
Information technology — Communication protocol — Open MUMPS Interconnect

*Technologies de l'information — Protocole de communication —
Interconnexion de systèmes ouverts pour le langage MUMPS*



Contents

1 Scope..... 1

1.1 Scope..... 1

1.2 Purpose..... 1

1.3 Application..... 1

2 Normative references 1

3 Definitions 1

4 General description..... 2

4.1 OMI and MUMPS..... 2

4.2 OMI and the OSI network model..... 3

4.3 Client-server protocol..... 3

4.3.1 Sessions 3

4.3.2 The role of the agent..... 3

4.3.3 Transactions 4

4.3.4 Complex locks 4

4.4 National character sets 4

4.5 Security 4

4.5.1 Privacy 4

4.5.2 Data Integrity..... 5

4.5.3 Authentication 5

4.5.4 Identification..... 5

4.5.5 Authorization 5

4.6 Replication 5

4.7 Environments 6

4.8 OMI version negotiation 6

4.9 Exception handling..... 6

4.9.1 Database errors 6

4.9.2 OMI protocol exceptions 6

4.9.3 Session establishment errors..... 6

4.9.4 Virtual circuit failure..... 6

4.10 Implementation limits and portability..... 6

4.11 Extensions to the standard..... 7

5 Message form and content 7

5.1 Characters..... 7

5.2 Fields..... 7

5.3 Message form 7

5.3.1 Request header..... 8

5.3.2 Response header..... 8

5.3.3 Global reference..... 12

5.4 Requests and responses 12

5.4.1 Connect..... 12

5.4.2 Status 14

5.4.3 Disconnect 14

5.4.4 Set..... 14

5.4.5 Set piece 14

5.4.6 Set extract..... 15

5.4.7 Kill..... 15

5.4.8 Get 15

5.4.9 Define..... 16

5.4.10 Order 16

5.4.11 Reverse order 17

5.4.12 Query..... 17

5.4.13 Lock..... 17

5.4.14 Unlock 18

5.4.15 Unlock client..... 18

5.4.16 Unlock all..... 18

Normative Annex A 19

A.1 Implementations..... 19

A.2 Programs..... 19

Informative Annex B 21

B.1 Implementation..... 21

B.2 Network management..... 21

B.3 Authorization 21

B.4 Protocol stack..... 21

B.5 Sessions..... 22

B.6 Complex operations 22

B.6.1 Lock 22

B.6.2 Merge..... 22

B.7 Registering implementations..... 22

B.8 Compliance verification..... 23

Informative Annex C..... 24

C.1 Requests and responses 24

C.1.1 Increment..... 24

C.1.2 Reverse query..... 24

C.2 Other database support 25

C.3 Beyond database functions..... 25

C.4 Performance enhancements..... 25

C.5 Other protocols 25

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 15851 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

Annex A forms an integral part of this International Standard. Annexes B and C are for information only.

Information technology — Communication protocol — Open MUMPS Interconnect

1 Scope

1.1 Scope

Open MUMPS Interconnect defines a method for network access to MUMPS databases. The protocol provides all basic operations on the sparse tree-structured MUMPS database.

1.2 Purpose

Because the MUMPS language standard (see 2) defines the operations on its database facilities, a MUMPS-specific protocol is required to extend these facilities into an open systems environment.

1.3 Application

Developed primarily for connection of different implementations of the MUMPS language, the protocol may also be used by other languages to gain access to a MUMPS database or to provide a MUMPS database service.

OMI may also be used for inter-task operations on a single computer, for example between two different products.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this International Standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this International Standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ANSI X3.4-1990, *American standard code for information interchange*.

ANSI/MDC X11.1-1995, *American national standard for information systems — programming languages — MUMPS*.

ISO/IEC 8859-1:1998, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1*.

3 Definitions

For the purpose of this International Standard, the following definitions apply.

3.1 MUMPS: name of the programming language defined by ANSI/MDC X11.1.

3.2 OMI: an acronym for Open MUMPS Interconnect.

- 3.3 OSI:** the Open Systems Interconnect model of computer communication protocols.
- 3.4 message:** a string of 8-bit characters, allowing all character codes from 0 to 255.
- 3.5 circuit:** an error-free, sequence-preserving path for messages. The virtual circuit provided by an OSI level 4 service.
- 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 response messages.
- 3.8 transaction:** one request message and the related response message. An OMI transaction is not the same as a transaction processing transaction.
- 3.9 agent:** a process that manages transactions on behalf of one or more clients.
- 3.10 session:** a connection between exactly one agent and exactly one server. When establishing a session, the two processes authenticate each other and negotiate the parameters of transactions to follow.
- 3.11 user:** the human being for whom a client process runs.
- 3.12 user ID:** a number identifying a user for authorization purposes.
- 3.13 group ID:** a number identifying a group of users for authorization purposes.
- 3.14 OMI node:** Usually a node is a computer, but two or more independent implementations might share a computer without sharing their databases, and thus be separate nodes. Conversely, an implementation might occupy several computers connected by a non-OMI method, and thus be one OMI node. Specifically, an OMI node is a set of one or more environments that are accessible without use of the OMI network. That is, a program can refer to an environment in its own node without using OMI.

4 General description

4.1 OMI and MUMPS

ANSI/MDC X11.1 defines the components of the MUMPS database and operations on them. OMI makes some of these operations, listed here, available over a network. The following references are to the formal definitions in ANSI/MDC X11.1; the additional text is for information only:

– environment: 7.1.2.4 identifies a specific set of all possible names — a name space. A particular name may appear only once in an environment. Typical implementations place an environment in a directory or a user class, and — with a network — on a node as well.

– gvn: 7.1.2.4 Global variables are persistent data records organized into trees. gvn is the name of a global variable, for example:

^INV(5321,"Denver","qu")

A global variable may optionally have a value, and it may optionally have descendants. A tree of global variables is informally called a "global."

– naked reference: 7.1.2.4 a shorthand form of gvn referring to the last-used gvn.

– nref: 8.2.12 The objects of database lock operations are organized into trees. Their names are similar to gvns, but they relate to gvns only by an application's convention. nrefs have no values.

– \$Data: 7.1.5.3 Function of a gvn indicates whether the variable it names has a value and whether it has descendants.

– \$Get: 7.1.5.7 Function of a gvn returns the value of the variable it names, or the empty string if the variable is not defined.

– \$Job: 7.1.4.10 An unsigned decimal integer uniquely identifies a process on a particular computer.

- *Kill*: 8.2.11 deletes a global variable and its descendants.
- *Lock+*: 8.2.12 claims exclusive use of an nref or a list of nrefs. If the claim succeeds, other concurrent processes' claims on those nrefs will not succeed.
- *Lock-*: 8.2.12 releases a claim on an nref or a list of nrefs.
- *Merge*: 8.2.13 assigns the value of one variable to another and then assigns the values of all the first variable's descendants to corresponding descendants of the second variable.
- *\$Order*: 7.1.5.11 Function of a gvn returns the final subscript of the next gvn in a defined tree-walking order.
- *\$Query*: 7.1.5.15 Function of a gvn returns the next gvn in a defined tree-walking order.
- *Set*: 8.2.18 creates a global variable and assigns it a value.
- *Set \$Extract*: 8.2.18 creates a global variable and modifies its value by assigning a range of its character positions.
- *Set \$Piece*: 8.2.18 creates a global variable and modifies its value by assigning one or more of its pieces.

B.6 (of this International Standard, not X11.1) suggests how implementations may combine these basic operations to achieve more complex MUMPS operations.

4.2 OMI and the OSI network model

OMI provides the services described by levels 5 and 6 of the Open Systems Interconnect model. Level 5, the session layer, involves creating and terminating communication sessions between cooperating systems, while Level 6, the presentation layer, deals with data formats. These OMI services make the network transparent to applications (OSI level 7) for their remote database operations.

In turn, OMI relies on the virtual circuits of an existing OSI level 4 service to provide reliable sequential transmission of messages.

Successful communication with OMI depends on hardware and software to establish compatible protocols at level 4 and below, which are beyond the scope of this International Standard.

4.3 Client-server protocol

OMI is a connection-oriented protocol based on the client-server model. An OMI node may offer client functions, server functions, or both, as its applications require and its implementer chooses.

4.3.1 Sessions

OMI uses one circuit to establish a session between exactly two OMI nodes. However, implementers may supply servers that support many sessions, to give several client nodes access to the server node's database. Implementers may also permit more than one session on a client node, so that different client processes may refer to data on several server nodes. However, a particular client shall connect to a particular server through only one session at a time, because duplicate paths between a client and a server could cause transactions to be processed out of sequence.

Initiation and termination of sessions shall be invisible to application programs. The implementation shall establish and terminate sessions as necessary to perform OMI operations.

A session is not symmetric. Requests from one client node are satisfied at the other server node. Another session is needed to handle requests in the opposite direction.

4.3.2 The role of the agent

OMI allows, but does not require, an agent process that multiplexes requests from any number of clients on one node in a single session. Thus several clients may share one connection to a server, and all OMI servers shall accept multiplexed requests.

NOTE – Although agents are an important conceptual part of this International Standard, their existence does not affect the form, content, or sequence of messages. A simple implementation could provide one client per session, and you may read "client" wherever "agent" appears.

This International Standard describes the agent as a separate process, but an implementation may provide its functions by any method.

The agent is synchronous. A session supports only one transaction at a time. The agent shall send one request and then no more until the server's response arrives or the circuit fails.

4.3.3 Transactions

An agent shall originate each transaction with exactly one request message. The server shall attempt the requested operation and shall reply with exactly one 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.

Only the agent shall initiate a transaction. If the server has something to tell the agent, for example "shutting down," the server shall indicate in its next response message that the agent should request the server's status.

Transactions are isolated. The server need not retain information about completed transactions.

Nonetheless, both server and agent shall retain the parameters of the session including, for example, agent's and server's names, and negotiated maximum lengths of gvns, values, and messages.

All OMI transactions shall have the same priority or the same class. That is, there are no out-of-band messages. Protocol management transactions such as capability negotiation, status updates, or startup and shutdown shall be handled in the same manner as database transactions.

4.3.4 Complex locks

A client may lock a list of nrefs that lie on different OMI nodes. The lock operation is atomic — it shall succeed only if all nrefs in the list are successfully claimed. This property, combined with the isolated transactions between the agent and its servers, requires the agent to assemble a complex lock from OMI lock requests to the respective servers.

The agent shall claim all nrefs of a complex lock. If all the claims succeed, then the entire lock succeeds. Otherwise, the agent shall release its successful claims and repeat the procedure until the lock succeeds or times out.

NOTE – To facilitate this procedure, the protocol defines a request to unlock all nrefs held by a client on one server.

4.4 National character sets

ANSI/MDC X11.1, Annex A, requires ASCII symbols for codes 0 – 127. Many English language terminals and all other language terminals require different encodings, some of which use codes 128 - 255 as well. Most information systems have addressed the question of printing a symbol on a printer that uses a different code from the keyboard that originated the symbol. Networked systems should also address the translation of codes between agents and servers that use different character sets.

All OMI messages shall use the ISO 8859-1 Latin alphabet No. 1, which equals ASCII for codes 0 – 127, and which specifies common European symbols for codes 128 – 255.

Both agents and servers shall send the standard character set on output and shall accept it as input, except for the following special case:

OMI provides an exemption from this requirement for an agent and a server that share a single non-standard character set internally. When establishing their session they may select untranslated messages, thus avoiding the burden of two reciprocal translations.

4.5 Security

Security of computer networks has many facets. This International Standard addresses some of them and leaves some to other layers of the protocol stack. The general principle is to afford the networked system as much security as typical single-computer commercial systems furnish.

4.5.1 Privacy

OMI offers no privacy protection. System managers who wish to thwart eavesdropping should specify lower-level protocols that provide encryption.

4.5.2 Data Integrity

OMI depends on the virtual circuit of its underlying level 4 service to provide integrity of the data in OMI messages. An OMI transaction either completes or fails entirely, therefore a broken connection cannot cause a partial database operation.

4.5.3 Authentication

When establishing a session, the server and the agent shall exchange passwords to verify each other's authenticity.

4.5.4 Identification

A server shall be identified by one or more names that are available to agents. Likewise, an agent shall be identified by one or more names that are available to servers. The server and agent shall exchange their names when establishing a session.

Each message 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.

4.5.5 Authorization

OMI supports authorization, the determination that the user has authority for the requested operation. The method encompasses most MUMPS implementations of authorization (see A.3).

The agent shall include a user ID and a group ID in each request. Both identifiers shall pertain to the server, that is, the agent shall translate its implementation's identifiers to the server's identifiers.

NOTE – A different translation may be necessary for each server.

The server shall test the request's user ID and group ID for authorization to perform the requested operation on the object of the request. If either the user ID or the group ID passes the test, then the request is authorized.

The server shall not perform an unauthorized request, but shall return an authorization exception response.

4.6 Replication

Implementers of OMI may offer replication of changes to selected global variables. That is, *set* and *kill* operations are also applied to global variables of the same name in one or more other environments, some of which may reside on other OMI nodes.

Replication is often applied symmetrically to keep the databases identical, whichever one a client changes explicitly. For example, ^XX on computer A is replicated to ^XX on computer B and vice versa, so a change to either one evokes an OMI request to change the other. Then the protocol must avoid an endless series of transactions caused by mutual replication.

Requests (*set*, *set extract*, *set piece*, and *kill*) that alter variables shall include a replication flag to indicate whether the request should be replicated.

NOTE – Implementations of OMI are not required to perform replication. They are required to provide the replication flag in order to cooperate with implementations that do perform replication.

All agents shall originate *set*, *set extract*, *set piece*, and *kill* requests with replication enabled. Servers shall attempt the requested operation whether or not replication is enabled, subject to other considerations like authorization, and shall reply as described in 4.3.3.

An implementation that supports replication shall take further action if and only if it successfully performs a *set*, *set extract*, *set piece*, or *kill* operation with replication enabled. (The operation may have originated from an OMI request or from a local *set*, *set extract*, *set piece*, or *kill* by a process on the server's own node.) The agent on that node shall:

- establish sessions with the replication destination nodes, if not already established.
- send the same request with replication disabled to each of the destinations, in effect saying, "This is a replication message. Do not replicate it further."