 identifies a MUMPS routine.

– *actuallist:* 8.1.7 a list of actual arguments passed to a subroutine or process.

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

OMI conveys information already required for the JOB command on a single computer; therefore little effect on the language processor of either client or server is expected.

The (possibly) delayed response to a *job* request does require the OMI server to retain a small amount of state information about one outstanding *job* request per client. If the server's operating system offers non-blocking process initiation, this should be straightforward. If not, the server will have to implement it.

4.3 Techniques and Costs for Compliance Verification

A verification facility for OMI compliance must be extended to verify the use of the messages specified herein. The extension may require a man-month of software development.

4.4 Legal Considerations

None.

5. Closely Related Standards Activities

5.1 Other X11 Proposals Under Consideration

None. Other proposals for which there are dependencies are listed in 1.5.

5.2 Other Related Standards Efforts

None.

5.3 Recommendations for Coordinating Liaison

None.

6. Associated Documents

Execution Environment, X11/95–2, specifies the syntax with which M routines may initiate M processes in other environments.

7. Issues, Pros and Cons, and Discussion

7.1 January 1995 MDC meeting

	Pro		Con
1	Most requested feature after OMI WRITE (0)	1	Server must keep information between transactions (0)
2	Needed for Execution environment (0)		

(Number of citations in the vote.)

7.2 June 1995 MDC meeting

Averting protocol blocking

Like the *lock* request, the *job* request can take an unbounded length of time to satisfy. In response to a client program's LOCK command, the agent may repeat a *lock* request until the server grants it or until it times out; however repeating a request for process initiation is not practical. In January 1995 the sponsors presented two protocols that avert blocking the OMI session while the server initiates a process. SC14/TG4 added B3, discussed the choices, and selected B1 by a straw poll, 4:0:1.

B1 This proposal presents a protocol in which the server checks the arguments of the job request and responds immediately. If the response shows that process initiation is pending, the agent follows up with job status requests until process initiation succeeds, times out, or fails. Requests to cancel job and to cancel all jobs terminate pending process initiation when the server can do so.

B2 The job request is a "best effort" request. The server checks its arguments, attempts process initiation, and responds immediately, indicating success, failure, or pending. There are no other transactions.

B3 Client chooses B1 for a timed JOB command, otherwise B2.

B1 requires the server to retain state information about *job* requests. The sponsors have minimized the amount of state information by limiting the protocol to one outstanding *job* request per client process. The server's procedure for initiating processes may not correspond closely with the transactions of B1, complicating the implementation. B1 does, however, provide the agent with complete information about the initiation of the process.

B2 is much simpler to define and to implement. It requires the M client program to determine the fate of the initiated process, probably communicating with it via LOCKs or global data. Many application programs do this anyway when "jobbing" processes on their own computer. B2 provides no support for the time out parameter of the JOB command, nor does it allow the client program to receive the $J of the new process, a (non-standard) feature of some M implementations.

Passing local variables

Some M implementations extend the language standard by starting a new M process with a copy of all the parent's local variables. Should OMI incorporate a standard means of passing the local variables' names and value? In January 1995, SC14/TG4 selected V2 by a straw poll, 1:5.

V1 Yes, so that mixed-vendor networks can offer the feature. Servers would indicate during session initiation whether they accept local variables.

V2 No, leave it to implementors to extend OMI, collaborating on a protocol if they want to support mixed-vendor networks.

Pros and cons

	Pro	Con
1	Continues work toward resolving canvass objection (0)	
2	Reflects latest changes (0)	

7.3 October 1995 MDC meeting

	Pro	Con
1	Continues work toward resolving canvass objection (1)	

7.4 March 1996 MDC meeting

	Pro	Con
1	Continues work toward resolving canvass objection (1)	

(Number of citations in the vote.)

8. Glossary

None.

9. Appendix: Amendments to OMI

Table 1 – Operation types

Type	Name	Description
61	Job	Requests initiation of a process on the server.
62	Job status	Requests the status of a preceding *job* request.
63	Cancel job	Cancels 1 incomplete *job* request.
64	Cancel all jobs	Cancels all incomplete *job* requests for all clients on the client node.

Table 2 – Error conditions

Type	Name	Description
		Database errors
2	Unknown <u>environment</u>	The <u>environment</u> in a request is not known to the server.
		Protocol errors
38	Parameter syntax	The server cannot parse a device parameter or a process parameter.
39	Unknown parameter	A process parameter is unknown, or the keyword of a device parameter is unknown, or a device parameter's position is unacceptable to the server.
40	Parameter value	The value of a device parameter or a process parameter is unacceptable to the server.
		Process initiation errors
61	Unknown routine	The routine named in a *job* request is unknown to the server.
62	Unknown label	The label given in a *job* request is not present in the routine on the server.
63	Unknown line	The line specified by *label* + *offset* in a *job* request is not present in the routine on the server.
64	Job timed out	The process requested by a *job* request was not initiated during the time specified.
65	Overlapping job requests	The server received a *job* request before a preceding *job* request from the same client was completed.
66	Unknown job request number	The job request number in a *job status* request or a *cancel job* request is not that of the last job request processed by the server for the client program.
67	Job cancel	A *cancel job* request attempted to cancel a *job* request after process initiation completed.
68	Job error	An error of process initiation occurred.

5.3.x Routine reference

[Insert this clause after 5.3.3 Global reference and renumber the following clauses.]

4.1 describes the name of a MUMPS routine, its environment, and the actual arguments that may be passed to it.

A routine reference shall be a long string <LS>. Its fields shall be strings containing the environment, routine name, and actual arguments. The fields of the *routine reference* and their sequence shall be:

a *Environment:* <LS> denotes the server's environment from which the routine shall be fetched.

NOTE – The job request may specify a different environment in which the routine is to execute.

b *Name:* <SS> A caret shall precede the name of the routine, as shown in X11.1 clause 8.1.6.1.

c *Argument count:* <SI> the number of argument fields following.

d *Arguments:* 0 or more (the number given in the argument count) <LS>. Each is the value of an actual argument passed to the routine.

An *argument count* of 0 shall denote an empty list of arguments. The *argument count* and *arguments* fields shall be omitted when no list of arguments is passed to the routine. The language syntax of X11.1, clauses 8.1.6.1 and 8.1.6.2, distinguishes an empty list from an omitted list of arguments.

<LS>		<SS>		<SI>	<LS>		<LS>		...
<LI>	Environ	<SI>	Name	Arg count	<LI>	Arg 1	<LI>	Arg 2	

Figure 1 – Form of routine reference

5.3.y Process parameters

[Insert this clause where appropriate in clause 5.3.]

The client may set parameters of a process initiated on another node. Process parameters are defined by the server's implementation; they appear in a *job* request as:

Process parameters: <LS> 0 or more individual process parameters.

<LI>	<LS>				
	<SS>		<SS>		...
	<SI>	Parameter	<SI>	Parameter	

Figure 2 – Form of process parameters

NOTE – When more than one process parameter is erroneous, the server may report any one of the errors.

5.4 Requests and responses

5.4.26 Job

(Operation type 61) initiates an M process on the server.

The fields of the *job* request and their sequence shall be:

a *Job environment:* <LS> the server's environment in which the process shall execute.

b *Client identifier:* <SS> $Job (see ANSI/MDC X11.1) of the client process making the request, an ASCII string of decimal digits.

c *Label:* <SS> the label of a line in the M routine named in this request. If *label* is the empty string it refers to the first line of the routine.

d *Offset:* <LI> the number, 0 or more, of lines following *label* at which execution shall begin.

e *Routine reference:* <LS> the M routine that the initiated process shall execute, including its name, the environment from which it shall be fetched, and its actual arguments. See 5.3.x.

f *Process parameters:* <LS> as defined in 5.3.y.

g *Time out:* <SS> The empty string indicates no time out; the server shall try indefinitely to initiate the process. Otherwise, the possibly non-integer decimal number of seconds that the server shall try before invoking the *job timed out* error condition.

NOTE – a *cancel job* request may cancel the server's attempt before process initiation succeeds.

<LS>		<SS>		<SS>		<LI>
<LI>	Job env	<SI>	Client id	<SI>	Label	Offset

<LS>		<LS>		<SS>	
<LI>	Routine ref	<LI>	Process parameters	<SI>	Time out

Figure 3 – Job request

A *job* request shall be erroneous while another *job* request from the same client is incomplete. A *job* request shall be complete when either of the following has occurred:

– The requested process has been initiated successfully, even if the server has not reported it to the client. That is, the *job* response may indicate *pending* and process initiation may complete later.

– The server responds with any error condition related to the *job* request.

The server shall respond as soon as possible, that is, it shall not wait for initiation of the requested process to complete. The response header shall indicate an error in the *job* request itself, or it shall contain error type 0, indicating success, and the error modifier shall denote the completion of process initiation: 0 shall indicate that process initiation has completed successfully, and 1 shall indicate that process initiation is pending—the agent should send a *job status* request to learn about completion.

The fields of the *job* response and their sequence shall be:

a *Job request number:* <LI> a number chosen by the server shall uniquely identify 1 job request from the time this response is sent until the job request is complete. *Job status* and *cancel job* requests shall use it to distinguish job requests. If process initiation has completed at the time of the *job* response, the value of *job request number* is unspecified.

b *Process identifier:* <SS> $Job (see ANSI/MDC X11.1) of the initiated process, an ASCII string of decimal digits. If the server's operating system does not provide this information, this field shall equal the empty string. While process initiation is pending, the value of *process identifier* is unspecified.

<LI> Job request #	<SI>	<SS> Process identifier

Figure 4 – Job response

The server shall retain the job request number and status of 1 *job* request per client. (It may do so for more than 1 *job* request at the option of the implementer.) The server shall retain the job request number and status of each *job* request until the server has reported the disposition of that *job* request once, specifically until the server has taken any one of the following actions:

– issued a *job* response indicating any error.

– issued a *job status* response indicating any error.

– issued a *job* response indicating that process initiation has completed successfully.

– issued a *job status* response indicating that process initiation has completed successfully.

– issued a *cancel job* response for that job request number.

– issued a *cancel all jobs* response.

NOTE – an *overlapping job requests* error therefore causes the status of preceding *job* requests to be lost. Acceptance of a *job* request implies that it does not overlap preceding *job* requests (unless the implementation supports multiple pending *job* requests). Therefore, when the server receives non-overlapping *job* requests, it need retain the job request number and status of only the most recent *job* request per client.

Subsequent *job status* requests may evoke an *unknown job request number* error.

5.4.27 Job status

(Operation type 62) requests the status of the last *job* request from the client program.

The one field of the *job status* request shall be:

Job request number: <LI> the job request number from the response to the *job* request whose status is being queried.

<LI> Job request #

Figure 5 – Job status request

The response header shall indicate an error of process initiation, or it shall contain error type 0, indicating success, and the error modifier shall denote the completion of process initiation: 0 indicates that process initiation has completed successfully, and 1 indicates that process initiation is pending—the agent should send another *job status* request to learn about completion.

The fields of the *job status* response and their sequence shall be:

Job request number: <LI> the job request number from the *job status* request.

b *Process identifier:* <SS> $Job (see ANSI/MDC X11.1) of the initiated process, an ASCII string of decimal digits. If the server's operating system does not provide this information, this field shall equal the empty string. While process initiation is pending, the value of *process identifier* is unspecified.

<LI>	<SS>	
Job request #	<SI>	Process identifier

Figure 6 – Job status response

5.4.28 Cancel job

(Operation type 63) cancels the preceding job request from a client program if the process has not yet been initiated.

The one field of the *cancel job* request shall be:

Job request number: <LI> the job request number from the response to the *job* request that is being canceled.

<LI>
Job request #

Figure 7 – Cancel job request

The *cancel job* response shall consist of a header only. The server shall indicate a *job cancel* error when process initiation has completed or has irrevocably begun. That is, a successful *cancel job* transaction shall assure that agent that the requested process did not and will not execute.

5.4.29 Cancel all jobs

(Operation type 64) The *cancel all jobs* request shall consist of a header only, indicating that all incomplete job requests from all clients on the client node shall be canceled.

NOTE – This operation is included for the convenience of implementers — there is no requirement that the request be sent in any specific condition.

The response shall consist of a header only. Unlike the *cancel job* response, the *cancel all jobs* response need not assure that all requested processes did not and will not execute.