

EventMgr Objects

Introduction

An **EventMgr** object manages the collection and distribution of event messages from the controller to the host. Events include normal motion completion, motor limits, fault conditions, data recorder buffer full, network node faults, etc. The EventNotify methods enable the application to request host notification for specified events, while ignoring other events. The EventMgr receives the event and then distributes the event via the Notify object to the tasks that are waiting for the event. Events that are faults are latched; the fault condition and the event must be cleared before the event can be generated again.

To collect events, the EventMgr must be serviced via the [mpiEventMgrService\(...\)](#) routine. The EventMgr can be polled periodically by calling the mpiEventMgrService(...) routine or can be placed in an interrupt service routine. For your convenience, there is an apputil module that provides a serviceCreate(...) function that creates an EventMgr service routine for Windows OSs. The Service thread can be configured for polling or interrupts. Sources for the apputil module are included with the software distribution, so you can port it to other OSs.

The controller has a circular buffer which holds up to 128 event messages. If the EventMgr is not serviced within 128 events, event messages will be overwritten as new events occur. For interrupt driven EventMgr service threads, this is not an issue. For a polling EventMgr service thread, this could be an issue if the service thread does not have enough CPU cycles. In the worst-case scenario, events could be lost.

The best way to avoid lost events is to use an interrupt driven EventMgr service routine. If you're using polling, the next best thing is to make sure the EventMgr services the events at a high enough rate to avoid controller message buffer rollover. Determining the worst case EventMgr service latency and the maximum event message rate can be tricky. A simple method is to estimate the event message frequency and make sure each EventMgr can poll at least once for every 32 events (safety factor of 4). If you know that some events occur more frequently than others, then you may want to increase the polling frequency for EventMgrs that process the most frequent events and decrease the polling frequency for EventMgrs that process less frequently.

Generally, use only one EventMgr in your application. Do not use multiple interrupt driven EventMgr service threads to collect the same event. Suppose you configure an EventMgr to service Motion Done events from MS 0 and you configure another EventMgr with MEIEventMgrServiceConfig.allProcesses = TRUE. If one EventMgr collects the event message and acknowledges the interrupt before the other EventMgr can respond, it will miss the event.

If you want to monitor ALL events with MEIEventMgrServiceConfig.allProcesses =

TRUE, then use polling (not interrupts). See the sample app EventLog.c for an example.

| [Error Messages](#) |

Methods

Create, Delete, Validate Methods

mpiEventMgrCreate	Create EventMgr object
mpiEventMgrDelete	Delete EventMgr object
mpiEventMgrValidate	Validate EventMgr object

Configuration and Information Methods

mpiEventMgrConfigGet	Get EventMgr config
mpiEventMgrConfigSet	Set EventMgr config
mpiEventMgrEvent	Request event notification for all Notify objects on EventMgr's list
meiEventMgrServiceConfigGet	Get processes that EventMgr will service
meiEventMgrServiceConfigSet	Set processes that EventMgr will service

Action Methods

mpiEventMgrFlush	Flush pending EventMgr events
mpiEventMgrService	Get list of all pending asynchronous events

Relational Methods

List Methods- for Control Objects

mpiEventMgrControl	Return handle of indexth Control object in list
mpiEventMgrControlAppend	Append Control's handle to list
mpiEventMgrControlCount	Count the number of Control objects associated with EventMgr (in list)
mpiEventMgrControlFirst	Return handle to first Control object in list
mpiEventMgrControlIndex	Return the index of a Control object in list
mpiEventMgrControlInsert	Insert Control handle into list
mpiEventMgrControlLast	Get handle to last Control object in list
mpiEventMgrControlListGet	Get list of Control objects associated with EventMgr
mpiEventMgrControlListSet	Create a list of Control objects associated with EventMgr
mpiEventMgrControlNext	Get handle to next Control object in list
mpiEventMgrControlPrevious	Get handle to previous Control object in list
mpiEventMgrControlRemove	Remove a Control object's handle from list

List Methods- for Notify Objects

mpiEventMgrNotify	Return handle to a Notify object associated with EventMgr
mpiEventMgrNotifyAppend	Append Notify object to list
mpiEventMgrNotifyCount	Return number of Notify objects in list
mpiEventMgrNotifyFirst	Get first Notify object in list

[mpiEventMgrNotifyIndex](#)

Get index value for a Notify object in list

[mpiEventMgrNotifyInsert](#)

Place a Notify object after another Notify object in list

[mpiEventMgrNotifyLast](#)

Get handle to the Notify object that is last on the list

[mpiEventMgrNotifyListGet](#)

Get a list of Notify objects

[mpiEventMgrNotifyListSet](#)

Create a list of Notify objects

[mpiEventMgrNotifyNext](#)

Get the Notify object just after notify in list

[mpiEventMgrNotifyPrevious](#)

Get the Notify object just before notify in list

[mpiEventMgrNotifyRemove](#)

Remove a Notify object from list

Data Types

[MPIEventMgrConfig](#)

[MPIEventMgrMessage](#)

[MEIEventMgrServiceConfig](#)

mpiEventMgrService

Declaration

```
long mpiEventMgrService(MPIEventMgr eventMgr ,
                        MPIControl control )
```

Required Header: stdmpi.h

Description

mpiEventMgrService obtains all pending asynchronous events from the motion controller (*control*).

Events generated by enabled sources are returned in First-In/First-Out (FIFO) order each time a thread calls mpiEventMgrService(...).

Typically, after a motion controller generates a hardware interrupt, the Interrupt Service Routine (ISR) is invoked, and the ISR in turn invokes mpiEventMgrService(...) directly or indirectly.

<i>If "control" is</i>	Then
MPIHandleVOID	events will be obtained from all motion controllers on the EventMgr's (<i>eventMgr</i>) control list
valid	<i>control</i> must also be present on the EventMgr's (<i>eventMgr</i>) control list

Return Values

[MPIMessageOK](#)

See Also

[meiEventMgrServiceConfigGet](#) | [meiEventMgrServiceConfigSet](#) | [mpiEventMgrFlush](#)

meiEventMgrServiceConfigGet

Declaration

```
long meiEventMgrServiceConfigGet(MPIEventMgr eventMgr,  
                                 MEIEventMgrServiceConfig *config)
```

Required Header: stdmpi.h

Description

meiEventMgrServiceConfigGet gets the configuration of an *EventMgr* object (eventMgr) and writes it into the structure pointed to by *config*.

Return Values

[MPIMessageOK](#)

See Also

[meiEventMgrServiceConfigSet](#) | [mpiEventMgrService](#)

meiEventMgrServiceConfigSet

Declaration

```
long meiEventMgrServiceConfigSet(MPIEventMgr eventMgr,  
                                 MEIEventMgrServiceConfig *config)
```

Required Header: stdmpi.h

Description

meiEventMgrServiceConfigSet sets (writes) the flash configuration for an EventMgr object (*eventMgr*) using data from the structure pointed to by *config*.

Return Values

[MPIMessageOK](#)

See Also

[meiEventMgrServiceConfigGet](#) | [mpiEventMgrService](#)

mpiEventMgrFlush

Declaration

```
long mpiEventMgrFlush(MPIEventMgr eventMgr)
```

Required Header: stdmpi.h

Description

mpiEventMgrFlush flushes any pending events from an EventMgr (*eventMgr*).

Return Values

[MPIMessageOK](#)

See Also

mpiEventMgrCreate

Declaration

```
MPIEventMgr mpiEventMgrCreate(MPIControl control)
```

Required Header: stdmpi.h

Description

mpiEventMgrCreate creates an EventMgr object, with ***control*** as the initial element in the list of Control objects from which the EventMgr obtains asynchronous events (***control*** may be MPIHandleVOID).

EventMgrCreate is the equivalent of a C++ constructor.

Return Values

handle	to an EventMgr object
MPIHandleVOID	if the object could not be created

See Also

[mpiEventMgrDelete](#) | [mpiEventMgrValidate](#)

mpiEventMgrDelete

Declaration

```
long mpiEventMgrDelete(MPIEventMgr eventMgr)
```

Required Header: stdmpi.h

Description

mpiEventMgrDelete deletes an EventMgr object and invalidates its handle (*eventMgr*). EventMgrDelete is the equivalent of a C++ destructor.

Deleting an EventMgr object does not delete any of the Control objects that supply the EventMgr with asynchronous events. However, deleting an EventMgr object will delete any unreceived events for that EventMgr.

Return Values

[MPIMessageOK](#)

See Also

[mpiEventMgrCreate](#) | [mpiEventMgrValidate](#)

mpiEventMgrValidate

Declaration

```
long mpiEventMgrValidate(MPIEventMgr eventMgr )
```

Required Header: stdmpi.h

Description

mpiEventMgrValidate validates an EventMgr object and its handle (*eventMgr*).

Return Values

[MPIMessageOK](#)

See Also

[mpiEventMgrCreate](#) | [mpiEventMgrDelete](#)

mpiEventMgrConfigGet

Declaration

```
long mpiEventMgrConfigGet(MPIEventMgr      eventMgr ,
                          MPIEventMgrConfig *config ,
                          void                *external )
```

Required Header: stdmpi.h

Description

mpiEventMgrConfigGet gets the configuration of an *EventMgr* object (eventMgr) and writes it into the structure pointed to by *config*, and also writes it into the implementation-specific structure pointed to by *external* (if *external* is not NULL).

The configuration information in *external* is in addition to the configuration information in *config*, i.e, the configuration information in *config* and in *external* is not the same information. Note that *config* or *external* can be NULL (but not both NULL).

Remarks

external either points to a structure of type **MEIEventMgrConfig{}** or is NULL.

Return Values

[MPIMessageOK](#)

See Also

[mpiEventMgrConfigSet](#)

mpiEventMgrConfigSet

Declaration

```
long mpiEventMgrConfigSet(MPIEventMgr      eventMgr ,  
                          MPIEventMgrConfig *config ,  
                          void                *external )
```

Required Header: stdmpi.h

Description

mpiEventMgrConfigSet sets (writes) the *eventMgr* configuration using data from the structure pointed to by *config*, and also using data from the implementation-specific structure pointed to by *external* (if *external* is not NULL).

The configuration information in *external* is in addition to the configuration information in *config*, i.e, the configuration information in *config* and in *external* is not the same information. Note that *config* or *external* can be NULL (but not both NULL).

Remarks

external either points to a structure of type **MEIEventMgrConfig{}** or is NULL.

Return Values

[MPIMessageOK](#)

See Also

[mpiEventMgrConfigGet](#)

mpiEventMgrEvent

Declaration

```
long mpiEventMgrEvent(MPIEventMgr eventMgr ,  
                     MPIEventStatus *status )
```

Required Header: stdmpi.h

Description

mpiEventMgrEvent requests that the EventMgr (*eventMgr*) call `mpiNotifyEvent(notify, status)` for all Notify objects on that Event Manager's list. Each Notify object represents a thread that has requested event notification. The Notify object will determine whether it accepts the event notification or not.

EventMgrEvent enables your application to use the Event Manager to distribute *user-created events* in the same way that events generated by the motion controller are distributed.

Return Values

[MPIMessageOK](#)

See Also

mpiEventMgrControl

Declaration

```
MPIControl mpiEventMgrControl(MPIEventMgr eventMgr ,
                               long index)
```

Required Header: stdmpi.h

Description

mpiEventMgrControl returns the element at the position on the list indicated by *index*.

eventMgr	a handle to the EventMgr object.
index	a position in the list.

Return Values

[MPIMessageOK](#)

handle

to the *index*th motion controller (Control) associated with an EventMgr (*eventMgr*)

MPIHandleVOID

if *eventMgr* is invalid
if *index* is less than 0
if *index* is greater than or equal to **mpiEventMgrCount(eventMgr)**

[MPIMessageARG_INVALID](#)

[MEIListMessageELEMENT_NOT_FOUND](#)

[MPIMessageHANDLE_INVALID](#)

See Also

mpiEventMgrControlAppend

Declaration

```
long mpiEventMgrControlAppend(MPIEventMgr eventMgr ,
                               MPIControl control )
```

Required Header: stdmpi.h

Description

mpiEventMgrControlAppend appends *control* to the list of motion controllers associated with an EventMgr (*eventMgr*). Add "control" to the end of the list.

eventMgr	a handle to the EventMgr object.
control	a handle to a Control object.

Return Values

[MPIMessageOK](#)

[MPIMessageHANDLE_INVALID](#)

[MPIMessageOBJECT_ON_LIST](#)

[MPIMessageNO_MEMORY](#)

See Also

mpiEventMgrControlCount

Declaration

```
long mpiEventMgrControlCount (MPIEventMgr eventMgr)
```

Required Header: stdmpi.h

Description

mpiEventMgrControlCount returns the number of elements on the list.

eventMgr	a handle to the EventMgr object.
-----------------	----------------------------------

Return Values

number	of motion controllers associated with an EventMgr (<i>eventMgr</i>)
-1	if <i>eventMgr</i> is an invalid handle
0	if <i>eventMgr</i> has no associated motion controllers

See Also

mpiEventMgrControlFirst

Declaration

```
MPIControl mpiEventMgrControlFirst(MPIEventMgr eventMgr)
```

Required Header: stdmpi.h

Description

mpiEventMgrControlFirst returns the first element in the list. This function can be used in conjunction with [mpiEventMgrControlNext\(...\)](#) in order to iterate through the list. MPIHandleVOID is returned if the list is empty.

eventMgr	a handle to the EventMgr object.
-----------------	----------------------------------

Return Values

handle	to the first motion controller (Control) associated with an EventMgr (<i>eventMgr</i>)
MPIHandleVOID	if <i>eventMgr</i> is invalid if <i>eventMgr</i> has no associated motion controllers
MPIMessageHANDLE_INVALID	

See Also

[mpiEventMgrControlLast](#)

mpiEventMgrControlNext

Declaration

```
MPIControl mpiEventMgrControlNext(MPIEventMgr eventMgr ,  
                                  MPIControl control )
```

Required Header: stdmpi.h

Description

mpiEventMgrControlNext returns the next element following "control" on the list. MPIHandleVOID is returned if "control" is the last element on the list, or if "control" is not in the list at all. This function can be used in conjunction with mpiEventMgrControlFirst() in order to iterate through the list.

eventMgr	a handle to the EventMgr object.
control	a handle to a Control object.

Return Values

handle	to the motion controller following <i>control</i> in the list of motion controllers associated with an EventMgr (<i>eventMgr</i>)
MPIHandleVOID	if <i>eventMgr</i> is invalid if <i>control</i> is the last motion controller
MPIMessageHANDLE_INVALID	

See Also

[mpiEventMgrControlPrevious](#)

mpiEventMgrControlLast

Declaration

```
MPIControl mpiEventMgrControlLast(MPIEventMgr eventMgr)
```

Required Header: stdmpi.h

Description

mpiEventMgrControlLast returns the last element in the list.

This function can be used in conjunction with `mpiEventMgrControlPrevious()` in order to iterate through the list backwards.

eventMgr	a handle to the EventMgr object.
-----------------	----------------------------------

Return Values

handle	to the last motion controller (Control) associated with an EventMgr (<i>eventMgr</i>)
---------------	---

MPIHandleVOID	if <i>eventMgr</i> is invalid if <i>eventMgr</i> has no associated motion controllers
----------------------	--

MPIMessageHANDLE_INVALID	
--	--

See Also

[mpiEventMgrControlFirst](#)

mpiEventMgrControlIndex

Declaration

```
long mpiEventMgrControlIndex(MPIEventMgr eventMgr ,  
                             MPIControl control )
```

Required Header: stdmpi.h

Description

mpiEventMgrControlIndex returns the position of "control" on the list.

eventMgr	a handle to the EventMgr object.
control	a handle to a Control object.

Return Values

index	of <i>control</i> in the list of motion controllers associated with an EventMgr (<i>eventMgr</i>)
-1	if <i>eventMgr</i> is invalid if <i>control</i> was not found in the list of motion controllers

See Also

mpiEventMgrControlInsert

Declaration

```
long mpiEventMgrControlInsert(MPIEventMgr eventMgr ,  
                               MPIControl control ,  
                               MPIControl insert )
```

Required Header: stdmpi.h

Description

mpiEventMgrControlInsert inserts a Control object (*insert*) after the Control object (*control*) in the list of motion controllers associated with *eventMgr*.

eventMgr	a handle to the EventMgr object.
-----------------	----------------------------------

Return Values

MPIMessageOK	if <i>EventMgrControlInsert</i> successfully inserts the Control object (<i>insert</i>) after another Control object (<i>control</i>) in the list of motion controllers
---------------------	---

See Also

mpiEventMgrControlListGet

Declaration

```
long mpiEventMgrControlListGet(MPIEventMgr eventMgr ,  
                               long          *controlCount ,  
                               MPIControl  *controlList )
```

Required Header: stdmpi.h

Description

mpiEventMgrControlListGet returns the list of Control objects associated with an EventMgr object (*eventMgr*).

EventMgrControlListGet also writes the number of Control handles (in the list) to the location pointed to by *controlCount*, and writes an array (of *controlCount* Control handles) to the location pointed to by *controlList*.

eventMgr	a handle to the EventMgr object.
-----------------	----------------------------------

Return Values

[MPIMessageOK](#)

See Also

[mpiEventMgrControlListSet](#)

mpiEventMgrControlListSet

Declaration

```
long mpiEventMgrControlListSet(MPIEventMgr eventMgr ,  
                                long          controlCount ,  
                                MPIControl   *controlList )
```

Required Header: stdmpi.h

Description

mpiEventMgrControlListSet creates a list of **controlCount** Control objects using the Control handles specified by **controlList**. Any existing motion controller list is completely replaced.

The **controlList** parameter is the address of an array of **controlCount** Control handles, or is NULL (if **controlCount** is equal to zero).

A motion controller list can also be created incrementally (i.e., one Control at a time) by using the append and/or insert methods described in this section. The initial Control of a control list may be specified using the **control** parameter of [mpiEventMgrCreate\(...\)](#). The list methods in this section can be used to examine and manipulate a motion controller list regardless of how the list was created.

Return Values

[MPIMessageOK](#)

See Also

[mpiEventMgrControlListGet](#)

mpiEventMgrControlPrevious

Declaration

```
MPIControl mpiEventMgrControlPrevious(MPIEventMgr eventMgr ,
                                       MPIControl control)
```

Required Header: stdmpi.h

Description

mpiEventMgrControlPrevious returns the previous element prior to "control" on the list. MPIHandleVOID is returned if "control" is the first element on the list, or if "control" is not in the list at all. This function can be used in conjunction with [mpiEventMgrControlLast\(...\)](#) in order to iterate through the list backwards.

eventMgr	a handle to the EventMgr object.
control	a handle to a Control object.

Return Values

handle	to the motion controller preceding <i>control</i> in the list of motion controllers associated with an EventMgr (<i>eventMgr</i>)
MPIHandleVOID	if <i>eventMgr</i> is invalid if <i>control</i> is the first motion controller
MPIMessageHANDLE_INVALID	

See Also

[mpiEventMgrControlNext](#)

mpiEventMgrControlRemove

Declaration

```
long mpiEventMgrControlRemove(MPIEventMgr eventMgr ,
                               MPIControl control )
```

Required Header: stdmpi.h

Description

mpiEventMgrControlRemove removes a Control object (*control*) from the list of Control objects associated with *eventMgr*.

eventMgr	a handle to the EventMgr object.
control	a handle to a Control object.

Return Values

MPIMessageOK	if <i>EventMgrControlRemove</i> successfully removes a Control object from the list of Control objects associated with an EventMgr (<i>eventMgr</i>)
---------------------	--

See Also

[mpiEventMgrControl](#)

mpiEventMgrNotify

Declaration

```
MPINotify mpiEventMgrNotify(MPIEventMgr eventMgr ,
                             long index )
```

Required Header: stdmpi.h

Description

mpiEventMgrNotify returns the element at the position on the list indicated by *index*.

eventMgr	a handle to the EventMgr object.
index	a position in the list.

Return Values

handle	to the <i>index</i> th Notify object associated with an EventMgr (<i>eventMgr</i>)
MPIHandleVOID	if <i>eventMgr</i> is invalid if <i>index</i> is less than 0 if <i>index</i> is greater than or equal to mpiEventMgrCount(eventMgr)
MPIMessageARG_INVALID	
MEIListMessageELEMENT_NOT_FOUND	
MPIMessageHANDLE_INVALID	

See Also

mpiEventMgrNotifyAppend

Declaration

```
long mpiEventMgrNotifyAppend(MPIEventMgr eventMgr ,
                             MPINotify notify)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyAppend appends a Notify object (*notify*) to the list of Notify objects maintained by an EventMgr object (*eventMgr*).

eventMgr	a handle to the EventMgr object.
notify	a handle to a Notify object.

Return Values

[MPIMessageOK](#)

[MPIMessageHANDLE_INVALID](#)

[MPIMessageOBJECT_ON_LIST](#)

[MPIMessageNO_MEMORY](#)

See Also

mpiEventMgrNotifyCount

Declaration

```
long mpiEventMgrNotifyCount(MPIEventMgr eventMgr)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyCount returns the number of elements on the list.

eventMgr	a handle to the EventMgr object.
-----------------	----------------------------------

Return Values

number	of Notify objects in the list (of Notify objects) maintained by an EventMgr (<i>eventMgr</i>)
-1	if <i>eventMgr</i> is invalid
0	if the list (of Notify objects) is empty

See Also

mpiEventMgrNotifyFirst

Declaration

```
MPINotify mpiEventMgrNotifyFirst(MPIEventMgr eventMgr)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyFirst returns the first element in the list. This function can be used in conjunction with [mpiEventMgrNotifyNext\(...\)](#) in order to iterate through the list.

eventMgr	a handle to the EventMgr object.
-----------------	----------------------------------

Return Values

handle	to the first Notify object in the list (of Notify objects) maintained by an EventMgr (<i>eventMgr</i>)
MPIHandleVOID	if <i>eventMgr</i> is invalid if the list (of Notify objects) is empty
MPIMessageHANDLE_INVALID	

See Also

[mpiEventMgrNotifyLast](#) | [mpiEventMgrNotifyNext](#)

mpiEventMgrNotifyNext

Declaration

```
MPINotify mpiEventMgrNotifyNext(MPIEventMgr eventMgr ,
                                MPINotify notify)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyNext returns the next element following **notify** on the list. This function can be used in conjunction with [mpiEventMgrNotifyFirst\(...\)](#) in order to iterate through the list.

eventMgr	a handle to the EventMgr object.
notify	a handle to a Notify object.

Return Values

handle	to the first Notify object in the list (of Notify objects) maintained by an EventMgr (<i>eventMgr</i>)
MPIHandleVOID	if <i>eventMgr</i> is invalid if the list (of Notify objects) is empty
MPIMessageHANDLE_INVALID	

See Also

[mpiEventMgrNotifyPrevious](#) | [mpiEventMgrNotifyFirst](#)

mpiEventMgrNotifyLast

Declaration

```
MPINotify mpiEventMgrNotifyLast(MPIEventMgr eventMgr)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyLast returns the last element in the list. This function can be used in conjunction with [mpiEventMgrNotifyPrevious\(...\)](#) in order to iterate through the list backwards.

eventMgr	a handle to the EventMgr object.
-----------------	----------------------------------

Return Values

handle	to the first Notify object in the list (of Notify objects) maintained by an EventMgr (<i>eventMgr</i>)
MPIHandleVOID	if <i>eventMgr</i> is invalid if the list (of Notify objects) is empty
MPIMessageHANDLE_INVALID	

See Also

[mpiEventMgrNotifyFirst](#) | [mpiEventMgrNotifyPrevious](#)

mpiEventMgrNotifyIndex

Declaration

```
long mpiEventMgrNotifyIndex(MPIEventMgr eventMgr ,
                            MPINotify notify)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyIndex returns the position of *notify* on the list.

eventMgr	a handle to the EventMgr object.
notify	a handle to the EventMgr object.

Return Values

index	of <i>notify</i> in the list (of Notify objects) maintained by an EventMgr (<i>eventMgr</i>)
-1	if eventMgr is invalid if <i>notify</i> was not found in the list

See Also

mpiEventMgrNotifyInsert

Declaration

```
long mpiEventMgrNotifyInsert (MPIEventMgr eventMgr ,  
                              MPINotify   notify ,  
                              MPINotify   insert )
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyInsert places a Notify object (*insert*) after another Notify object (*notify*) in the list (of Notify objects) maintained by an EventMgr object (*eventMgr*).

Return Values

[MPIMessageOK](#)

See Also

mpiEventMgrNotifyPrevious

Declaration

```
MPINotify mpiEventMgrNotifyPrevious(MPIEventMgr eventMgr ,
                                     MPINotify notify)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyPrevious returns the previous element prior to **notify** on the list. This function can be used in conjunction with `mpiEventMgrNotifyLast(...)` in order to iterate through the list backwards.

eventMgr	a handle to the EventMgr object.
notify	a handle to a Notify object.

Return Values	
handle	to the Notify object just before another Notify object (notify) in the list (of Notify objects) maintained by an EventMgr (eventMgr)
MPIHandleVOID	if eventMgr is invalid if notify is the first Notify object in the list
MPIMessageHANDLE_INVALID	either eventMgr or notify is an invalid handle

See Also

[mpiEventMgrNotifyNext](#) | [mpiEventMgrNotifyLast](#)

mpiEventMgrNotifyListGet

Declaration

```
long mpiEventMgrNotifyListGet(MPIEventMgr eventMgr ,  
                               long          *notifyCount ,  
                               MPINotify   *notifyList )
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyListGet returns the list of Notify objects for an EventMgr object (*eventMgr*). *EventMgrNotifyListGet* also sets (writes) the number (of Notify objects in the list) to the location pointed to by *notifyCount*, and sets (writes) an array (of *notifyCount* Notify handles) to the contents of the location pointed to by *notifyList*.

Return Values

[MPIMessageOK](#)

See Also

[mpiEventMgrNotifyListSet](#)

mpiEventMgrNotifyListSet

Declaration

```
long mpiEventMgrNotifyListSet(MPIEventMgr eventMgr ,  
                               long notifyCount ,  
                               MPINotify *notifyList)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyListSet creates a list of Notify objects, where the number of Notify objects is specified by **notifyCount**, using the Notify handles specified by **notifyList**. The **notifyList** argument is the address of an array of **notifyCount** Notify handles, or is NULL if **notifyCount** is zero. Any existing notify object list is completely replaced.

You can also create a Notify object list incrementally (i.e., one Notify object is created at a time), by using the EventMgrAppend and/or EventMgrInsert methods. After creating a list of Notify objects, use the EventMgrList methods to examine and manipulate the list, regardless of how the list was created.

Return Values

[MPIMessageOK](#)

See Also

[mpiEventMgrNotifyListGet](#)

mpiEventMgrNotifyRemove

Declaration

```
long mpiEventMgrNotifyRemove(MPIEventMgr eventMgr ,  
                             MPINotify notify)
```

Required Header: stdmpi.h

Description

mpiEventMgrNotifyRemove removes a Notify object (*notify*) from the list of Notify objects maintained by an EventMgr object (*eventMgr*).

eventMgr	a handle to the EventMgr object.
notify	a handle to a Notify object.

Return Values

[MPIMessageOK](#)

See Also

MPIEventMgrConfig

Definition

```
typedef MPIEmpty MPIEventMgrConfig;
```

Change History: Modified in the 03.03.00

Description

MPIEventMgrConfig is an empty structure. It is a placeholder for possible future EventMgr configurations.

See Also

MEIEventMgrServiceConfig

Definition

```
typedef struct MEIEventMgrServiceConfig {  
    MPI_BOOL  allProcesses; /* TRUE => collect  
                               events from all processes,  
                               else EventMgr process only */  
} MEIEventMgrServiceConfig;
```

Change History: Modified in the 03.03.00

Description

allProcesses

is a boolean value. If allProcesses=TRUE, then the event manager will handle events originating from all processes. If allProcesses=FALSE, then the event manager will only handle events originating from the same process in which the event manager was created.

See Also

MPIEventMgrMessage

Definition

```
typedef enum {  
    MPIEventMgrMessageEVENTMGR_INVALID,  
} MPIEventMgrMessage;
```

Description

MPIEventMessage is an enumeration of the EventMgr error messages that can be returned by the MPI library.

MPIEventMgrMessageEVENTMGR_INVALID

Not supported.

Sample Code

```
MPIControl      control;  
MPIEventMgr     eventMgr;  
long            returnValue;  
.  
eventMgr =  
    mpiEventMgrCreate(control);  
returnValue =  
    mpiEventMgrValidate(eventMgr);
```

See Also

[MPIEventMgr](#) | [mpiEventMgrCreate](#) | [mpiEventMgrValidate](#)