Version 3

Scalable Message Oriented Middleware for Distributed Computing

MomSys

Reference Manual

Printed in United States of America

X♦*IPC* VERSION 3.3.0

MOMSYS

REFERENCE MANUAL

Table of Contents

1. X•IPC VERSION 3.3.0: MomSys API

1.1 Overview

The table below lists the utility commands and API verbs that are employed for building MomSys applications. Included are definitions of all X•IPC and MomSys verbs necessary for using the MomSys subsystem. The Xipc*Xxx*() verbs are detailed in the X•IPC Version 3.3.0 User Guide and Reference Manual. Please refer to those documents for the relevant information.

`.cfg File`	MomSys configuration parameters
`XipcConnect()`	Connect to an X•IPC instance
`XipcDisconnect()`	Disconnect from an X•IPC instance
`XipcLogin()`	Log into an X•IPC instance.
`XipcLogout()`	Log out of an X•IPC instance.
`momview`	MomSys monitor and debugger utility program
`mrclean`	MomSys Message Repository cleanup utility
`MomAbortAsync()`	Abort a pending event or asynchronous operation
`MomAccess()`	Access AQid handle of an application queue or group.
`MomAttrSet()`	Set attributes in an application queue attribute block.
`MomCreate()`	Create an application queue within local instance.
`MomDelete()`	Delete an empty application queue from within local instance.
`MomDestroy()`	Destroy an application queue from within local instance.
`MomEvent()`	Wait for a MomSys event.
`MomInfoAppQueue()`	Get application queue information within local instance.
`MomInfoAppQueueWList()`	Get application-queue wait-list information
`MomInfoLink()`	Get network-link information.
`MomInfoMessage()`	Get latest message information from within local instance.
`MomInfoSys()`	Get MomSys information from local instance.
`MomInfoUser()`	Get MomSys user information of user within local instance.
`MomInfoUserAList()`	Get MomSys user asynchronous operation list information
`MomReceive()`	Receive a message from an application queue within local instance.
`MomSend()`	Send a message to one or more application queues.
`MomStatus()`	Get latest message status.
`MomStatusWait()`	Wait for message to achieve specified status.

1.2 Terminology

Throughout the following Reference Manual pages certain terms are employed interchangeably to describe various aspects of the MomSys subsystem and its API. Some are synonyms, others are short-hand abbreviations. The following is a list of these terms:

- ***Program, Application, Process, Thread*** - are terms used to describe an executing body of code that is performing MomSys functions. Except where denoted, these terms are interchangeable from the perspective of the following *X•IPC* manual pages.
- ***X•IPC Catalog Namespace, X•IPC Catalog, X•IPC Namespace, Namespace*** - are terms that are interchangeably used to describe a single *X•IPC* Namespace.
- ***X•IPC Application Queue, Application Queue, App-Queue*** - are terms that are used interchangeably to describe a queue of application messages.
- ***Message Tracking Level, Message Track Level, Tracking Level*** - are terms that are used interchangeably to describe the degree to which a message is tracked as it moves between sender and receiver programs.
- ***Message Identification Integer Handle, Message-Id, Message Handle, MsgId*** - are terms used interchangeably to describe the opaque value that is assigned to a MomSys message when it is sent, and for subsequent message tracking.
- ***AppQueue Id, AppQid, AQid*** - are terms used interchangeably to describe the integer value that is associated with an app-queue when it is created by MomCreate(), or referenced by MomAccess().

Note as well that function arguments that are to be set with return values (“output arguments”) during the execution of the function are generally named **Ret*Xxx*** to indicate that they are return variables..

2. MOMSYS CONFIGURATION

2.1 MomSys Parameters

2.1.1 X♦IPC Instance Configuration - MOMSYS PARAMETERS

NAME

X♦IPC Instance Configuration - MomSys parameter definitions for .cfg files

SYNTAX

[XIPC]
… General XIPC section parameters related to MomSys, defined below …

[MOMSYS]
… General MomSys parameters, defined below …
… Message Repository parameters, defined below …
… Communication Manager parameters, defined below …

[MOMSYS.*protocol*]
… Protocol-specific MomSys parameters, defined below …

PARAMETERS

This section defines the instance configuration parameters that relate to the MomSys subsystem. The parameters will be presented in the following categories, each addressing a different aspect of MomSys:

- General X♦IPC parameters
- General MomSys parameters
- Message Repository parameters
- Communication Manager parameters
- Protocol-specific parameters

Note that if multiple MomSys instances are to be started on a single platform, each instance must have its configuration file in a separate directory because MomSys generates files in that directory that will conflict with one another.

2.1.2 GENERAL X♦IPC PARAMETERS

The table below lists the general instance configuration parameters, i.e., parameters that go within the [XIPC] section of the instance configuration file in support of the MomSys programming model. Each parameter is presented with its name, description and default value. The order that parameters appear within the [XIPC] section of the configuration is not significant.

Parameter Name	Description	Default Value
NAMESPACE	The name of the *X♦IPC* namespace to affiliate the instance with.	There is no default

2.1.3 GENERAL MOMSYS PARAMETERS

The table below lists the general MomSys configuration parameters. Each parameter is presented with its name, description and default value. The order that parameters appear within the [MOMSYS] section of the configuration is not significant. The default values shown do *not* represent limits for the values that any particular user may require.

Parameter Name	Description	Default Value
MAX_USERS	The maximum number of concurrent MomSys users (real users and pending asynchronous operations) that can be supported by the subsystem.	32
MAX_DISK_AQ	The maximum number of disk-based app-queues.	16
MAX_REMOTE_AQ	The maximum number of remote app-queues to be accessed at any one time.	31
MAX_MSG_LENGTH	The maximum message size. Note that when two instances are communicating, MAX_MSG_LENGTH must be the same for both instances; otherwise, results will be unpredictable.	1024

2.1.4 MESSAGE REPOSITORY PARAMETERS

The table below lists the message repository configuration parameters. They too are part of the [MOMSYS] section within an instance configuration file. Each parameter is presented with its name, description and default value. The order that parameters appear within the [MOMSYS] section of the configuration is not significant.

Parameter Name	Description	Default Value
TIMEOUT_EXPIRE_MRO	The time that incomplete outbound messages are allowed to remain incomplete within the MRO. Time is specified as a string such as “12h” or “30m”, etc., where the format is “nUNITS” where UNITS is: s, m, h, d or w; or infinite indicating that incomplete messages are never made eligible for purging.	infinite

Parameter Name	Description	Default Value
TIMEOUT_EXPIRE_MRI	The time that inbound messages are allowed to remain undelivered within the MRI. Time is specified as a string such as “12h” or “30m”, etc., where the format is “nUNITS” where UNITS is: s, m, h, d or w; or `infinite` indicating that undelivered messages are never made eligible for purging.	`infinite`
TIMEOUT_RETIRE_MRO	The time that “completed” outbound messages are kept within the MRO after “completing”. Time is specified as a string such as “12h” or “30m”, etc., where the format is “nUNITS” where UNITS is: s, m, h, d or w; or `immediate` indicating that completed messages are immediately made eligible for purging..	`immediate`
TIMEOUT_RETIRE_MRI	The time that delivered inbound messages are kept within the MRI after delivery. Time is specified as a string such as “12h” or “30m”, etc., where the format is “nUNITS” where UNITS is: s, m, h, d or w; or `immediate` indicating that delivered messages are immediately made eligible for cleaning.	`60m`
SCHED_MR_CLEAN	A “schedule-string” defining when MomSys cleans MRI and MRO of expired or retired messages; or “`none`” indicating no automatic cleaning. (Refer to the MomSys User Guide, section 8.1.5, for definitions of “schedule-string” syntax.)	`0,30 * * * *` (Clean occurs every 30 minutes)
MODE_MR_CLEAN	Some combination of the following three keywords: • `STARTUP` – indicating that MR clean is to occur at instance start. • `SCHEDULED` – indicating that MR clean is to occur based on value of `SCHED_MR_CLEAN` • `CONTINUOUS` – indicating that a partial, but incomplete, clean should occur on-the-fly.	`STARTUP SCHEDULED CONTINUOUS` (all three expressed as a single parameter, separated by spaces)
SLOT_SIZE_MRI	Should be set to the 90%-tile message size (in bytes) of messages to use MomSys. If you change the `SLOT_SIZE` in the `.cfg` file, and then attempt to restart an instance `xipcstart` may fail. You must start a fresh instance if you plan to change the `SLOT_SIZE.`	`256`
MAX_FILES_MRI	Maximum number of disk files to be used by MRI	`512`
FILE_SIZE_MRI	Initial size of MRI files when created (in KBs).	`1024` `(= 1 MB)`
MAX_MAPPED_MEMORY_MRI	Maximum bytes of MRI data mapped into system at any one time (in KBs).	`32768` `(= 32 MB)`

Parameter Name	Description	Default Value
SLOT_SIZE_MRO	Should be set to the 90%-tile message size (in bytes) of messages to use MomSys. If you change the SLOT_SIZE in the .cfg file, and then attempt to restart an instance xipcstart may fail. You must start a fresh instance if you plan to change the SLOT_SIZE.	256
MAX_FILES_MRO	Maximum number of disk files to be used by MRO	512
FILE_SIZE_MRO	Initial size of MRO files when created (in KBs).	1024 (= 1 MB)
MAX_MAPPED_MEMORY_MRO	Maximum bytes of MRO data mapped into system at any one time (in KBs).	32768 (= 32 MB)
DATABASE_MRI	Path of inbound message repository. Note that it is *not* possible to have MRI databases from two instances sharing a single directory; naming conflicts will occur. In such a case, set the two instances' DATABASE_MRI parameters to point to separate file-system directories.	Path of instance .cfg file.
DATABASE_MRO	Path of outbound message repository. Note that it is *not* possible to have MRO databases from two instances sharing a single directory; naming conflicts will occur. In such a case, set the two instances' DATABASE_MRO parameters to point to separate file-system directories.	Path of instance .cfg file.
JOURNAL_EXPIRED_MSGS_MRI	The fully qualified filename in which expired MRI messages are to be journaled. (See note below.)	No default
JOURNAL_RETIRED_MSGS_MRI	The fully qualified filename in which retired MRI messages are to be journaled. It may be the same as the above filename. (See note below.)	No default
JOURNAL_EXPIRED_MSGS_MRO	The fully qualified filename in which expired MRO messages are to be journaled. (See note below.)	No default
JOURNAL_RETIRED_MSGS_MRO	The fully qualified filename in which retired MRO messages are to be journaled. It may be the same as the above filename. (See note below.)	No default

NOTE: If any journal parameter is not specified, no journaling occurs for that message class. The two MRI journal filenames may both refer to the same file, as may the two MRO filenames, but no one file may be specified as the journal file for both MRI and MRO messages. For example, JOURNAL_EXPIRED_MSGS_MRI and JOURNAL_EXPIRED_MSGS_MRO must be distinct if they are both specified. There is no default journal file.

2.1.5 COMMUNICATION MANAGER PARAMETERS

The table below lists the general instance configuration parameters. They too are part of the [MOMSYS] section within an instance configuration file. Each parameter is presented with its name, description and default value. The order that parameters appear within the section of the configuration is not significant.

Parameter Name	Description	Default Value
MAX_INSTANCES_LINKS	Maximum number of remote instances that can be linked to this instance at any one time.	31

2.1.6 PROTOCOL SPECIFIC PARAMETERS

The table below lists the protocol-specific instance configuration parameters. Each parameter is presented with its name, description and default value.

A protocol-specific section having a section header of the form [MOMSYS.*protocol*] is required for each protocol to be supported by the instance being configured. The only value for *protocol* currently supported is tcpip. The order that parameters appear within the [MOMSYS.*protocol*] section is not significant.

The following is the list of protocol-specific MomSys configuration parameters.

Parameter Name	Description	Default Value
[MOMSYS.*protocol*]	MomSys protocol header where the only currently supported *protocol* value is tcpip.	-N/A-
LINK_RETRY_INTERVAL	The time between retries of trying to create a new link. Time is specified as a string such as "12h" or "30m", etc., where the format is "nUNITS" where UNITS is: s, m, h, d or w.	60s
LINK_PING_INTERVAL	The time between internal instance-ping messages sent to check if remote instances are still active. Time is specified as a string such as "12h" or "30m", etc., where the format is "nUNITS" where UNITS is: s, m, h, d or w.	120s
LINK_PING_TIMEOUT	The wait time for hearing a response to an instance-ping. If no response is received, the link to the remote instance is assumed down.	60s
MSG_RESPONSE_TIMEOUT	The wait time for receiving an internal "ack" on an application message forwarded to a remote instance.	60s
QUEUE_PROBE_TIMEOUT	The wait time for a response to an internal queue-probe message. If no response is received the queue probe fails.	10s
QUEUE_PROBE_RETRY_INTERVAL	The time between queue-probe attempts.	120s

3. MOMSYS UTILITY PROGRAMS

3.1 MomSys Utilities

3.1.1 momview - VIEW AN INSTANCE'S MOMSYS

NAME

momview - View an Instance's MomSys

SYNTAX

`momview [Interval] [InstName]`

PARAMETERS

Name	Description
Interval	The initial time interval between screen updates (in milliseconds). The default value is 1000.
InstName	The instance file name of the instance, or the *@registered_name* of the instance to be monitored The default is the value of the `XIPC` environment variable.

RETURNS

Value	Description

No return value.

DESCRIPTION

The `momview` utility is used for real-time monitoring of the activities occurring within an instance's MomSys. The specified *InstName* identifies the instance to be monitored. If *InstName* is not specified, the value of the `XIPC` environment variable is used to identify the instance to be monitored.

While `momview` is running, it is possible to control its operation by entering commands. The "command key" is operating system dependent and is defined in the respective Platform Notes. (E.g., on Windows NT this is the Ctrl-C key; on most UNIX platforms this is the terminal's "`interrupt`" key)

At the '`Command>`' prompt, one of the following commands can be entered (text in **bold** is are to be typed in as specified; items in *italics* represents values are to be provided by the user):

Main Window Commands:

i*n*	Set time interval to *n* milliseconds. Example: `i100`.
zu*n*	Zoom in on user *n*. Example: `zu5`.
zq*n*	Zoom in on app-queue *n*. Example: `zq1.2`.
zl*n*	Zoom in on instance-link *n*. Example: `zl.3`
zs	Zoom in on general subsystem status information. Example: `zs`
zmr	Zoom in on Message Repository status information. Example: `zmr`

zmri Zoom in on detailed Message Repository input (MRI) information. Example: `zmri`

zmro Zoom in on detailed Message Repository output (MRO) information. Example: `zmro`

u Un-zoom, close the zoom window.

lq View local app-queues (this is the startup mode)

rq View remote app-queues

pu*n* Pan view to user *n*. Example: `pu3`

pq*n* Pan view to queue *n*. Example: `pq1.4`

po Pan view to 'origin', (i.e., first app-queue, first user)

l Open the "Links" window to view the status of instance-links. Refer to the Links Window Commands below for the list of commands that can be performed from within the Links window.

b*n* Open the 'Browse" window to browse the contents messages on queue *n*, following the natural sequence. (Example: `b2.4` opens the browse window on app-queue 2.4.) Refer to the Browse Window Commands below for the list of commands that can be performed from within the Browse window.

q Quit. Exit the monitor

Links Window Commands:

p*n* Pan view to instance-link *n*. Example: `p5`

po Pan view to 'origin', (i.e., first instance-link)

q Quit. Close the Links window.

Browse Window Commands:

b*n* Browse the messages on queue *n*, following the time strand. Example: `b1.8`
(Note that `momview` can browse messages up to 10k bytes.)

q Quit. Browse window.

Navigating on an App-Queue:

Þ (*right arrow*) Move to the next message on the current sequence.

Ü *(left arrow)* Move to the previous message on the current sequence.

n Move to the nth message on the current sequence

+n Move forward n messages.

-n Move backward n messages.

f Move to the first message on the current sequence.

l Move to the last message on the current sequence.

Navigating within a message:

Ý *(up arrow)* Scroll the current message up one line.

ß *(down arrow)* Scrolls the current message down one line.

PAGE-UP Scroll the current message one page up.

PAGE-DOWN Scroll the current message one page down.

HOME Scroll the current message to its top.

END Scroll the current message to its bottom.

String pattern searching:

/IBM/	Search forward in the current message for the string "IBM".
//	Repeat the search.
/	Same.
\IBM	Search backwards in the current message for the string "IBM".
****	Repeat the search.
****	Same.
g/IBM/	Search forward for "IBM" through all messages to the end of the queue.
g//	Repeat the search.
g/	Same.
g\IBM	Search backwards for "IBM" through all messages to the start of the queue.
g	Repeat the search.
g	Same.

Hexadecimal pattern searching:

/4f37/x	Search forward for the hex pattern "4f37" within the current message.
g/4f37/x	Same search, but forward through all messages on the queue.
g//x	Same.
\4f37\x	Searches backwards for the hex pattern "4f37" within the current message.
g\4f37\x	Same search, but backwards through all messages on the queue.
g\\x	Same.

ERRORS

Display messages.

3.1.2 mrclean - MOMSYS MESSAGE REPOSITORY CLEANUP UTILITY

NAME

mrclean- MomSys Message Repository Cleanup Utility

SYNTAX

`mrclean [`*`InstName`*`]`

PARAMETERS

Name	Description
InstName	The instance file name of the instance, or the registered name of the instance to be MR cleaned. The default value is the value of the `XIPC` environment variable.

RETURNS

Value	Description

No return value.

DESCRIPTION

`mrclean` is a utility program that may be run by the MomSys user for manually removing "retired" and "expired" messages from the specified instance's message repository.

Retired and expired messages are either journaled or deleted depending on the instance's message repository configuration parameters. Instance configuration (i.e., the `SCHED_MR_CLEAN` parameter) also allows user-specification of times when *automatic* MR cleaning is to occur. Refer to the MomSys Configuration Parameter man-page in this Reference Manual for the definitions of these parameters.

The `mrclean` utility is provided as a method of *manually* performing the message repository clean operation. Note that `mrclean` can only be run after the targeted instance has been started, and is currently up.

ERRORS

Display messages.

4. MOMSYS API FUNCTIONS

4.1 MomSys Functions

4.1.1 MomAbortAsync() - ABORT AN EVENT OR ASYNCHRONOUS OPERATION

NAME

MomAbortAsync()- Abort an Event or Asynchronous Operation

SYNTAX

```
#include "xipc.h"

XINT
MomAbortAsync(
        XINT AUid)
```

PARAMETERS

Name	Description
AUid	An integer identifying a MomSys event or an asynchronous operation that is to be aborted. (See Description below for more details.)

RETURNS

Value	Description
RC >= 0	Abort successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomAbortAsync() aborts a pending MomSys event or asynchronous operation. The *AUid* argument passed to MomAbortAsync() was returned by *X•IPC* to the program that created the event (via a call to MomEvent()), or that initiated the asynchronous operation within the Asynchronous Control Block (ACB) data structure that was specified as part of that original call. In particular, the ACB's `AUid` field was set by *X•IPC* at the time of the original call. Refer to the "Asynchronous Operations" section of the *X•IPC* User Guide for details on this topic.

If the aborted asynchronous operation was originally issued by the same *X•IPC* user now calling MomAbortAsync(), the *BlockOpt* of the aborted operation is ignored and the Asynchronous Result Control Block (ACB) is not set.

If the aborted operation was issued by a different user, a return code of MOM_ER_ASYNCABORT is placed in the *RetCode* field of the aborted operation's ACB and the action specified in the *BlockOpt* of the operation is carried out, e.g., a callback routine is invoked.

ERRORS

Code	Description
MOM_ER_BADUID	Invalid *AUid* parameter.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_SYSERR	An internal error has occurred while processing the request.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

Syntax

momabortasync *AsyncUserId*

ARGUMENTS

AsyncUserId AUid of MomSys event or asynchronous MomSys operation to be aborted

EXAMPLES

```
xipc>  momreceive 1.0 first x callback(cb1,a)
       RetCode = -1097
       Operation continuing asynchronously

xipc>  acb a
       AUid = 17
       .
       .

xipc>  momabortasync 17
       Callback function CB1 executing ...
       RetCode = -1098
       Asynchronous operation aborted
```

4.1.2 MomAccess() - ACCESS HANDLE FOR APPLICATION QUEUE

NAME

MomAccess() - Access AQid handle for an application queue

SYNTAX

```
#include "xipc.h"

XINT
MomAccess(
        CHAR *Name)
```

PARAMETERS

Name	Description
Name	Application queue name for which a handle is desired. ***Name*** may be alternatively specified as `MOM_NOVERIFY(`***Name***`)`. See Description for details.

RETURNS

Value	Description
RC >= 0	Application Queue-Id (AQid) handle of accessed application queue
RC < 0	Error (see error codes below).

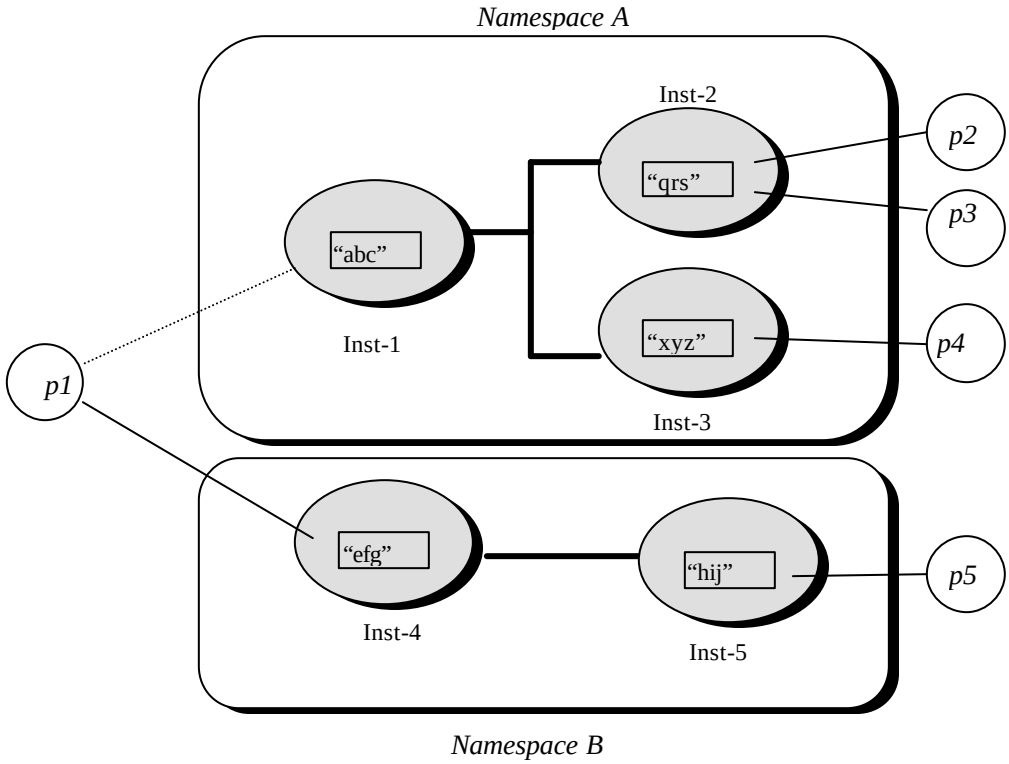
DESCRIPTION

MomAccess() is used for accessing an AQid handle of an app-queue. The returned AQid is then passed to other verbs for manipulating that app-queue. Examples include: MomSend() and MomReceive() for sending and receiving messages to/from app-queues; MomDelete() and MomDestroy() for administering app-queue contents; or MomInfoAppQueue() for monitoring app-queue statistics.

The AQid handle returned by MomAccess() is central to working with an app-queue. The degree that a process can manipulate and perform operations on an app-queue depends on the app-queue's location relative to it. Understanding what is meant by *"an app-queue's relative location to a user*" requires an appreciation of the MomSys programming model in general.

The following is a brief outline of the MomSys programming model and definitions that emerge. (Refer to the MomSys User Guide for a more detailed description of this topic.)

Consider the following diagram:



We see two X•IPC namespaces, A and B. There are three instances in namespace A. Three app-queues named "abc", "qrs" and "xyz" are situated in the three instances. Similarly, there are two instances, each containing an app-queue, in namespace B. The dashed and solid lines indicate user processes logged into instances. Dashed lines are logins that are not current. Processes p1 through p5 are logged into the various X•IPC instances.

The following definitions are pertinent:

Namespace

An X•IPC namespace is a collection of X•IPC instances. A namespace is managed by a catalog that is maintained on one or more platforms, for achieving redundancy. (Catalogs are not shown in the above diagram. Refer to the MomSys User Guide for a detailed discussion of X•IPC namespaces and catalog servers.)

Local Instance

A process must log into an X•IPC instance that is affiliated with a namespace before performing MomSys operations within that namespace. This instance is referred to as the process's local instance within that namespace. A process which is to work additionally within a second namespace can do so by logging into an instance within that namespace. Such a process is said to have two local instances, one per namespace. The instance to which a process is currently connected is the process's *Current Local Instance*. The corresponding namespace is the process's *Current Namespace.*

In the above diagram, process p1 is working within namespaces A and B and has, accordingly, logged into Inst-1 and Inst-4; these are p1's local instances. Similarly, p2, p3 and p4 are working within namespace A.

Processes p2 and p3 are using Inst-2 as their local instance, and p4 is using Inst-3. Process p5, on the other hand, is working within namespace B; its local instance is Inst-5.

Remote Instance

As just described, when a process works within an *X•IPC* namespace, the instance that it logs into within that namespace is that process's local instance within that namespace. All remaining instances within that namespace–those it is *not* logged into–are remote instances, relative to it. For example, Inst-2 and Inst-3 are p1's remote instances within namespace A. Inst-5 is a remote instance, relative to p1, within namespace B.

Local App-Queue

App-queues situated within a process's local instance are referred to as local app-queues relative to it. For example, app-queue "abc" in the above diagram is local relative to process p1. Similarly, app-queue "qrs" is local relative to processes p2 and p3.

A process may perform all forms of app-queue manipulation operations on local app-queues, including: MomSend(), MomReceive(), MomDelete(), MomDestroy() and MomInfoAppQueue().

Remote App-Queue

App-queues situated within a process's remote instances are referred to as remote app-queues relative to that process. For example, app-queue "qrs" is a remote app-queue relative to process p1. Similarly, app-queue "abc" is a remote app-queue relative to processes p2 and p3.

A process may only perform MomAccess(), MomSend() and MomInfoAppQueue() operations on remote app-queues.

Specifying the MomAccess() App-Queue *Name* Argument

With these definitions we can escribe the possible values of the *Name* argument passed to MomAccess():

Local app-queue names

A local app-queue is specified to MomAccess() using the string originally passed to the MomCreate() function when the app-queue was created within the calling process's local instance. Thus, the AQid handle of an app-queue that was created locally via `MomCreate("foo", …)` is accessed via `MomAccess("foo")`.

MomAccess() specifying a local app-queue will fail if the app-queue does not exist within the local instance at the time of the call.

Remote app-queue names

A remote app-queue is specified to MomAccess() using a string starting with the '@' character. A number of variations are possible:

@foo - indicates that the app-queue name "foo" is located somewhere within the calling process's current namespace. When such a name is passed to MomAccess(), *X•IPC* resolves the location of the app-queue via the *X•IPC* namespace catalog. This allows processes to send messages to app-queues without knowing where the app-queues are located within the namespace.

@SomeNode:SomeInstance:foo - indicates that app-queue "foo" is to be found within instance "SomeInstance," where that instance is to be found on node "SomeNode."

The normal behavior of MomAccess() is to fail when the specified remote app-queue entity name is not currently registered within the calling process's current *X•IPC* namespace.

Specifying *Name* as `MOM_NOVERIFY(Name)`, one can force MomAccess() to succeed even if the named app-queue is not found at the time of the call. In such a case, the AQid handle returned by

MomAccess() is a *virtual* app-queue handle. Messages sent using that handle are stored in the local message repository until an app-queue identified as *Name* is subsequently created and registered within the namespace. At that point, *X•IPC* forwards messages to that app-queue.

ERRORS

Code	Description
MOM_ER_BADAPPQUENAME	Invalid *Name* parameter.
MOM_ER_CAPACITY_CS	MomSys communicator server table full.
MOM_ER_CAPACITY_INSTANCE	MomSys instance link table full.
MOM_ER_CAPACITY_MR	MomSys message repository full.
MOM_ER_CAPACITY_REMOTEQUEUE	MomSys remote queue table full.
MOM_ER_CATERROR	Error in catalog server processing request.
MOM_ER_CATUNREACHABLE	All catalog servers unreachable.
MOM_ER_INTERNAL	Error occurred in one of MomSys servers (see log file).
MOM_ER_NONETWORK	Remote queue access cannot be performed since the instance is not configured for network connection.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTFOUND	Application queue with *Name* does not exist.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`momaccess` *Name* [`noverify`]

ARGUMENTS

Name	Name of app-queue to be accessed
[`noverify`]	Option specifying that `momaccess` should succeed even if named app-queue is not found. In that case, returned handle is *virtual* in nature. Messages sent to app-queue will be held until app-queue with *Name* is subsequently registered within XIPC namespace.

EXAMPLES

```
# Access an app-queue that is within local instance.

xipc>  momaccess abc
       AQid = 1.4
```

```
# Access an app-queue that is within some remote instance
# and that has been registered with the XIPC namespace.

xipc>  momaccess @xyz
       AQid = 2.1


# Access an app-queue that is within some remote instance
# and that has been registered with the XIPC namespace.

xipc>  momaccess @abc noverify
       AQid = 2.5



# Access an app-queue by specifying its fully qualified
# name, i.e., its node, instance and app-queue name.

xipc>  momaccess @somenode:someinstance:abc
       AQid = 1.7
```

4.1.3 MomAttrSet() - SET APP-QUEUE ATTRIBUTE BLOCK VALUES

NAME

MomAttrSet() - Set app-queue attribute block values

SYNTAX

```
#include "xipc.h"

XINT
MomAttrSet(
        MOM_ATTRBLOCK_APPQUEUE *AttrBlock,
        ... AttrSetOption)
```

PARAMETERS

Name	Description
AttrBlock	Pointer to variable of type `MOM_ATTRBLOCK_APPQUEUE`
AttrSetOption	One of the following options. Details are provided in the Description below:

`MOM_ATTR_SET_INITIALIZE`	Initialize attribute block to default values.
`MOM_ATTR_SET_DISK`	Create disk-based app-queue.
`MOM_ATTR_SET_PRIORITY`	Natural sequencing of messages is by message priority.
`MOM_ATTR_SET_TIME`	Natural sequencing of messages is by time of message arrival.
`MOM_ATTR_SET_AUTO_REGISTER`	Automatically register app-queue at catalog when app-queue is created.
`MOM_ATTR_SET_AUTO_REGISTER_UPDATE`	Automatically change the registration information of a registered app-queue.

RETURNS

Value	Description
RC >= 0	Success.
RC < 0	Error (see error codes below).

DESCRIPTION

MomAttrSet() is used to set the attributes within an app-queue attribute block. The attribute block is then typically passed to MomCreate() to create an app-queue with the specified app-queue attributes.

Each call to MomAttrSet() sets *one* attribute within the specified block. Setting multiple attributes requires a separate call per attribute. An attribute block must initially be set with `MOM_ATTR_SET_INITIALIZE`.

The following are descriptions of the possible app-queue attributes. Note that certain attributes are grouped within a vertical bracket, indicating that the grouped attributes are mutually *exclusive* (i.e., at most one attribute of the group may be specified). Within the marked groups:

- An asterisk appearing on *one* of the grouped attributes indicates that the marked attribute is the default value, if none is set.
- Where no attribute is marked, the group is optional and there is no default value.

MOM_ATTR_SET_INITIALIZE

This attribute sets the block to default values. It should be the first attribute set.

MOM_ATTR_SET_DISK

This attribute indicates that app-queues created using the given attribute block should move messages in a disk-based, non-volatile manner. This attribute supports asynchronous guaranteed message delivery of sent messages. This is the default attribute for app-queues that are created.

*** MOM_ATTR_SET_TIME**

This attribute indicates that app-queues created using the given attribute block should have a natural message sequencing that stores incoming messages in time order (i.e., oldest arriving message at the front).

MOM_ATTR_SET_PRIORITY

This attribute indicates that app-queues created using the given attribute block should have a natural message sequencing that stores incoming messages in priority order (i.e., highest priority message at the front).

MOM_ATTR_SET_AUTO_REGISTER

This attribute indicates that app-queues created using the given attribute block should be automatically registered when created, and deregistered when destroyed. Registration occurs in the creating process's current *X•IPC* catalog namespace. By default, app-queues are *not* automatically deregistered.

MOM_ATTR_SET_AUTO_REGISTER_UPDATE

This attribute updates an app-queue's registration data. It is typically used to relocate an app-queue from its current location to a new location and to have all programs that are currently sending messages to that app-queue have their messages subsequently be sent to the new location. (See the MomSys User Guide, Section 4.1.6 for further details.)

MOM_ATTRBLOCK_APPQUEUE - Block Definition:

MomAttrSet() sets the appropriate fields within the referenced attribute block based on the attribute setting that it is passed. Accordingly, MomAttrSet() should be viewed as a convenient tool for correctly populating attribute-block fields.

The following is the definition of the MOM_ATTRBLOCK_APPQUEUE data structure, as well as descriptions of possible values for each fields. (Note that there are more fields present than are supported by the current Version 3.0 product. These fields should be avoided as they are reserved for use in future *X•IPC* releases.)

```
typedef struct _MOM_ATTRBLOCK_APPQUEUE
{
    XINT AttrBlockType;             /* MOM_ATTR_TYPE_INITIALIZE */
    FLAG MemoryQueue;               /* FALSE */
    XINT MemoryQueueByteLimit;                  /* RESERVED */
    XINT MemoryQueueMsgsLimit;                  /* RESERVED */
    FLAG PriorityQueue;             /* TRUE if natural seq. is priority, or
                                     * FALSE if natural sequence is time */
    FLAG AutoRegister;              /* TRUE if app-queue is auto registered, or
                                     * FALSE if app-queue is not auto registered */
    XINT ExpireTime;                        /* RESERVED */
    XINT RetireTime;                        /* RESERVED */
    XINT KeyType;                               /* RESERVED */
    XINT KeyOffset;                             /* RESERVED */
    XINT KeySize;                               /* RESERVED */
    XINT T0;                                /* RESERVED */
    XINT T1;                                /* RESERVED */
    FLAG UserAttached;                          /* RESERVED */
    XINT CatFilterType;                     /* RESERVED */
    CHAR CatGroupName[XIPC_LEN_XIPCNAME+1];     /* RESERVED */
    CHAR CatFilter[MOM_LEN_FILTER+1];           /* RESERVED */
}
MOM_ATTRBLOCK_APPQUEUE;
```

A program may use the above definition for querying the attributes of a previously created app-queue. An example of this occurs when calling the MomInfoAppQueue() function for accessing information about an existing app-queue. The returned MOMINFOAPPQUEUE structure includes an embedded MOM_ATTRBLOCK_APPQUEUE block containing attribute data that was used when the referenced app-queue was created.

ERRORS

Code	Description
MOM_ER_BADATTRBLOCK	One of the following conditions exist: - Attribute block pointer is NULL. - The specified attribute block was not initialized by a call with MOM_ATTR_INITIALIZE.
MOM_ER_BADATTR	Invalid attribute value.
MOM_ER_NOTSUPPORTED	The specified attribute is not supported in the current version of MomSys.

INTERACTIVE COMMAND

No interactive command. Function supported via MomCreate() verb.

4.1.4 MomCreate() - CREATE AN APPLICATION QUEUE

NAME

MomCreate() - Create an application queue.

SYNTAX

```
#include "xipc.h"

XINT
MomCreate(
        CHAR *AppQueName,
        MOM_ATTRBLOCK_APPQUEUE *AttrBlock)
```

PARAMETERS

Name	Description

AppQueName Name of application queue, or `MOM_PRIVATE`

AttrBlock Pointer to variable of type `MOM_ATTRBLOCK_APPQUEUE`, or one of the following predefined default attribute block definitions:

`MOM_APPQUEUE_DISK` - a predefined attribute block for creating a disk app-queue with default settings (see Description).

`MOM_APPQUEUE_DISK_REGISTER` – a predefined attribute block used for creating an app-queue that has default attribute settings, with the exception that the created app-queue is automatically registered within the caller's current namespace. (Auto-registration is not the default) It is an error to register an app-queue name that is already registered. (See Description).

`MOM_APPQUEUE_DISK_REGISTER_UPDATE` - a predefined attribute block used for creating an app-queue that has default attribute settings, with the exception that the created app-queue is automatically registered within the caller's current namespace. (Auto-registration is *not* the default) In case the app-queue already exists, its attributes are updated with the attributes passed in the current call. It is typically used to relocate an app-queue from its current location to a new location and to have all programs that are currently sending messages to that app-queue have their messages now be sent to the new location. (See the Description below and the MomSys User Guide, Section 4.1.6, for further details.)

[**Note**: If specifying one of the above default values, do *not* specify pointers to them; rather, specify them as is. They are defined as pointers to predefined attribute blocks of type `MOM_ATTRBLOCK_APPQUEUE`.]

RETURNS

Value	Description
RC >= 0	Application Queue-Id (*AQid*) of created application queue.
RC < 0	Error (see error codes below).

DESCRIPTION

MomCreate() creates an application queue within the calling user's local instance. MomCreate() returns an *AQid* app-queue handle that may be passed as an argument to other MomSys API functions for locally manipulating the created app-queue.

If the application queue is named (i.e., not `MOM_PRIVATE`), its name must be unique within the calling process's local instance. Specifying `MOM_PRIVATE` as *AppQueName* directs *X•IPC* to create an app-queue that will be inaccessible by name from any other *X•IPC* user. Such an app-queue is useful for designing a scalable client/server communication mechanism:

> *In an architecture of this type, each client creates its own private app-queue (via `MOM_PRIVATE`) for receiving response messages from servers. Client request messages are then sent to servers via MomSend(), including the specification of the client's private app-queue as the reply app-queue. Server programs receiving such messages from a population of clients can respond directly to the sending client's private response app-queue without needing to identify the app-queue by name.*
>
> *Such an architecture allows for the "on the fly" addition of new clients into an already-running client/server application without concern for app-queue naming conflicts. This topic is discussed in the "Client/Server Interaction" section of the MomSys User Guide.*

The new app-queue is created having the set of attributes defined within the app-queue attribute block identified by *AttrBlock*. (Refer to MomAttrSet() for details on possible app-queue attributes.)

AttrBlock points to an app-queue attribute block (i.e., of type `MOM_ATTRBLOCK_APPQUEUE`) that was already initialized via calls to MomAttrSet(), or that is one of the predefined values provided by *X•IPC*, namely:

- `MOM_APPQUEUE_DISK` - Creates a disk app-queue with default settings.
- `MOM_APPQUEUE_DISK_PRIORITY` - Creates a disk app-queue with default settings, with the exception that the created app-queue is defined as a priority-based queue. (The default is time-based.)
- `MOM_APPQUEUE_DISK_REGISTER` - Creates a predefined attribute block used for creating an app-queue that has default attribute settings, with the exception that the created app-queue is automatically registered within the caller's current namespace. (Auto-registration is not the default.) It is an error to register an app-queue name that is already registered. (See Description).
- `MOM_APPQUEUE_DISK_PRIORITY_REGISTER` - Creates a predefined attribute block used for creating an app-queue that has default attribute settings, with two exceptions: The created app-queue is automatically registered within the caller's current namespace–auto-registration is not the default–and the app-queue is delivered as a priority queue. (The default is time.) It is an error to register an app-queue name that is already registered. (See Description).
- `MOM_APPQUEUE_DISK_REGISTER_UPDATE` - Creates an app-queue that has default attribute settings, with the exception that the created app-queue is automatically registered within the caller's current

namespace. (Auto-registration is *not* the default) In case the app-queue already exists, its attributes are updated with the attributes passed in the current call. It is typically used to relocate an app-queue from its current location to a new location and to have all programs that are currently sending messages to that app-queue have their messages now be sent to the new location. (See Section 4.1.6 in the MomSys User Guide for further details.)

If *AttrBlock* has the `MOM_ATTR_SET_AUTO_REGISTER` attribute set, then the call to MomCreate() will automatically update the caller's current *X•IPC* namespace catalog appropriately. Refer to the description of MomAttrSet() for details on default attribute block settings.

ERRORS

Code	Description
MOM_ER_BADAPPQUENAME	Invalid *Name* parameter.
MOM_ER_CAPACITY_LOCALQUEUE	MomSys local queue table full.
MOM_ER_CAPACITY_MR	MomSys message repository full.
MOM_ER_DUPLICATE	Application queue with *Name* already exists.
MOM_ER_INTERNAL	MomSys internal error (refer to log file).
MOM_ER_BADATTRBLOCK	Attribute block not initialized correctly.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_NOTSUPPORTED	Function not supported in current version of MomSys.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`momcreate` {*Name* | `@PRIVATE`} [`time` | `priority`] [`register` | `update`]

ARGUMENTS

Name Name of app-queue to be created or `@PRIVATE`

EXAMPLES

```
xipc> momcreate xyz autoregister
      AQid = 1.0
```

4.1.5 MomDeaccess() – RELEASE HANDLE OF REMOTE APPLICATION QUEUE

NAME

MomDeaccess() – Release Handle of Remote Application Queue

SYNTAX

```
#include "xipc.h"

XINT
MomDeaccess(
        XINT    AQid)
```

PARAMETERS

Name	Description
AQid	App-queue AQid Handle

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

The MomDeaccess() function frees the association of the user with the remote app-queue identified by *AQid*. MomSys keeps track of the number of local users accessing a remote app-queue. When the count drops to zero, MomSys frees the internal resources that supported the remote access.

AQid should reference a remote app-queue. If *AQid* represents a local app-queue, MomDeaccess() returns "success." This has no effect, however, on MomSys resources.

ERRORS

Code	Description
MOM_ER_BADAQID	Invalid AQid.
MOM_ER_INTERNAL	Error occurred in one of the MomSys servers (see log file).
MOM_ER_NONETWORK	Remote queue access cannot be performed since the instance is not configured for network connection.

MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`momdeaccess` *AQid*

ARGUMENTS

AQid App-queue-id of app-queue to deaccess

EXAMPLES

```
xipc> momdeaccess 1.5
      RetCode = 0
```

4.1.6 MomDelete() - DELETE AN APPLICATION QUEUE

NAME

MomDelete() - Delete an Application Queue

SYNTAX

```
#include "xipc.h"

XINT
MomDelete(
        XINT    AQid)
```

PARAMETERS

Name	Description
AQid	App-queue AQid Handle

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

The MomDelete() function deletes the app-queue identified by *AQid* from the caller's local instance. MomDelete() will fail if the specified app-queue is not empty or if any users are waiting to send or receive messages from it.

If the specified app-queue was created with the `MOM_ATTR_AUTO_REGISTER` attribute set, then a successful deletion of the app-queue will additionally deregister the app-queue from the caller's current *X•IPC* namespace.

AQid must reference an app-queue that is within the calling process's local instance.

ERRORS

Code	Description
MOM_ER_BADAQID	No application queue with *AQid*.

MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTEMPTY	The application queue is not empty.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_REMOTE_QUEUE	*AQid* is of a remote app-queue.
MOM_ER_WAITEDON	A user is waiting for a message on *AQid*.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`momdelete` *AQid*

ARGUMENTS

AQid App-queue-id of app-queue to delete

EXAMPLES

```
xipc> momdelete 1.4
      RetCode = 0
```

4.1.7 MomDestroy() - DESTROY AN APPLICATION QUEUE

NAME

MomDestroy() - Destroy an Application Queue

SYNTAX

```
#include "xipc.h"

XINT
MomDestroy(
        XINT    AQid )
```

PARAMETERS

Name	Description
AQid	App-queue AQid Handle

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

The MomDestroy() function deletes the app-queue identified by *AQid* regardless of whether it has messages on it or has users waiting to send or receive messages from it.

If the specified app-queue was created with the `MOM_ATTR_AUTO_REGISTER` attribute set, then a successful deletion of the app-queue will automatically deregister the app-queue from the caller's current *X•IPC* namespace.

AQid must reference an app-queue that is within calling process's local instance.

ERRORS

Code	Description
MOM_ER_BADAQID	No application queue with *AQid*.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).

MOM_ER_REMOTE_QUEUE	*AQid* is of a remote app-queue.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`momdestroy` *AQid*

ARGUMENTS

AQid	App-queue-id of app-queue to destroy

EXAMPLES

```
xipc> momdestroy 1.2
      RetCode = 0
```

4.1.8 MomEvent() - CREATE A MOMSYS EVENT

NAME

MomEvent() - Create a MomSys Event

SYNTAX

```
#include "xipc.h"

XINT
MomEvent(
        ... EventDescr,
        ... Notification)
```

PARAMETERS

Name	Description
EventDescr	Description of MomSys event to be created
Notification	Notification option (see Description below)

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error. (See error codes below. Note: When *Notification* is of an asynchronous variety, an RC of `MOM_ER_ASYNC` means that the event has successfully been placed in the background.)

DESCRIPTION

The MomEvent() function is used to register interest in the occurrence of a MomSys event. If and when the event occurs, the appropriate notification response occurs according to the specified *Notification* argument.

The following table defines the events that can be specified as the *EventDescr* argument within the current version of the product:

`MOM_EV_MSG_STATUS(`*`MsgId, Status`*`)`	This event occurs when a previously sent message, identified by *MsgId*, attains a message status of *Status.* Refer to the MomSys User Guide, Appendix A, Message Status and Tracking Levels, for details on the various stages of a message's movement from sender to receiver process.
`MOM_EV_APPQUE_MSGS_HI(`*`AQid, NumMsgs`*`)`	This event occurs when the number of messages on a local appqueue is/becomes greater than *NumMsgs.*

`MOM_EV_APPQUE_MSGS_LOW(`*`AQid, NumMsgs`*`)` This event occurs when the number of messages on a local appqueue is/becomes less than *NumMsgs.*

The *Notification* argument may be any of the following, where the occurrence of the event causes the specified notification:

`MOM_WAIT`	The calling process blocks synchronously until the event occurs.
`MOM_TIMEOUT(`*`t`*`)`	The calling process blocks synchronously for up to `t` seconds until the event occurs.
`MOM_NOWAIT`	The calling process synchronously checks the status of the event and returns immediately. This Notification argument can be used for polling an event, where such an approach is appropriate.
`MOM_CALLBACK(`*`Func, &Acb`*`)`	MomEvent() returns immediately after registering the event. The specified callback function is invoked when the specified event occurs. Warning: The specified Acb buffer must be kept intact until the callback function is called. The block may, however, be released as part of the callback function's processing. (See the discussion of Asynchronous Operations and Asynchronous Control Blocks in the Advanced Topics Chapter of the *X•IPC* User Guide.)
`MOM_POST(`*`Sid, &Acb`*`)`	MomEvent() returns immediately after registering the event. The specified *X•IPC* semaphore (identified by *`Sid`*) is set when the event occurs. Warning: The specified semaphore and *`Acb`* must be kept intact until the event occurs.
`MOM_IGNORE(&`*`Acb`*`)`	MomEvent() returns immediately after registering the event. The *`Acb's`* completion flag is set when the event occurs. Warning: The specified *`Acb`* must be kept intact until the event occurs.
`MOM_SPAWN(`*`Command, &Acb`*`)`	MomEvent() returns immediately after registering the event. The program specified by the *`Command`* string is started when the event occurs, and the *`Acb's`* completion flag is set. (Note that the *`Acb`* pointer can be NULL.) *`Command`* must be the path of an executable program. The Command string should not include any command arguments. It is not possible to pass parameters to the started command.

An optional `MOM_RETURN` flag may be specified as part of the MomEvent() call. This is done by ORing it to left of the operation's *Notification* argument, as in the following example:

```
MomEvent(. . ., MOM_RETURN | MOM_CALLBACK(UserCallBack,&UserAcb));      .
```

The MOM_RETURN flag—which is only valid when accompanying one of the three *asynchronous* blocking options, MOM_CALLBACK, MOM_POST or MOM_IGNORE—directs *X•IPC* to complete the operation *synchronously* if there is no need to block, and to "go asynchronous" only if the operation cannot be completed immediately. This flag allows a user to issue a MomEvent() call for creating an asynchronous event handler only if the event state has not occurred. Otherwise the function call returns synchronously with a return code of 0 indicating that the event state has occurred.

Events created by MomEvent() are by default attached to the creating *X•IPC* user. This means that, when the user logs out, the event is deleted from the system. The user may override this by logically ORing the MOM_EV_DETACHED flag to the left of the *Notification* option, in which case the event is *not* associated with the creating user.

In such a case, the user may create the event and then log out and even terminate; even so, when the event occurs, the requested action will still take place.

Consider the following example:

```
/*
 * Set event to automatically start program "HeGotIt" when a sent message achieves
 * the status of MOM_STATUS_DELIVERED.
 *
 * MOM_EV_DETACHED flag allows user to log out and exit after sending the message
 * and creating the event.
 */


MOM_MSGID  RetMsgId;
ASYNCRESULT Acb;


MomSend( SomeAQid, "hello world", 12L, MOM_PRIORITY_NORMAL, MOM_TRACK_DELIVERED,
            MOM_REPLY_NONE, &RetMsgId, MOM_WAIT);


MomEvent(
    MOM_EV_MSG_STATUS(RetMsgId, MOM_TRACK_DELIVERED),   /* Set event for msg */
    MOM_EV_DETACHED | MOM_SPAWN("HeGotIt",&Acb));       /* Detach event from caller */
                                                        /* Specify program to spawn */

XipcLogout();

exit();
```

By specifying the MOM_EV_DETACHED option, the user is able to log out and exit after creating the event, without the event being deleted. This option is only applicable when it is ORed to the left of a *Notification* option that does not require the caller's continued presence. Accordingly, the MOM_EV_DETACHED flag is only valid with the MOM_SPAWN *Notification* option since it deals with an executable program which may exist independently of the creator process.

By contrast, it would be an error to specify MOM_EV_DETACHED with other *Notification* options. This is because it doesn't make sense, for example, to specify a user callback function as the *Notification* option and expect it to be invoked *after* the process has terminated.

When MOM_EV_DETACHED is specified with MOM_SPAWN, the user's Acb will not be updated with completion information, even if the event occurs while the user is still logged in. Upon successful return from MomEvent(), the Acb will show an *AsyncStatus* of XIPC_ASYNC_DETACHED and the AUid of the associated

event; the status will never show up as "completed." The only notification that the event occurred will be the execution of the program specified to `MOM_SPAWN`.

MomEvent() Event Semantics

A MomSys event is defined to have occurred whenever the monitored entity is in the awaited state. Depending on the *Notification* argument specified, MomEvent() can be used to poll the *current* state of a MomSys entity, or to wait (synchronously or asynchronously) for the entity to enter a *future* state.

When MomEvent() is called to create a new event and the specified target is *already* in the awaited state, MomEvent() considers the event to have just occurred and the prescribed notification happens, with the qualification that the `MOM_RETURN` option can cause asynchronous notifications to return synchronously, as described above.

MomSys Event Monitoring

Pending MomSys events are treated as ordinary asynchronous MomSys operations in that they are assigned an Asynchronous User ID (AUid) while they are pending. The major advantage of this is that all pending asynchronous MomEvent() operations may be monitored via MomInfoUser() function calls or on the `momview` monitor. Similarly, a MomSys event may be removed from the subsystem via a call to the MomAbortAsync() function. Refer to that function's description for details.

ERRORS

Code	Description
MOM_ER_ASYNC	Operation is being performed asynchronously.
MOM_ER_BADAQID	No application queue with *AQid*.
MOM_ER_BADCOMMANDOPTION	Bad value for command line program specified.
MOM_ER_BADOPTION	Invalid *Options* parameter.
MOM_ER_CAPACITY_ASYNC_USER	MomSys async user table full.
MOM_ER_MSGQDESTROYED	The queue with the message pending is destroyed.
MOM_ER_INTERRUPT	Operation was interrupted.
MOM_ER_MEMORY	No memory for MomEvent control blocks.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_REMOTE_QUEUE	Operation invalid for remote queue.
MOM_ER_TIMEOUT	Operation has timed out.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`momevent` *EventSpecifier EventParms BlockingOpt*

ARGUMENTS

EventSpecifier Event to be tracked; currently only `msg_status`

EventParms Parameters to the *EventSpecifier:*
`msg_status` takes a MsgId variable followed by a tracking level (see Example).

BlockingOpt See the Blocking Options discussion in the `xipc` Interactive Command Processor section of the *X•IPC* User Guide and Reference Manual.

EXAMPLES

```
# Initiate a momevent operation that waits until MomSys message
# MsgId ‘a’ achieves a status of ‘delivered’

xipc>  momevent msg_status a delivered wait
       RetCode = 0
```

4.1.9 MomInfoAppQueue() - GET APPLICATION QUEUE INFORMATION

NAME

MomInfoAppQueue() - Get application queue information.

SYNTAX

```
#include "xipc.h"

XINT
MomInfoAppQueue(
        XINT    AQid,
        MOMINFOAPPQUEUE *InfoAppQueue)
```

PARAMETERS

Name	Description
AQid	App-Queue AQid Handle, or MOM_INFO_FIRST, or MOM_INFO_NEXT(*AQid*)
InfoAppQueue	Pointer to a structure of type MOMINFOAPPQUEUE where application queue information will be returned

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomInfoAppQueue() may be used to query the status of an app-queue. The *AQid* argument may be specified as one of the following:

- *AQid* - an app-queue ID identifying a specific app-queue
- MOM_INFO_FIRST - identifies the first valid app-queue ID
- MOM_INFO_NEXT(*AQid*) - identifies the next valid app-queue ID, following *AQid*

A program reviewing the status of all app-queues within an instance should call MomInfoAppQueue() specifying MOM_INFO_FIRST, followed by repeated calls to the function specifying MOM_INFO_NEXT until the MOM_ER_NOMORE error code is returned.

This function returns information regarding an application queue. The information is received in a structure of type MOMINFOAPPQUEUE. The structure is defined as follows. Note that fields marked as [L] are populated with valid data when referencing a Local *AQid*. Fields marked as [R] are populated with valid data when referencing a Remote *AQid*. Otherwise the fields are undefined.

```
typedef struct _MOMINFOAPPQUEUE
{
    MOM_ATTRBLOCK_APPQUEUE AttrBlock;        /* [L,R] Attr block of app-queue */
                                             /* See MomAttrSet() for details */
    XINT AQid;                               /* [L,R] of AppQueue*/
    XINT CreateTime;                      /* [L,R] of creation on entry */
    UXINT CreateUid;                         /* [L] Uid of creating user */
    XINT LocalRemote;                        /* [L,R] MOM_APPQUEUE_LOCAL or
                                                      MOM_APPQUEUE_REMOTE */
    XINT RemoteState;                        /* [R] State data re: remote app-queue:
                                                     MOM_APPQUEUE_VERIFIED or
                                                     MOM_APPQUEUE_NOTVERIFIED */
    XINT CountMsgsIn;                        /* [L] Total num of msgs enqueued on queue */
    XINT CountMsgsOut;                       /* [L] Total num of msgs dequeued from queue */
    XINT CountMsgsSentTo;                    /* [R] Total num of msgs sent to remote queue */
    XINT NumMsgs;                            /* [L] Current number of msgs on app-queue */
    XINT NumBytes;                           /* [L] Current number of bytes on app-queue */
    XINT NumMsgsHWM;                      /* [L] Number of messages high water mark */
    XINT NumBytesHWM;                        /* [L] Number of bytes high water mark */
    XINT AttachedUid;                        /* [L] Uid of the attached user (if any)
                                                    [This field is RESERVED for later
                                                    versions of XIPC]*/
    XINT WListTotalLength;                   /* [L] Number of items in app-queue's WList */
    XINT WListInitialCursor;              /* [L] Initial value for scanning WList */
                                             /*     See MomInfoAppQueueWList() below. */
    MOM_APPQUEUEWLISTITEM WListFirstItem;    /* [L] Structure defined below */

    CHAR Name[XIPC_LEN_XIPCNAME+1];          /* [L,R] Name of application queue */
    CHAR InstName[XIPC_LEN_NETNAME+1];       /* [L,R] Instance name where app-queue is */
    CHAR NodeName[XIPC_LEN_HOSTNAME+1];      /* [L,R] Node name where app-queue is */

} MOMINFOAPPQUEUE;
```

```
typedef struct _MOM_APPQUEUEWLISTITEM
{
    XINT  OpCode;                   /* MOM_OPCODE_SEND or MOM_OPCODE_RECEIVE */
    union
    {
        struct
        {
            XINT Uid;               /* User (or AUid) blocked */
            XINT MsgSize;           /* Sending Msg */
            XINT MsgPrio;           /* Msg Priority (Combined Trip & Queue)*/
            XINT TrackingLevel;     /* Tracking Level */
            UXINT Time;             /* When the MomSend() blocked */
        } Send;

        struct
        {
            XINT Uid;               /* User (or AUid) blocked */
            XINT MsgSequence;       /* See Advanced Msg Selection appendix ... */
            XINT MsgSelector;       /* ... for possible values for these 2 fields */
            UXINT Time;             /* When the MomReceive() blocked */
        } Receive;

    } u;

} MOM_APPQUEUEWLISTITEM;
```

AQid may reference either a local or a remote app-queue, relative to the calling process. If *AQid* references a local app-queue, then the MOMINFOAPPQUEUE structure–or, more precisely, that part of it marked as [L] above–is populated with data about the app-queue. (The data is retrieved from the local instance.) If the app-queue referenced by *AQid* is remote, then the function fills in as much information about the remote app-queue as is known within the calling process's local instance at the time of the call.

A large part of data returned about the app-queue is returned within the *AttrBlock* element of the MOMINFOAPPQUEUE data structure. For a local app-queue, this block reflects the attribute block values that were passed to MomCreate() when the app-queue was originally created. For a remote app-queue this block is filled with whatever data is locally known about the remote app-queue. Refer to the MomAttrSet() function description for details about fields within the MOM_ATTRBLOCK_APPQUEUE data structure.

Refer to the MomSys User Guide section "Understanding MomSys Information Verbs" for sample programs.

ERRORS

Code	Description
MOM_ER_BADAQID	No application queue with specified *AQid*.
MOM_ER_BADBUFFER	Return buffer pointer is NULL.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance
MOM_ER_NOMORE	No more app-queues.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`mominfoappqueue` *AQid* | `all` | `first` | `next(`***AQid***`)`

ARGUMENTS

One of the following:

AQid	App-queue-id of app-queue for which to acquire information
`all`	Acquire information for all app-queues
`first`	Acquire information for first app-queue
`next(`***AQid***`)`	Acquire information for next app-queue after ***AQid***

EXAMPLES

```
xipc> mominfoappqueue 1.2
      AQid = 1.2        Name = xyz
      Location = 'LOCALNODE:someinstance'
      AppQueue is LOCAL DISK Queue
      Natural Sequence is TIME
      App-queue is not AUTO Registered
      CreateTime: Fri Oct 25 12:02:47 1996
      CreateUid = 3
      CountMsgIn: 204         CountMsgOut: 200
      NumMsgs: 4              NumBytes: 10240
      NumMsgHWM: 380          NumBytesHWM: 90840
      Retire Time: 0          Expire Time: 30 minutes
```

4.1.10 MomInfoAppQueueWList() - GET APP-QUEUE WAIT-LIST INFORMATION

NAME

MomInfoAppQueueWList() - Get app-queue Wait-List information.

SYNTAX

```
#include "xipc.h"

XINT
MomInfoAppQueueWList(
        XINT                    AQid,
        XINT                    *Cursor,
        MOM_APPQUEUEWLISTITEM   *WListItem)
```

PARAMETERS

Name	Description
AQid	AQid handle of app-queue whose wait-list data is to be retrieved.
Cursor	Before the call, **Cursor* (the integer that Cursor points to) should represent the position of the current item within the wait-list. During the call, **Cursor* is advanced to the position of the next item, the data of which is then retrieved and stored in *WListItem.
WListItem	Pointer to a structure of type `MOM_APPQUEUEWLISTITEM` which is to be populated with wait-list information. (The data structure is defined as part of the MomInfoAppQueue() description.)

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomInfoAppQueueWList() may be used to traverse an app-queue's wait list. Note that MomInfoAppQueue() returns the *first* list item as part of the `MOMINFOAPPQUEUE` data that it retrieves. MomInfoAppQueue() additionally returns with the initial cursor value for referencing the first item. This cursor should be used as the starting point for using this function to traverse the remaining (2nd through last) WList items.

A program reviewing the second through last wait-list items would make repeated calls to MomInfoAppQueueWList() using the cursor until the `MOM_ER_NOMORE` error code is returned. Note that *Cursor* is first moved to reference the next WList item as part of each such call. The `MOM_APPQUEUEWLISTITEM` structure is defined as part of the MomInfoAppQueue() function description. Refer there for details.

Refer to the MomSys User Guide section "Understanding MomSys Information Verbs" for sample programs.

ERRORS

Code	Description
MOM_ER_BADAQID	No application queue with *AQid*.
MOM_ER_BADBUFFER	Return buffer pointer is NULL.
MOM_ER_BADVAL	Cursor pointer is NULL.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_REMOTE_QUEUE	*AQid* is of a remote app-queue.
MOM_ER_NOMORE	No more wait queue list entries.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

Incorporated as part of the `mominfoappqueue` command

4.1.11 MomInfoLink() - GET NETWORK LINK INFORMATION

NAME

MomInfoLink() - Get network link information.

SYNTAX

```
#include "xipc.h"

XINT
MomInfoLink(
        XINT LinkId,
        MOMINFOLINK *InfoLink)
```

PARAMETERS

Name	Description
LinkId	Id of Link, or `MOM_INFO_FIRST`, or `MOM_INFO_NEXT(LinkId)`
InfoLink	Pointer to a structure of type `MOMINFOLINK` where link information will be returned.

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomInfoLink() may be used to query the status of a network link connecting to a remote instance. The *LinkId* argument may be specified as one of the following values:

- *LinkId* - an integer Link ID identifying a specific link
- `MOM_INFO_FIRST` - identifying the first valid link-ID within the instance
- `MOM_INFO_NEXT(LinkId)` - identifying the next valid link-ID, following *LinkId*

A program reviewing the status of all links within an instance would call MomInfoLink() specifying `MOM_INFO_FIRST`, followed by repeated calls to the function specifying `MOM_INFO_NEXT` until the `MOM_ER_NOMORE` error code is returned.

The MOMINFOLINK data structure is defined as follows:

```
typedef struct _MOMINFOLINK
{
    XINT LinkId;         /* Link Id */
    XINT Attributes;     /* Attributes of link
                            [This field is RESERVED for later
                            versions of XIPC]*/
    XINT Protocol;          /* Protocol of link:
                            XIPC_PROT_TCPIP */
    XINT LinkStatus;     /* Current status of link:
                            MOM_LINKSTATUS_UP or MOM_LINKSTATUS_DOWN */
    XINT InCommServerPID;    /* Process ID of CSI supporting link */
    XINT OutCommServerPID;   /* Process ID of CSO supporting link */
    XINT CountMsgsSent;      /* Total messages sent out over link */
    XINT CountMsgsReceived;  /* Total messages received in over link */
    XINT NumBacklogMsgs; /* Number of messages waiting to exit over link */
    UXINT StartupTime;       /* Time link was started (in UTC form) */
    CHAR TimeOut[XIPC_LEN_TIMELIMIT +1];     /* When link times out */
    CHAR RetryInterval[XIPC_LEN_TIMELIMIT +1];   /* How often are retries attempted */
    CHAR RemoteInstance[XIPC_LEN_NETNAME +1];    /* Name of remote instance */
    CHAR RemoteNode[XIPC_LEN_HOSTNAME +1];       /* Name of remote node */
}
MOMINFOLINK;
```

The following are descriptions of the MOMINFOLINK data fields:

LinkId	The integer ID of the link.
Attributes	[This field is RESERVED for future product versions]
Protocol	XIPC_PROTOCOL_TCPIP
LinkStatus	Status can be one of the following values: • MOM_LINKSTATUS_UP • MOM_LINKSTATUS_DOWN
InCommServerPID	The Process-ID of the CSI process that is managing this link.
OutCommServerPID	The Process-ID of the CSO process that is managing this link.
CountMsgsSent	The total number of messages that have been sent out over the link since the link was started.
CountMsgsReceived	The total number of messages that have come in over the link since the link was started.
NumBacklogMsgs	The number of outbound messages that are currently queued up for shipment.
StartupTime	The time that the link was started.
TimeOut	String defining amount of time that is considered a time-out for this link.
RetryInterval	String defining how often to try to connect to remote end of link.
RemoteInstance	The name of the instance at the opposite end of the link.
RemoteNode	The network node of the instance at the opposite end of the link.

ERRORS

Code	Description
MOM_ER_BADLINKID	No link with *LinkId*.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_BADBUFFER	Return buffer pointer is NULL.
MOM_ER_NOMORE	No more links.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`mominfolink` *LinkId* | `all` | `first` | `next(`*LinkId*`)`

ARGUMENTS

One of the following:

LinkId	Link-Id of instance link for which to acquire information.
`all`	Acquire information for all links.
`first`	Acquire information for first link
`next(`*LinkId*`)`	Acquire information for next link after ***`LinkId`***.

EXAMPLES

```
xipc> mominfolink first
      LinkId: [1]
      Remote Node: 'helios'    Remote Instance: 'test'
      Network Protocol: TCPIP   Link Status: DOWN
      CountMsgSent: 29880  CountMsgReceived: 102 NumBacklogMsg: 24
      StartupTime: Wed Sep 30 19:05:10 1996


xipc> mominfolink all

      Id    Instance          Protocol    Status      Messages
      ------------------------------------------------------
      1     helios:test       TCP/IP      DOWN        24
      2     titan:test1       TCP/IP      UP          14
      3     juno:product      TCP/IP      UP          0
      4     moon:test2        TCP/IP      DOWN        1040
```

4.1.12 MomInfoMessage() - GET MESSAGE INFORMATION

NAME

MomInfoMessage() - Get message status and other information.

SYNTAX

```
#include "xipc.h"

XINT
MomInfoMessage(
        MOM_MSGID MsgId,
        MOMINFOMESSAGE *InfoMessage)
```

PARAMETERS

Name	Description
MsgId	Message Id of message
InfoMessage	Pointer to a structure of type `MOMINFOMESSAGE` , where message information will be returned.

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomInfoMessage() may be used to query status of a message from either the *sender* or *receiver* perspective.

- Calling MomInfoMessage() from the *sender* perspective is useful for tracking the status or recalling details about a previously sent message. From the sender perspective, the *MsgId* argument is the message-id that was returned by MomSend() for identifying the message sent. This message-id value is only valid within the context of the sending process's local instance.
- Calling MomInfoMessage() from the *receiver* perspective is useful for learning details about a previously received message. From the receiver perspective, the *MsgId* argument is the message-id that was returned by MomReceive() when the message was received. This message-id value is only valid within the context of the receiving process's local instance at the time of the MomReceive() call. **Note:**

MomReceive() can be used to return the MOMINFOMESSAGE data *directly*, as part of the MomReceive() call. Refer to the description of MomReceive() for details.

The MOMINFOMESSAGE data structure is defined as follows:

```
typedef struct _MOMINFOMESSAGE
{
    MOM_MSGID MsgId;     /* Message Id */
    XINT MsgLength;          /* Length of message */
    XINT AQid;               /* AQid that the message was sent to */
    XINT ReplyAQid;          /* AQid of response app-queue */
    UXINT SendTime;          /* Time message was sent (in UTC form) */
    UXINT ExpirationTime;    /* Time message will expire (in UTC form) */
    XINT TripPriority;       /* Trip priority of message */
    XINT QueuePriority;      /* App-queue priority of message */
    XINT TrackingLevel;      /* Tracking Level that the message was sent with:
                                MOM_TRACK_SHIPPED or
                                MOM_TRACK_DELIVERED */
    XINT LatestStatus;       /* Latest known message status:
                                MOM_STATUS_HELD,
                                MOM_STATUS_SHIPPED or
                                MOM_STATUS_DELIVERED */
    XINT SenderUid;          /* Uid of sending user */

    CHAR SenderInstance[XIPC_LEN_NETNAME +1];  /* Name of Instance of sending user */
    CHAR SenderNode[XIPC_LEN_HOSTNAME +1];      /* Name of Network node of sending user */
    CHAR MessageTag[XIPC_LEN_MESSAGETAG +1];   /* Message Tag */
                                                /* [This field is RESERVED for
                                                   later versions of XIPC]*/
    CHAR DataTranslationType[XIPC_LEN_DTSDATATYPE+1]; /* Message's data translation
                                                         /* type [This field is RESERVED
                                                         /* for later versions of XIPC]*/
}
MOMINFOMESSAGE;
```

The following are descriptions of the MOMINFOMESSAGE data fields. The interpretation of the returned data fields (for certain fields) depends on the caller's perspective. These distinctions are mentioned below, where necessary.

MsgId	Sender: MsgId is the message-id that was returned from the MomSend() call that submitted the message into the sender's local instance. Receiver: MsgId is the message-id that was returned from the MomReceive() call that retrieved the message.
MsgLength	The number of bytes in the message.
AQid	Sender: AQid identifies the targeted app-queue to which the message was sent. Receiver: AQid identifies the source app-queue from which the message was received.
ReplyAQid	The AQid of the reply app-queue associated with the message.
SendTime	The UTC (Coordinated Universal Time) time that the message was sent.
ExpirationTime	The UTC (Coordinated Universal Time) time that the message will expire.
TripPriority	The "trip" priority value that was specified as part of the MomSend() call that sent the message.

`QueuePriority`	The "queue" priority value that was specified as part of the MomSend() call that sent the message.
`TrackingLevel`	The tracking-level value that was specified as part of the MomSend() call that sent the message.
`LatestStatus`	Sender: LatestStatus is the latest known status of the sent message. A message may be tracked to differing degrees depending on the tracking-level specified when the message is sent. Refer to the MomSys User Guide, Appendix A, Message Status and Tracking Levels, for more details. Receiver: LatestStatus is either `MOM_STATUS_SHIPPED` or `MOM_STATUS_DELIVERED`, depending on whether the message was removed from the app-queue by MomReceive() or not.
`SenderUid`	The instance user-id of the sending user.
`SenderInstance`	The instance name of the sending user.
`SenderNode`	The network node name of the sending instance.
`MessageTag`	`[This field is RESERVED for future product versions]`
`DataTranslationType`	`[This field is RESERVED for future product versions]`

ERRORS

Code	Description
MOM_ER_BADBUFFER	Return buffer pointer is NULL.
MOM_ER_NOMSG	Specified message not found.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGED IN	User not logged into instance (User never logged in, was aborted or disconnected).
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`mominfomessage` *MsgId*

ARGUMENTS

MsgId Message-Id of message for which to acquire information.

EXAMPLES

```
xipc> momsend 1.0 "This is another message" normal shipped c nowait
```

```
xipc> mominfomessage c
       MsgId: [849730340.512,849730340.512]
       AQid: 1.0       ReplyAQid: 0.-1
       SendTime: Wed Dec  4 15:15:34 1996
       ExpirationTime: Wed Dec 31 19:00:00 1996
       Trip Priority: 794744   Queue Priority: 32768
       Tracking Level: SHIPPED
       LatestStatus:   SenderUid: 1
       Sender Node: 'LOCALNODE'      Sender Instance: 'test3'
```

4.1.13 MomInfoSys() - GET MOMSYS INFORMATION

NAME

MomInfoSys() - Get MomSys subsystem information

SYNTAX

```
#include "xipc.h"

XINT
MomInfoSys(
        MOMINFOSYS  *InfoSys,
        MOMINFOMR   *InfoMR,
        MOMINFOCM   *InfoCM)
```

PARAMETERS

Name	Description
InfoSys	Pointer to a structure of type `MOMINFOSYS` where general MomSys data will be returned, or `NULL`.
InfoMR	Pointer to a structure of type `MOMINFOMR` where MomSys Message Repository information will be returned, or `NULL`.
InfoCM	Pointer to a structure of type `MOMINFOCM` where MomSys Communication Manager information will be returned, or `NULL`.

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomInfoSys() may be used to query status of the MomSys subsystem. MomInfoSys() provides access to three aspects of MomSys:

- *General subsystem information*
- *Message Repository (MR) information*
- *Communication Manager (CM) information*

The call to MomInfoSys() may request data regarding any combination of these three aspects, depending on the combination of arguments passed to the function.

The function takes three positional arguments:

- The first argument is a pointer to a data structure of type `MOMINFOSYS`, or it can be `NULL`. Unless this pointer is `NULL`, the function populates the data structure with general information about the configuration and status of the MomSys subsystem.
- The second argument is a pointer to a data structure of type `MOMINFOMR`, or it can be `NULL`. Unless this pointer is `NULL`, the function populates the data structure with specific information about the configuration and status of the MomSys message repository.
- The third argument is a pointer to a data structure of type `MOMINFOCM`, or it can be `NULL`. Unless this pointer is `NULL`, the function populates the data structure with specific information about the configuration and status of the MomSys communication manager.

The `MOMINFOSYS` data structure is defined as follows:

```
typedef struct _MOMINFOSYS
{
    XINT    MaxUsers;                       /* Max. configured users */
    XINT    MaxDiskAppQueues;               /* Max. configured disk app-queues */
    XINT    MaxMemAppQueues;            /* Max. configured mem app-queues */
    XINT    MaxRemoteAppQueues;             /* Max. configured remote app-queues */
    XINT    MaxMemText;                     /* Max. configured mem-queue text */
    XINT    MaxMemMessages;                 /* Max. configured mem-queue messages */
    XINT    MaxMsgLength;                   /* Max. configured message size. */
    XINT    CurrUsers;                      /* Current number of current users */
    XINT    CurrDiskAppQueues;              /* Current number of disk app-queues */
    XINT    CurrMemAppQueues;               /* Current number of mem app-queues */
    XINT    CurrRemoteAppQueues;        /* Current number of remote app-queues */
    XINT    CurrMemText;                /* Current mem-queue text in use */
    XINT    CurrMemMessages;            /* Current mem-queue messages in use */
    XINT    EpisodeNumber;                  /* Number of current episode (0 is first) */
    UXINT   CreateTime;                     /* Instance create time (UTC format) */
    UXINT   StartTime;                      /* Instance start time (UTC format) */
    CHAR    InstName[XIPC_LEN_NETNAME +1];       /* Instance network name */
    CHAR    CfgFileName[XIPC_LEN_PATHNAME +1];   /* Instance file name */
    CHAR    LogFileName[XIPC_LEN_PATHNAME +1];   /* Instance log file name */
    CHAR    NameSpace[XIPC_LEN_XIPCNAME+1];  /* Namespace affiliation*/

}
MOMINFOSYS;
```

The following fields (included above) in this data structure are *reserved* for later versions of *X•IPC* and should not be used:

```
    XINT    MaxMemAppQueues;            /* Max. configured mem app-queues */
    XINT    MaxMemText;                     /* Max. configured mem-queue text */
    XINT    MaxMemMessages;                 /* Max. configured mem-queue messages */
    XINT    CurrMemAppQueues;               /* Current number of mem app-queues */
    XINT    CurrMemText;                /* Current mem-queue text in use */
    XINT    CurrMemMessages;            /* Current mem-queue messages in use */
    XINT    EpisodeNumber;                  /* Number of current episode (0 is first) */
    CHAR    LogFileName[XIPC_LEN_PATHNAME +1];   /* Instance log file name */
    CHAR    Namespace[XIPC_LEN_XIPCNAME+1];      /* Namespace affiliation*/
```

The MOMINFOMRX data structure is defined as follows:

```
typedef struct _MOMINFOMRX
{
    XINT    CountReads;                     /* Total reads from MR */
    XINT    CountInserts;                   /* Total inserts into MR */
    XINT    CountDeletes;                   /* Total deletes from MR */
    XINT    CountUpdates;                   /* Total updates within MR */
    XINT    CurrSize;                       /* Current size of MR */
    XINT    CurrFree;                       /* Free space in MR */
    XINT    RetiredMsgs;                    /* Not implemented */
    XINT    ExpiredMsgs;                    /* Not implemented */
    XINT    CleanupInProgress;                  /* 1 - Cleanup in progress; 0 - Not */
    CHAR    FileName[XIPC_LEN_PATHNAME +1];         /* Msg repository inbound */
    CHAR    JournalExpiredMsgs[XIPC_LEN_PATHNAME +1];    /* For expired messages */
    CHAR    JournalRetiredMsgs[XIPC_LEN_PATHNAME +1];    /* For purged messages */

}
MOMINFOMRX;
```

The MOMINFOMR data structure is defined as follows:

```
typedef struct MOMINFOMR
{
    MOMINFOMRX MRI;                             /* MRI information */
    MOMINFOMRX MRO;                             /* MRO information /
    UXINT NextCleanupTime;                      /* Next scheduled MR Cleanup */
    CHAR ScheduleCleanup[MOM_LEN_SCHEDULE + 1]   /* MR Cleanup schedule */
    CHAR ScheduleCompact[MOM_LEN_SCHEDULE + 1]   /* MR Compaction schedule */
}
MOMINFOMR;
```

The MOMINFOCM data structure is defined as follows:

```
typedef struct _MOMINFOCM
{

    XINT    MaxLinks;           /* Max. configured number of links */
    XINT    CurrLinksUp;    /* Current number of links in UP state */
    XINT    CurrLinksDown;      /* Current number of links in DOWN state */
    XINT    ActiveCSIs;         /* Number of CSIs running */
    XINT    ActiveCSOs;         /* Number of CSOs running */
}
MOMINFOCM;
```

ERRORS

Code	Description
MOM_ER_BADBUFFER	Return buffer pointer is `NULL`.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

```
mominfosys
```

ARGUMENTS

none

EXAMPLES

```
xipc>  mominfosys

            [MOMSYS SYSTEM INFORMATION]
            ---------------------------
Instance Name: ‘test’               Instance Cfg File ‘/usr/sample’
Instance NameSpace: 'projectSnowball'
Max Users: 20                       Current Users: 7
Max Disk AppQueues: 15              Current Disk AppQueues: 12
Max Memory AppQueues: 0             Current Memory AppQueues: 0
Max Memory Text: 0            Max Message Length: 1024
CreateTime: Wed Dec 13 19:07 1996


         [MOMSYS MESSAGE REPOSITORY SERVER INFORMATION]
         ---------------------------------------------
 Counter                 MRI                     MRO
 ============================================================
 MR Path                 /home/xipc/commIn /home/xipc/commIn
 Number of Reads         108                     0
 Number of Inserts 22                    0
 Number of Deletes 14                    0



            [MOMSYS COMM SERVERS INFORMATION]
            ---------------------------------
Max Links: 31                       Active CSI, CSO:  10, 10
Current Active Links: 12      Current Inactive Links: 2
```

4.1.14 MomInfoUser() - GET MOMSYS USER INFORMATION

NAME

MomInfoUser() - Get MomSys user information.

SYNTAX

```
#include "xipc.h"

XINT
MomInfoUser(
        XINT Uid,
        MOMINFOUSER *InfoUser)
```

PARAMETERS

Name	Description
Uid	A valid User Id, or `MOM_INFO_FIRST`, or `MOM_INFO_NEXT(Uid)`
InfoUser	Pointer to a structure of type `MOMINFOUSER` where user information will be returned

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomInfoUser() may be used to query the status of a MomSys user. The *Uid* argument may be specified as one of the following values:

- *Uid* - an integer User ID identifying a specific user
- `MOM_INFO_FIRST` - identifying the first valid User ID within the instance
- `MOM_INFO_NEXT(Uid)` - identifying the next valid User ID, following *Uid*

A program reviewing the status of all users currently within an instance would call MomInfoUser() specifying `MOM_INFO_FIRST`, followed by repeated calls to the function specifying `MOM_INFO_NEXT` until the `MOM_ER_NOMORE` error code is returned.

This function retrieves MomSys status information regarding an *X•IPC* user. The information is received in a structure of type MOMINFOUSER. The structure is defined as follows:

```
typedef struct _MOMINFOUSER
{
    XINT Uid;                    /* User Id */
    XINT Pid;                    /* Process Id of user */
    XINT Tid;                    /* Thread Id of user */
    UXINT LoginTime;             /* Time of login (UTC) */
    XINT CountMsgsSent;          /* Number of messages sent by user */
    XINT CountMsgsReceived;      /* Number of messages received by user*/

    XINT WListTotalLength;               /* Should in V.3 never exceed 1 */
    XINT WListInitialCursor;         /* Initial value for scanning WList */
                                         /* In V.3 this field is not used */
    MOM_USERWLISTITEM WListFirstItem;    /* Structure defined below */

    XINT AListTotalLength;               /* Number of async ops for user */
    XINT AListInitialCursor;         /* Initial value for scanning AList */
                                         /* See MomInfoUserAList() below. */
    MOM_USERALISTITEM AListFirstItem;    /* Structure defined below */
    CHAR Name[XIPC_LEN_XIPCNAME+1];      /* Name of user */

} MOMINFOUSER;
```

```
typedef struct _MOM_USERWLISTITEM
{
    XINT OpCode; /* MOM_OPCODE_SEND, MOM_OPCODE_RECEIVE, or MOM_OPCODE_EVENT */
    UXINT TimeOut;   /* (UTC) Time that operation will time out */
    union
    {
        struct
        {
            XINT AQid;                  /* App-queue blocked */
            XINT MsgSize;               /* Sending Msg */
            XINT MsgPrio;               /* Msg Priority (combined) */
            XINT TrackingLevel;         /* of sent message */
            UXINT Time;                 /* that MomSend() blocked */
        }
        Send;

        struct
        {
            XINT AQid;                  /* App-queue blocked */
            XINT MsgSequence;           /* Natural */
            XINT MsgSelector;
            UXINT Time;                 /* that MomReceive() blocked */
        }
        Receive;

        struct
        {
            XINT EventCode;             /* Event description */
            UXINT Time;                 /* that MomEvent() blocked */
        }
        Event;
    }
    u;
} MOM_USERWLISTITEM;


typedef struct _MOM_USERALISTITEM
{
    XINT    AUid;   /* AUid of asynchronous operation */
}
MOM_USERALISTITEM;
```

Refer to the MomSys User Guide section "Understanding MomSys Information Verbs" for sample programs.

ERRORS

Code	Description
MOM_ER_BADBUFFER	Return buffer pointer is NULL.
MOM_ER_BADUID	No user with specified *Uid*.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_NOMORE	No more users.

XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

SYNTAX

`mominfouser` *UserId* | `all` | `first` | `next(`*UserId*`)`

ARGUMENTS

One of the following:

UserId	User-Id of user for which to acquire information.
`all`	Acquire information of all users.
`first`	Acquire information of first user.
`next(`*UserId*`)`	Acquire information of next user after ***UserId***.

EXAMPLES

```
xipc> mominfouser 2
      User Id: 2        Name: 'tester'
      Pid: 21573        Tid: 0
      LoginTime: Wed Sep 30 12:05:10 1996
      CountMsgSent: 5021      CountMsgReceived: 1450
```

4.1.15 MomInfoUserAList() - GET MOMSYS USER ASYNC-LIST INFORMATION

NAME

MomInfoUserAList() - Get MomSys user Async-List information

SYNTAX

```
#include "xipc.h"

XINT
MomInfoUserAList(
        XINT                Uid,
        XINT                *Cursor,
        MOM_USERALISTITEM   *AListItem)
```

PARAMETERS

Name	Description
Uid	User-id of user whose async-list data is to be retrieved.
Cursor	Before the call, **Cursor* (the integer that Cursor points to) should represent the position of the current item within the async-list. During the call, **Cursor* is advanced to the position of the next item, the data of which is then retrieved and stored in *AListItem.
AListItem	Pointer to a structure of type `MOM_USERALISTITEM` which is to be populated with async-list information. (The data structure is defined as part of the MomInfoUser() description.)

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomInfoUserAList() may be used to traverse a MomSys user's AList of asynchronous operations. Note that MomInfoUser() returns the *first* list item as part of the `MOMINFOUSER` data that it retrieves. MomInfoUser() additionally returns with the initial cursor value for referencing the first item. This cursor should be used as the starting point for using this function to traverse the remaining (2nd through last) AList items.

A program reviewing the second through last asynchronous operations pending for a user would make repeated calls to MomInfoUserAList() until the `MOM_ER_NOMORE` error code is returned. Note that *Cursor* is first moved to reference the next AList item as part of each call.

The `MOM_USERALISTITEM` structure is defined as part of the MomInfoUser() function description. Refer there for details.

Refer to the MomSys User Guide section "Understanding MomSys Information Verbs" for sample programs.

ERRORS

Code	Description
MOM_ER_BADBUFFER	Return buffer pointer is NULL.
MOM_ER_BADUID	No user with specified *Uid*.
MOM_ER_BADVAL	Cursor pointer is NULL.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_NOMORE	No more user async-list entries.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.

INTERACTIVE COMMAND

Information is provided as part of the `mominfouser` interactive command.

4.1.16 MomReceive() - RECEIVE A MESSAGE

NAME

MomReceive() - Receive a message from an Application Queue

SYNTAX

```
#include "xipc.h"

XINT
MomReceive(
        XINT SourceAQid,
        XANY *MsgBuf,
        XINT MsgBufLen,
        ... MsgSpecifier,
        XINT *RetReplyAQid,
        MOM_MSGID *RetMsgId,
        MOMINFOMESSAGE *RetInfoMessage,
        ... BlockOpt)
```

PARAMETERS

Name	Description
SourceAQid	Source app-queue AQid handle
MsgBuf	Message buffer for receiving message
MsgBufLen	Length of message buffer
MsgSpecifier	Message specification parameter indicating which message is to be retrieved from the app-queue
RetReplyAQid	Pointer to an integer variable that will be set with the sender-specified reply-Aqid; or, the value can be `NULL`
RetMsgId	Pointer to a `MOM_MSGID` variable that will be set with the message id of the received message; or, the value can be `NULL`
RetInfoMessage	Pointer to `MOMINFOMESSAGE` data structure variable that will be populated with extended information about the returned message; or, the value can be `NULL`
BlockOpt	Blocking option

RETURNS

Value	Description
RC >= 0	Length of received message.
RC < 0	Error (see error codes below).

DESCRIPTION

The MomReceive() function is used to receive a message from an application queue. The receiving application identifies the source application queue via the *SourceAQid* argument. The *SourceAQid* handle will have typically been acquired via a prior call to MomCreate() or MomAccess(). *SourceAQid* must reference an app-queue that is within the calling process's current local instance. Note, however, that when a specified MsgId is on an AQid other than the AQid specified (i.e., the AQid associated with the MsgId), the message will still be retrieved.

MsgBuf and *MsgBufLen* define the location and length of the receiving buffer. If the incoming message is larger than *MsgBufLen* bytes, an error code is returned and the message is not retrieved. The macro `MOM_TRUNCATE(`*`MsgBufLen`*) can be specified instead to request truncation of the message if it exceeds *MsgBufLen* bytes in size. In such a case, the first *MsgBufLen* bytes of the message are copied into *MsgBuf*, the remaining bytes are lost and no error is returned.

It is also possible to retrieve the length of a message prior to the actual MomReceive() so that space can be allocated for receipt of the entire message, if needed and desired. This is achieved by doing a MomReceive() with the `MOM_TRUNCATE(`*`1`*`)` and `MOM_NOREMOVE` options, followed by a `MomInfoMessage(`*`of the message`*`).` This returns the message length, allowing the user to allocate space, if appropriate, prior to doing a MomReceive() for the message itself.

Message specification is accomplished via the *MsgSpecifier* argument to MomReceive(). *MsgSpecifier* identifies which message is to be retrieved from *SourceAQid*. Proper utilization of this argument requires a basic understanding of how messages reside on an app-queue.

When an app-queue is created, one of its defining attributes specifies the *natural* sequencing of messages on that app-queue. (Refer to MomAttrSet() for details on app-queue attribute specification.) An app-queue has one of the following attributes: natural sequencing by *Time* or natural sequencing by *Priority*.

Message specification semantics can differ depending on the natural sequencing of messages on an app-queue. For example, specifying the “first” message on an app-queue means the “oldest” message if the natural sequencing is by *Time*; it means the “highest” priority message if the natural sequencing is by *Priority*.

The following are possible values for *MsgSpecifier* :

`MOM_MESSAGE_FIRST`	Retrieve the first message from natural sequence. If Time, the oldest message is returned. If Priority, the highest priority message is returned.
`MOM_MESSAGE_LAST`	Retrieve the last message from natural sequence. If Time, the newest message is returned. If Priority, the lowest priority message is returned.
`MOM_MESSAGE_NEXT(`*`MsgId`*`)`	Retrieve the next message from within natural sequence following the message identified by *`MsgId`*.*. If Time, the next oldest message is returned. If Priority, the next highest priority message is returned.
`MOM_MESSAGE_PREV(`*`MsgId`*`)`	Retrieve the previous message from within natural sequence following the message identified by *`MsgId`*.*. If Time, the previous oldest message is returned. If Priority, the previous highest priority message is returned.
`MOM_MESSAGE_DIRECT(`*`MsgId`*`)`	Retrieve the message identified by *`MsgId`*. *.
`MOM_MESSAGE_DIRECT_RMT (`*`RmtNode, RmtInstance, RmtMsgId`*`)`	Retrieve a message based on its *remote* identification: – *`RmtNode`* is name of sender node. – *`RmtInstance`* is name of sender instance. – *`RmtMsgId`* is the MsgId that was assigned to the message when it was sent via the sender instance.
`MOM_MESSAGE_REPLYTO(`*`MsgId`*`)`	Retrieve the (server) response message to the (client) request message that was previously sent by MomSend() and identified as *`MsgId`*.

** Note: The message represented by MsgId, where indicated with an asterisk, must still be on the app-queue at the time of the MomReceive() call. This is typically accomplished by having performed an earlier call to MomReceive() in which the MOM_NOREMOVE flag was set. The MsgId returned from that call can serve as the "cursor" for subsequent MomReceive() calls.*

The above listed values for *MsgSpecifier* are actually macros that are based on a more general syntax of message specification. Refer to the MomSys User Guide, Appendix C, Message Specification In MomReceive(), for details of this syntax.

The following information is returned with a message, following a successful call to MomReceive():

- The **message length** is returned as the function's return value.
- A **reply AQid** that was set by the sender program in its call to MomSend() when it sent the message, indicating where a response message should be sent, or `MOM_REPLY_NONE` if no reply AQid was stipulated when the message was sent. This value is returned within the integer variable pointed at by the *RetReplyAQid* argument. Specifying `NULL` causes no value to be returned.
- A **message-ID** that uniquely identifies the message within the local instance is returned within the `MOM_MSGID` variable pointed at by the *RetMsgId* argument. The message-ID can be used to send a reply regarding it via MomSend(). Specifying `NULL` causes no such value to be returned.
- An extensive set of **extended information** about the retrieved message is returned within the `MOMINFOMESSAGE` data structure pointed at by *RetInfoMessage*. The information returned is identical to the data provided by the MomInfoMessage() function. Refer to that function's description for details. Specifying `NULL` causes no such data to be returned.

Blocking options for MomReceive() govern how the function behaves when the specified message is not present on the source app-queue at the time of the MomReceive() call. The following blocking options are available:

MOM_NOWAIT	MomReceive() returns to the calling process *immediately*, either with a message or with a return code indicating that no suitable message was found.
MOM_WAIT	MomReceive() returns to the calling process as soon as the specified message is received. If the application queue does not currently have such a message, the caller blocks until such a message arrives.
MOM_TIMEOUT(*t*)	MomReceive() returns to the calling application as soon as a suitable message is received or the specified time-out of *t* seconds expires, whichever occurs first.
MOM_CALLBACK(*Acb, Func*)	MomReceive() returns to the calling application immediately. The specified callback function is subsequently invoked when a suitable message is received. Warning: The specified *MsgBuf* buffer must be kept intact until the callback function is called. The buffer may, however, be released as part of the callback function's processing.
MOM_POST(*Acb, Sid*)	MomReceive() returns to the calling application immediately. The specified *X♦IPC* semaphore (identified by *Sid*) is subsequently set when a suitable message is received. Warning: The specified *MsgBuf* buffer must be kept intact until the semaphore becomes set.
MOM_IGNORE(*Acb*)	MomReceive() returns to the calling application immediately. *Acb's* completion flag is subsequently set when a suitable message is received. Warning: The specified *MsgBuf* buffer must be kept intact until *Acb's* completion flag becomes set.

Two flags may be specified as part of the call to MomReceive(). This is done by ORing them to the left of the operation's *BlockOpt* argument, as follows: `MomReceive(..., MOM_NOREMOVE | MOM_WAIT).`

Possible flags are:

MOM_NOREMOVE	This flag directs *X♦IPC* that the message should not be removed from the application queue. Rather, a copy of the retrieved message is returned to the calling process.

MOM_RETURN	This flag (which is only valid when accompanying one of the three asynchronous blocking options, MOM_CALLBACK, MOM_POST or MOM_IGNORE) directs *X•IPC* to complete the operation synchronously if there is no need to block (e.g., the desired message is on the app-queue) and to "go asynchronous" only if the operation cannot be completed immediately.

Note that it is possible to interrupt MomReceive() with a signal.

ERRORS

Code	Description
MOM_ER_ASYNC	Operation is being performed asynchronously.
MOM_ER_ASYNCABORT	Asynchronous operation aborted before completion. This error code is not returned by the function call. It is set in the Asynchronous Result Control Block *RetCode* field.
MOM_ER_BADBLOCKOPT	Invalid *BlockOpt*.
MOM_ER_BADBUFFER	*MsgBuf* is NULL.
MOM_ER_BADLENGTH	Invalid *MsgLength* parameter.
MOM_ER_MEMORY	No memory for MomReceive() control blocks.
MOM_ER_NOTSUPPORTED	Function not supported in current version of MomSys.
MOM_ER_BADOPTION	Invalid *Options* parameter.
MOM_ER_BADMSGSPEC	Invalid message specifier.
MOM_ER_INTERNAL	Internal MomSys error (see log file).
MOM_ER_BADAQID	Bad AQid in AQidList (= **AQidPtr*).
MOM_ER_CAPACITY_ASYNC_USER	MomSys async user table full.
MOM_ER_CAPACITY_CS	MomSys communicator server table full.
MOM_ER_CAPACITY_INSTANCE	MomSys instance link table full.
MOM_ER_CAPACITY_MR	MomSys message repository full.
MOM_ER_CAPACITY_REMOTEQUEUE	MomSys remote queue table full.
MOM_ER_DESTROYED	Another user destroyed an application queue that the blocked MomReceive() call was waiting on. The blocked MomReceive() operation was cancelled. No message was received.
MOM_ER_INTERRUPT	Operation was interrupted.
MOM_ER_ISFROZEN	A *BlockOpt* of MOM_WAIT or MOM_TIMEOUT() was specified after the instance was frozen by the calling user.
MOM_ER_NOASYNC	An asynchronous operation was attempted with no asynchronous environment present.
MOM_ER_NOMSG	Specified message not found..
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_NOWAIT	*BlockOpt* of MOM_NOWAIT was specified and request was not immediately satisfied.
MOM_ER_TIMEOUT	The blocked MomReceive() operation timed out.
MOM_ER_TOOBIG	The size of the message exceeds *MsgLength* and MOM_TRUNCATE was not specified.

XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.
XIPCNET_ER_TOOBIG	Message text exceeds instance's size limit.

INTERACTIVE COMMAND

SYNTAX

`momreceive` *AQid MsgSpecifier RetMsgId* `[noremove,] [return,]` *BlockingOpt*

ARGUMENTS

AQid	App-Queue Id of source app-queue

MsgSpecifier	One of the following values:
	`first, last, next(`***`MsgId`***`), prev(`***`MsgId`***`), replyto(`***`MsgId`***`),`
	`direct(`***`MsgId`***`), directrmt(`***`Node`***`, `***`Instance`***`, `***`MsgId`***`)`

RetMsgId	MsgId variable (letter `a` ...`z`) that is set with MsgId data of received message
`[noremove,]`	Optional flag directing API to leave the message on the app-queue
`[return,]`	Optional flag directing *X•IPC* to complete the operation synchronously if there is no need to block. It may be specified only if *BlockingOpt* is asynchronous.
BlockingOpt	See the Blocking Options discussion in the `xipc` Interactive Command Processor section of the *X•IPC* User Guide and Reference Manual.

EXAMPLES

```
xipc>  momreceive 1.0 first a wait
       MsgId: [846266850.256.846266850.256]
       ReplyAQid: NONE
       Text = "hello world"

xipc>  momreceive 1.0 first a noremove,wait
       MsgId: [850146338.1536.850146282.1281]
       ReplyAQid: 1.1
       Text = "some other message text"
```

4.1.17 MomSend() - SEND A MESSAGE

NAME

MomSend() - Send a Message

SYNTAX

```
#include "xipc.h"

XINT
MomSend(
        XINT TargetAQid,
        XANY *MsgBuf,
        XINT MsgLen,
        UXINT Priority,
        XINT TrackingLevel,
        XINT ReplyAQid,
        MOM_MSGID *RetMsgId,
        [... Optional Args ]
        ... BlockOpt)
```

PARAMETERS

Name	Description
TargetAQid	Target app-queue AQid handle
MsgBuf	Pointer to the message buffer
MsgLen	Length of message
Priority	Priority of the message being sent
TrackingLevel	Tracking level of the message being sent
ReplyAQid	The AQid of a response app-queue or, if no response is expected, `MOM_REPLY_NONE`
RetMsgId	Pointer to a variable of type `MOM_MSGID` that is set with the message-id assigned to the sent message, or `NULL` if not desired
[Optional Args]	Zero or more optional arguments to the function for further controlling its behavior (See Description.)
BlockOpt	Blocking option

RETURNS

Value	Description
RC >= 0	Operation successful.

RC < 0 Error (see error codes below).

DESCRIPTION

The MomSend() verb is used to send a message. The sending application identifies the target application queue via the *TargetAQid* argument. The *TargetAQid* handle will have been typically acquired via a prior call to MomCreate() or MomAccess().

MsgBuf and *MsgLen* define the location and length of the message to be sent. *MsgBuf* may be `NULL` for sending a null message (no text), in which case *MsgLen* should be set to zero.

The *Priority* argument provides a means for assigning a level of urgency to the message being sent, relative to other messages moving through the system. Typically, *Priority* is specified as an integer values between 1 and 65535, where: 1 is the "lowest relative urgency;" 65535 is the "highest relative urgency;" and 32767 is defined as "normal urgency." For user convenience, *X•IPC* provides three predefined values that may be passed as the *Priority* argument. They are: `MOM_PRIORITY_HIGHEST`, `MOM_PRIORITY_LOWEST` and `MOM_PRIORITY_NORMAL`.

Beyond these values, the *Priority* argument can be specified in a number of additional ways to provide more granular control over a message's relative urgency. A complete discussion of *Priority* specification alternatives, what they mean and when to use them is provided in the MomSys User Guide, Appendix B, Message Priority Specification. Refer there for details.

TrackingLevel defines the extent to which *X•IPC* will track the sent message. There are two possible values:

`MOM_TRACK_SHIPPED`	Track message until it has been successfully shipped to the target app-queue.
`MOM_TRACK_DELIVERED`	Track message until it has been retrieved, and removed from the target app-queue (i.e., delivered to a receiving program).

If the sending application requires a reply, it should identify a reply app-queue in the *ReplyAQid* argument. Specifying `MOM_REPLY_NONE` as *ReplyAQid* notifies MomSend() that no reply queue is being provided. In such a case, the sending application does not expect any reply from the receiver of the message.

When a message is successfully sent, *X•IPC* creates a unique message identification handle for that message. This message-id is returned to the calling program within the `MOM_MSGID` variable pointed at by the *RetMsgId* argument.

RetMsgId may be used as a message tracking handle by passing it to message tracking verbs such as: MomEvent(), MomStatus(), MomStatusWait() and MomInfoMessage(). Status information regarding the sent message is kept within the local instance's disk-based message repository even after the message has achieved its prescribed tracking-level. By default, message status data is kept indefinitely. (Refer to the discussions on message expiration, message retirement and MR cleaning found in the "Message Repository" section of the MomSys User Guide, for more details on these topics.)

Optional arguments may be specified as part of the call to MomSend(). They are:

MOM_REPLYTO(*MsgId*)	This option directs *X•IPC* to correlate the message being sent as a reply to a previously received message that had a message-id of *MsgId*. (See "Client/Server Programming Techniques" below for details on the usage of this option.)
MOM_EXPIRE(*TimeLimit*)	This option directs *X•IPC* to assign a specific expiration time limit (*TimeLimit)* in seconds for the current message being sent. This value overrides the expiration time value specified in the instance configuration file.

Blocking options for MomSend() govern how the API behaves in dispatching the message when the local *X•IPC* instance has become congested. The following blocking options are available:

MOM_NOWAIT	MomSend() attempts to send the message; if the local *X•IPC* environment is congested, MomSend() fails with an error.
MOM_WAIT	MomSend() attempts to send the message; if the local *X•IPC* environment is congested, MomSend() blocks until congestion is alleviated.
MOM_TIMEOUT(*t*)	MomSend() attempts to send the message; if the local *X•IPC* environment is congested, MomSend() blocks for at most *t* seconds until congestion is alleviated. Otherwise, MomSend() fails.
MOM_CALLBACK(*Acb, Func*)	MomSend() attempts to send the message. The specified callback function is invoked when the requested message is successfully sent: immediately, if there is no congestion; otherwise, later. Warning: The specified *MsgBuf* buffer must be kept intact until the callback function is called. The buffer may, however, be released as part of the callback function's processing.
MOM_POST(*Acb, Sid*)	MomSend() attempts to send the message. The specified *X•IPC* semaphore (identified by *Sid*) is set when the requested message is successfully sent: immediately, if there is no congestion; otherwise, later. Warning: The specified *MsgBuf* buffer must be kept intact until the semaphore becomes set.
MOM_IGNORE(*Acb*)	MomSend() attempts to send the message. *Acb*'s completion flag is set when the message is successfully sent: immediately, if there is no congestion; otherwise, later. Warning: The specified MsgBuf buffer must be kept intact until Acb's completion flag becomes set.

Optional flags may be specified as part of the call to MomSend(). This is done by ORing the flag to the left of the operation's *BlockOpt* argument, as follows: `MomSend(..., MOM_FASTPATH | MOM_WAIT) .`

The two flags are:

`MOM_FASTPATH`	This flag directs *X•IPC* to complete the MomSend() operation (i.e., sending the specified message and returning control to the user) without first synchronizing the message's data to the disk. That is, *X•IPC* performs a "lazy write" of message data when sending this message. This has the advantage of increasing the performance of such MomSend() operations, but has the disadvantage that messages sent with such a flag are not recoverable following a system failure. (Note that the "lazy" update of Message Repository contents occurs at both the sender and receiver instances.) Refer to the MomSys User Guide for certain performance qualifications regarding this flag.
`MOM_RETURN`	This flag (which is only valid when accompanying one of the three asynchronous blocking options, `MOM_CALLBACK`, `MOM_POST` or `MOM_IGNORE`) directs *X•IPC* to complete the operation synchronously if there is no need to block and to "go asynchronous" only if the operation cannot be completed immediately.

Client/Server (Inquiry-Response) Programming Techniques

The following steps summarize the inquiry-response steps that occur during a typical client/server exchange of messages:

1. A client creates a private app-queue (i.e., created with name `MOM_PRIVATE`) within its local instance.
2. The client sends an inquiry message to a server via a call to MomSend() in which it specifies the AQid of its private app-queue as the reply-AQid where it expects to receive a response message.
3. The server receives the inquiry message via a call to MomReceive(); given with it is the message-ID of the received inquiry message, as well as the reply-AQid of the response app-queue (i.e., the AQid of the client's private app-queue).
4. The server processes the message and sends a response message to the client via a call to MomSend() by specifying the client's private app-queue AQid as the target AQid and by specifying the `MOM_REPLYTO(`*`MsgId`*`)` option, where *MsgID* identifies the received inquiry message, so that the response message being sent correlates with the original inquiry message.
5. The client issues a MomReceive() call on its private app-queue to receive the response message to its inquiry message. The MomReceive() call specifies the `MOM_MESSAGE_REPLYTO(`*`MsgId`*`)` message-specifier, where *MsgID* identifies the client's originally sent message, so that it receives the response message sent by the server.

In this manner client-sent messages that arrive at a server may be responded to by the server without any awareness of the location or identification of the originating client. (Refer to the "Client/Server Interaction" section of the MomSys User Guide, for more details and sample programs on these topics.)

ERRORS

Code	Description
MOM_ER_ASYNC	Operation is being performed asynchronously.

MOM_ER_ASYNCABORT	Asynchronous operation aborted before completion. This error code is not returned by the function call. It is set in the Asynchronous Result Control Block *RetCode* field.
MOM_ER_BADBLOCKOPT	Invalid *BlockOpt*.
MOM_ER_BADBUFFER	*MsgBuf* is NULL.
MOM_ER_BADLENGTH	Invalid *MsgLength* parameter.
MOM_ER_BADOPTION	Invalid *Options* parameter.
MOM_ER_NOTSUPPORTED	Option not supported in current version of MomSys.
MOM_ER_BADPRIORITY	*Priority* parameter is 0.
MOM_ER_MEMORY	Insufficient memory for MomSend() control block.
MOM_ER_BADAQID	Bad target AQid.
MOM_ER_BADREPLYAQID	Invalid reply AQid.
MOM_ER_INTERNAL	Internal MomSys() error (see log file).
MOM_ER_CAPACITY_ASYNC_USER	MomSys async user table full.
MOM_ER_CAPACITY_INTERNAL	MomSys internal instance full.
MOM_ER_CAPACITY_MR	MomSys message repository full.
MOM_ER_CAPACITY_REQUEST	MomSys request table full.
MOM_ER_DESTROYED	Another user destroyed an application queue that the blocked MomSend() call was waiting on. The blocked MomSend() operation was cancelled. No message was sent.
MOM_ER_INTERRUPT	Operation was interrupted.
MOM_ER_NOASYNC	An asynchronous operation was attempted with no asynchronous environment present.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_NOWAIT	*BlockOpt* of MOM_NOWAIT was specified and the request was not immediately satisfied.
MOM_ER_TIMEOUT	The blocked MomSend() operation timed out.
MOM_ER_TOOBIG	The size of the message exceeds the maximum configured message size.
XIPCNET_ER_CONNECTLOST	Connection to instance lost.
XIPCNET_ER_NETERR	Network transmission error.
XIPCNET_ER_SYSERR	Operating system error.
XIPCNET_ER_TOOBIG	Message text exceeds instance's size limit.

INTERACTIVE COMMAND

SYNTAX

`momsend` *AQid Message Priority TrackLevel [ReplyAQid] RetMsgId [OptArgs]*
`[fastpath]` *BlockingOpt*

ARGUMENTS

AQid	App-Queue Id of target app-queue
Message	String message being sent

Priority	One of the following values: `highest, normal, lowest`; or an integer between `1` and `32767`
TrackLevel	One of the following values: `held, shipped, delivered`
[ReplyAQid]	*AQid* of local app-queue for reply. If not specified, reply is not expected.
RetMsgId	MsgId variable (letter `a` ...`z`) that is set with MsgId data of sent message
[OptArgs]	Zero or more of the following separated by commas: `replyto(`*msgid_variable*`),` `expire(`*time_in_seconds*`)`
`[fastpath]`	Optionally specify that *X•IPC* should send the message via the **`fastpath`** mechanism.
BlockingOpt	See the Blocking Options discussion in the `xipc` Interactive Command Processor section of the *X•IPC* User Guide and Reference Manual.

EXAMPLES

```
xipc> Wait indefinitely until message reaches remote app-queue.
xipc>  momsend 1.3 “hello world” normal shipped a wait
       MsgId: [846266850.256.846266850.256]

xipc>  # If fastpath message isn't delivered within 1 hr, it expires.
xipc>  momsend 1.0 "hello world" normal delivered a expire(3600)
 fastpath, wait
       MsgId: [846266850.256.846266850.256]


xipc>  # Receive the message, then send correlated response;  note
xipc>  # use of the 'b' msgid in momreceive and then in the
xipc>  # replyto(b) option of momsend.
xipc>  momreceive 2.0 first b wait
       MsgId : [850146338.256,850146282.257]
       ReplyAQid = 2.3
       Text = “Sample inquiry text”
xipc>  momsend 1.0 “Sample response” normal delivered c replyto(b) wait
       MsgId: [846266850.256.846266850.256]
```

4.1.18 MomStatus(), MomStatusWait() - MESSAGE STATUS

NAME

MomStatus(), MomStatusWait() - Message status functions

SYNTAX

```
#include "xipc.h"

XINT
MomStatus (
        XINT MsgId,
        XINT *RetStatus)


XINT
MomStatusWait(
        XINT MsgId,
        XINT Status,
        ... BlockOpt)
```

PARAMETERS

Name	Description
MsgId	Message Id of message being tracked
RetStatus	Pointer to an integer that is returned with the message's latest status
Status	Message status level to wait for
BlockOpt	Blocking option

RETURNS

Value	Description
RC >= 0	Operation successful.
RC < 0	Error (see error codes below).

DESCRIPTION

MomStatus() is used to obtain a message's latest status as known in the caller's local instance. *MsgId* identifies the message to track; *RetStatus* is returned with the discovered status. MomStatusWait() is used to wait until a message reaches a particular status; *MsgId* identifies the message to track; *Status* identifies the status to wait for; *BlockOpt* specifies how to wait. It is possible to interrupt MomStatusWait() with a signal.

Refer to the MomSys User Guide, Appendix A, Message Status and Tracking Levels, for details of possible message status levels.

BlockOpt may be any of the following values:

`MOM_WAIT`	The calling process blocks synchronously until the message attains the specified status.
`MOM_TIMEOUT(`*`t`*`)`	The calling process blocks synchronously for up to *`t`* seconds until the message attains the specified status.
`MOM_NOWAIT`	The calling process synchronously checks the status of the message and returns immediately.
`MOM_CALLBACK(`*`Acb, Func`*`)`	MomStatusWait() returns immediately. The specified callback function is invoked when the message attains the specified status. Warning: The specified *`Acb`* buffer must be kept intact until the callback function is called. The buffer may be released as part of the callback function's processing.
`MOM_POST(`*`Acb, Sid`*`)`	MomStatusWait() returns immediately. The specified *X•IPC* semaphore (identified by *`Sid`*) is set when the message attains the specified status. Warning: The specified semaphore and *`Acb`* must be kept intact until the message attains the specified status.
`MOM_IGNORE(`*`Acb`*`)`	MomStatusWait() returns immediately. *`Acb`*'s completion flag is set when the message attains the specified status. Warning: *`Acb`* must be kept intact until the message attains the specified status.

[**Note**: MomStatus() and MomStatusWait() have been defined upon other MomSys functions; MomStatus() is defined using the MomInfoMessage() API, and MomStatusWait() using the more general MomEvent() API. Refer to the MomSys User Guide, Appendix D, MomStatus() Function Definitions, for details.]

ERRORS

Code	Description
MOM_ER_ASYNC	Operation is being performed asynchronously.
MOM_ER_ASYNCABORT	Asynchronous operation aborted before completion. This error code is not returned by the function call. It is set in the Asynchronous Result Control Block *RetCode* field.
MOM_ER_BADBLOCKOPT	Invalid *BlockOpt*.

MOM_ER_CAPACITY_ASYNC_USER	MomSys async user table full.
MOM_ER_NOTLOGGED IN	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_NOMSG	Specified message not found.
MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.

INTERACTIVE COMMAND

SYNTAX

`momstatus` *MsgId*

`momstatuswait` *MsgId TrackLevel BlockingOpt*

ARGUMENTS

MsgId	MsgId variable (letter 'a' …'z') of message to acquire status on
TrackLevel	One of the following values: `held, shipped, delivered`
BlockingOpt	See the Blocking Options discussion in the `xipc` Interactive Command processor section of the *X•IPC* User Guide and Reference Manual.

EXAMPLES

```
xipc> momsend 1.0 "hello world" delivered a wait
      MsgId: [846266850.256.846266850.256]

xipc> momstatus a
      MsgId: [846266850.256.846266850.256]
      Latest Status: SHIPPED

xipc> momstatuswait a delivered wait
      . . .
      MsgId: [846266850.256.846266850.256]
      Latest Status: DELIVERED
```

5. MOMSYS ERROR CODES

5.1 By Symbolic Error Name

Symbolic Error Name	Number	Description
MOM_ER_ASYNC	-1097	Operation is being performed asynchronously.
MOM_ER_ASYNCABORT	-1098	Asynchronous operation aborted before completion. This error code is not returned by the function call. It is set in the Asynchronous Result Control Block *RetCode* field.
MOM_ER_BADAPPQUENAME	-2503	Invalid *Name* parameter.
MOM_ER_BADAQID	-2500	Bad AQid in AQidList (= **AQidPtr*).
MOM_ER_BADATTRBLOCK	-2514	Attribute block not initialized correctly.
MOM_ER_BADBLOCKOPT	-1031	Invalid *BlockOpt*.
MOM_ER_BADBUFFER	-2507	Return buffer pointer is NULL.
MOM_ER_BADBUFFER	-2507	*MsgBuf* is NULL.
MOM_ER_BADCOMMANDOPTION	-2564	Bad value for command line program specified.
MOM_ER_BADLENGTH	-2506	Invalid *MsgLength* parameter.
MOM_ER_BADLINKID	-2530	No link with *LinkId*.
MOM_ER_BADMSGSPEC	-2510	Invalid message specifier.
MOM_ER_BADOPTION	-1030	Invalid *Options* parameter.
MOM_ER_BADPRIORITY	-2526	*Priority* parameter is 0.
MOM_ER_BADREPLYAQID	-2535	Invalid reply AQid.
MOM_ER_BADUID	-1023	No user with specified *Uid*.
MOM_ER_BADVAL	-1024	Cursor pointer is NULL.
MOM_ER_CAPACITY_ASYNC_USER	-2546	MomSys async user table full.
MOM_ER_CAPACITY_ASYNC_USER	-2546	MomSys async user table full.
MOM_ER_CAPACITY_CS	-2544	MomSys communicator server table full.
MOM_ER_CAPACITY_INSTANCE	-2545	MomSys instance link table full.
MOM_ER_CAPACITY_INTERNAL	-2548	MomSys internal instance full.
MOM_ER_CAPACITY_LOCALQUEUE	-2542	MomSys local queue table full.
MOM_ER_CAPACITY_MR	-2547	MomSys message repository full.
MOM_ER_CAPACITY_REMOTEQUEUE	-2543	MomSys remote queue table full.
MOM_ER_CAPACITY_REQUEST	-2549	MomSys request table full.
MOM_ER_CAPACITY_USER	-2541	MomSys user table full.
MOM_ER_CATERROR	-2559	Error in catalog server processing request.
MOM_ER_CATUNREACHABLE	-2560	All catalog servers unreachable.
MOM_ER_DESTROYED	-1035	Another user destroyed an application queue that the blocked MomReceive() call was waiting on. The blocked MomReceive() operation was cancelled. No message was received.

Symbolic Error Name	Number	Description
MOM_ER_DUPLICATE	-1032	Application queue with *Name* already exists.
MOM_ER_INTERNAL	-2517	MomSys internal error (refer to log file).
MOM_ER_INTERRUPT	-1100	Operation was interrupted.
MOM_ER_ISFROZEN	-1007	A *BlockOpt* of MOM_WAIT or MOM_TIMEOUT() was specified after the instance was frozen by the calling user.
MOM_ER_MEMORY	-1121	No memory for MomReceive() control blocks.
MOM_ER_MSGQDESTROYED	-2571	The queue with the message pending is destroyed.
MOM_ER_NOASYNC	-1006	An asynchronous operation was attempted with no asynchronous environment present.
MOM_ER_NOMESSAGE	-2512	Specified message not found..
MOM_ER_NOMORE	-1038	No more data.
MOM_ER_NOMSG	-2512	Specified message not found.
MOM_ER_NONETWORK	-2518	Remote queue access cannot be performed since the instance is not configured for network connection.
MOM_ER_NOSUBSYSTEM	-1004	MomSys is not configured in the instance.
MOM_ER_NOTEMPTY	-2504	The application queue is not empty.
MOM_ER_NOTFOUND	-1033	Application queue with *Name* does not exist.
MOM_ER_NOTLOGGEDIN	-1002	User not logged into instance (User never logged in, was aborted or disconnected).
MOM_ER_NOTSUPPORTED	-1040	Function not supported in current version of MomSys.
MOM_ER_NOWAIT	-1034	*BlockOpt* of MOM_NOWAIT was specified and request was not immediately satisfied.
MOM_ER_REMOTE_QUEUE	-2501	*AQid* is of a remote app-queue.
MOM_ER_TIMEOUT	-1099	The blocked MomReceive() operation timed out.
MOM_ER_TOOBIG	-2508	The size of the message exceeds *MsgLength* and MOM_TRUNCATE was not specified.
MOM_ER_WAITEDON	-2505	A user is waiting for a message on *AQid*.

5.2 By Message Number

NUMBER	SYMBOLIC ERROR NAME	DESCRIPTION
-1002	MOM_ER_NOTLOGGEDIN	User not logged into instance (User never logged in, was aborted or disconnected).
-1004	MOM_ER_NOSUBSYSTEM	MomSys is not configured in the instance.
-1006	MOM_ER_NOASYNC	An asynchronous operation was attempted with no asynchronous environment present.
-1007	MOM_ER_ISFROZEN	A *BlockOpt* of MOM_WAIT or MOM_TIMEOUT() was specified after the instance was frozen by the calling user.
-1023	MOM_ER_BADUID	No user with specified *Uid*.
-1024	MOM_ER_BADVAL	Cursor pointer is NULL.
-1030	MOM_ER_BADOPTION	Invalid *Options* parameter.
-1031	MOM_ER_BADBLOCKOPT	Invalid *BlockOpt*.
-1032	MOM_ER_DUPLICATE	Application queue with *Name* already exists.
-1033	MOM_ER_NOTFOUND	Application queue with *Name* does not exist.
-1034	MOM_ER_NOWAIT	*BlockOpt* of MOM_NOWAIT was specified and request was not immediately satisfied.
-1035	MOM_ER_DESTROYED	Another user destroyed an application queue that the blocked MomReceive() call was waiting on. The blocked MomReceive() operation was cancelled. No message was received.
-1038	MOM_ER_NOMORE	No more data.
-1040	MOM_ER_NOTSUPPORTED	Function not supported in current version of MomSys.
-1097	MOM_ER_ASYNC	Operation is being performed asynchronously.
-1098	MOM_ER_ASYNCABORT	Asynchronous operation aborted before completion. This error code is not returned by the function call. It is set in the Asynchronous Result Control Block *RetCode* field.
-1099	MOM_ER_TIMEOUT	The blocked MomReceive() operation timed out.
-1100	MOM_ER_INTERRUPT	Operation was interrupted.
-1121	MOM_ER_MEMORY	No memory for MomReceive() control blocks.
-2500	MOM_ER_BADAQID	Bad AQid in AQidList (= **AQidPtr*).
-2501	MOM_ER_REMOTE_QUEUE	*AQid* is of a remote app-queue.
-2503	MOM_ER_BADAPPQUENAME	Invalid *Name* parameter.
-2504	MOM_ER_NOTEMPTY	The application queue is not empty.
-2505	MOM_ER_WAITEDON	A user is waiting for a message on *AQid*.
-2506	MOM_ER_BADLENGTH	Invalid *MsgLength* parameter.
-2507	MOM_ER_BADBUFFER	*MsgBuf* is NULL.
-2507	MOM_ER_BADBUFFER	Return buffer pointer is NULL.
-2508	MOM_ER_TOOBIG	The size of the message exceeds *MsgLength* and MOM_TRUNCATE was not specified.

NUMBER	SYMBOLIC ERROR NAME	DESCRIPTION
-2510	MOM_ER_BADMSGSPEC	Invalid message specifier.
-2512	MOM_ER_NOMESSAGE	Specified message not found..
-2512	MOM_ER_NOMSG	Specified message not found.
-2514	MOM_ER_BADATTRBLOCK	Attribute block not initialized correctly.
-2517	MOM_ER_INTERNAL	MomSys internal error (refer to log file).
-2518	MOM_ER_NONETWORK	Remote queue access cannot be performed since the instance is not configured for network connection.
-2526	MOM_ER_BADPRIORITY	*Priority* parameter is 0.
-2530	MOM_ER_BADLINKID	No link with *LinkId*.
-2535	MOM_ER_BADREPLYAQID	Invalid reply AQid.
-2541	MOM_ER_CAPACITY_USER	MomSys user table full.
-2542	MOM_ER_CAPACITY_LOCALQUEUE	MomSys local queue table full.
-2543	MOM_ER_CAPACITY_REMOTEQUEUE	MomSys remote queue table full.
-2544	MOM_ER_CAPACITY_CS	MomSys communicator server table full.
-2545	MOM_ER_CAPACITY_INSTANCE	MomSys instance link table full.
-2546	MOM_ER_CAPACITY_ASYNC_USER	MomSys async user table full.
-2546	MOM_ER_CAPACITY_ASYNC_USER	MomSys async user table full.
-2547	MOM_ER_CAPACITY_MR	MomSys message repository full.
-2548	MOM_ER_CAPACITY_INTERNAL	MomSys internal instance full.
-2549	MOM_ER_CAPACITY_REQUEST	MomSys request table full.
-2559	MOM_ER_CATERROR	Error in catalog server processing request.
-2560	MOM_ER_CATUNREACHABLE	All catalog servers unreachable.
-2564	MOM_ER_BADCOMMANDOPTION	Bad value for command line program specified.
-2571	MOM_ER_MSGQDESTROYED	The queue with the message pending is destroyed.