EventMgr Objects

Introduction

An **EventMgr**

- Obtains asynchronous events from the Control object(s) that the EventMgr is associated with
- Generates Event objects for enabled event sources
- Awakens any threads that are waiting for events.

Methods

Create, Delete, Validate Methods

mpiEventMgr**Create**	Create EventMgr object
mpiEventMgr**Delete**	Delete EventMgr object
mpiEventMgr**Validate**	Validate EventMgr object

Configuration and Information Methods

mpiEventMgr**ConfigGet**	Get EventMgr config
mpiEventMgr**ConfigSet**	Set EventMgr config
mpiEventMgr**Event**	Request event notification for all Notify objects on EventMgr's list
meiEventMgr**ServiceConfigGet**	Get processes that EventMgr will service
meiEventMgr**ServiceConfigSet**	Set processes that EventMgr will service

Action Methods

mpiEventMgr**Flush**	Flush pending EventMgr events
mpiEventMgr**Service**	Get list of all pending asynchronous events

Relational Methods

List Methods- for Control Objects

mpiEventMgr**Control**	Return handle of indexth Control object in list
mpiEventMgr**ControlAppend**	Append Control's handle to list
mpiEventMgr**ControlCount**	Count the number of Control objects associated with EventMgr (in list)
mpiEventMgr**ControlFirst**	Return handle to first Control object in list
mpiEventMgr**ControlIndex**	Return the index of a Control object in list
mpiEventMgr**ControlInsert**	Insert Control handle into list
mpiEventMgr**ControlLast**	Get handle to last Control object in list
mpiEventMgr**ControlListGet**	Get list of Control objects associated with EventMgr
mpiEventMgr**ControlListSet**	Create a list of Control objects associated with EventMgr
mpiEventMgr**ControlNext**	Get handle to next Control object in list
mpiEventMgr**ControlPrevious**	Get handle to previous Control object in list
mpiEventMgr**ControlRemove**	Remove a Control object's handle from list

List Methods- for Notify Objects

mpiEventMgr**Notify**	Return handle to a Notify object associated with EventMgr

mpiEventMgr**NotifyAppend**	Append Notify object to list
mpiEventMgr**NotifyCount**	Return number of Notify objects in list
mpiEventMgr**NotifyFirst**	Get first Notify object in list
mpiEventMgr**NotifyIndex**	Get index value for a Notify object in list
mpiEventMgr**NotifyInsert**	Place a Notify object after another Notify object in list
mpiEventMgr**NotifyLast**	Get handle to the Notify object that is last on the list
mpiEventMgr**NotifyListGet**	Get a list of Notify objects
mpiEventMgr**NotifyListSet**	Create a list of Notify objects
mpiEventMgr**NotifyNext**	Get the Notify object just after notify in list
mpiEventMgr**NotifyPrevious**	Get the Notify object just before notify in list
mpiEventMgr**NotifyRemove**	Remove a Notify object from list

Data Types

MPIEventMgr**Message**
MEIEventMgr**ServiceConfig**

mpiEventMgrCreate

Declaration

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

Required Header stdmpi.h

Description

EventMgrCreate 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

EventMgrDelete 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	if *EventMgrDelete* successfully deletes an EventMgr object and invalidates its handle

See Also mpiEventMgrCreate | mpiEventMgrValidate

mpiEventMgrValidate

Declaration `long mpiEventMgrValidate(MPIEventMgr eventMgr)`

Required Header stdmpi.h

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

Return Values	
MPIMessageOK	if EventMgr is a handle to a valid object.

See Also mpiEventMgrCreate | mpiEventMgrDelete

mpiEventMgrConfigGet

Declaration

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

Required Header

stdmpi.h

Description

EventMgrConfigGet 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).

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

Return Values

MPIMessageOK	if *EventMgrConfigGet* successfully gets the EventMgr's configuration and writes it into the structure(s)

See Also

mpiEventMgrConfigSet

mpiEventMgrConfigSet

Declaration

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

Required Header

stdmpi.h

Description

EventMgrConfigSet 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).

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

Return Values

MPIMessageOK	if *EventMgrConfigSet* successfully sets (writes) the EventMgr's configuration using data from the structure(s)

See Also

mpiEventMgrConfigGet

mpiEventMgrEvent

Declaration

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

Required Header

stdmpi.h

Description

EventMgrEvent 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	if *EventMgrEvent* successfully requests that the EventMgr call mpiNotify Event(notify, status) for all Notify objects on that Event Manager's list

See Also

meiEventMgrServiceConfigGet

Declaration

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

Required Header

stdmei.h

Description

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

Return Values

MPIMessageOK	if *EventMgrServiceConfigGet* successfully gets the EventMgr's configuration and writes it into the structure(s).

See Also

meiEventMgrServiceConfigSet | mpiEventMgrService

meiEventMgrServiceConfigSet

Declaration

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

Required Header

stdmei.h

Description

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

Return Values

MPIMessageOK	if *EventMgrServiceConfigSet* successfully sets (writes) the EventMgr's flash configuration using data from the structure(s)

See Also

meiEventMgrServiceConfigGet | mpiEventMgrService

mpiEventMgrFlush

Declaration `long mpiEventMgrFlush(MPIEventMgr eventMgr)`

Required Header stdmpi.h

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

Return Values	
MPIMessageOK	if *EventMgrFlush* successfully flushes any pending EventMgr events

See Also

mpiEventMgrService

Declaration

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

Required Header

stdmpi.h

Description

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

Events generated by sources for which no prior call to ***mpiEventMgrEnable(...)*** has been made are discarded. Events generated by enabled sources are returned in First-In/First-Out (FIFO) order each time a thread calls **mpiEventMgrWait(...)**.

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.

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

Return Values

MPIMessageOK	if *EventMgrService* successfully obtains all pending asynchronous events from the motion controller

See Also

meiEventMgrServiceConfigGet | meiEventMgrServiceConfigSet

mpiEventMgrControl

Declaration

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

Required Header

stdmpi.h

Description

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 ***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	***index*** is a negative number.
MEIListMessageELEMENT_NOT_FOUND	***index*** is greater than or equal to the number of elements in the list.
MPIMessageHANDLE_INVALID	***eventMgr*** is an invalid handle.

See Also

mpiEventMgrControlAppend

Declaration

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

Required Header

stdmpi.h

Description

EventMgrControlAppend 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	if *EventMgrControlAppend* successfully appends control to the list of motion controllers associated with an EventMgr
MPIMessageHANDLE_INVALID	Either ***eventMgr*** or ***control*** is an invalid handle.
MPIMessageOBJECT_ON_LIST	***control*** is already on the list.
MPIMessageNO_MEMORY	Not enough memory was available.

See Also

mpiEventMgrControlCount

Declaration

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

Required Header

stdmpi.h

Description

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 (***eventMgr***)
-1	if ***eventMgr*** is an invalid handle
0	if ***eventMgr*** has no associated motion controllers

See Also

mpiEventMgrControlFirst

Declaration

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

Required Header

stdmpi.h

Description

EventMgrControlFirst returns the first element in the list. This function can be used in conjuntion 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 (***eventMgr***)
MPIHandleVOID	if ***eventMgr*** is invalid if ***eventMgr*** has no associated motion controllers
MPIMessageHANDLE_INVALID	if ***eventMgr*** is an invalid handle.

See Also

mpiEventMgrControlLast

mpiEventMgrControlIndex

Declaration

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

Required Header

stdmpi.h

Description

EventMgrControlIndex 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 ***control*** in the list of motion controllers associated with an EventMgr (***eventMgr***)
-1	if ***eventMgr*** is invalid if ***control*** 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

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

Return Values	
MPIMessageOK	if *EventMgrControlInsert* successfully inserts the Control object (***insert***) after another Control object (***control***) in the list of motion controllers

See Also

mpiEventMgrControlLast

Declaration `MPIControl mpiEventMgrControlLast(MPIEventMgr eventMgr)`

Required Header stdmpi.h

Description **EventMgrControlLast** returns the last element in the list.
This function can be used in conjuntion 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 (***eventMgr***)
MPIHandleVOID	if ***eventMgr*** is invalid if ***eventMgr*** has no associated motion controllers
MPIMessageHANDLE_INVALID	if ***eventMgr*** is an invalid handle.

See Also mpiEventMgrControlFirst

mpiEventMgrControlListGet

Declaration

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

Required Header

stdmpi.h

Description

EventMgrControlListGet 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***.

Return Values

MPIMessageOK	if *EventMgrControlListGet* successfully returns the list of Control objects associated with an EventMgr object

See Also

mpiEventMgrControlListSet

mpiEventMgrControlListSet

Declaration

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

Required Header

stdmpi.h

Description

EventMgrControlListSet 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	if *EventMgrControlListSet* successfully creates a list of Control objects using the Control handles

See Also

mpiEventMgrControlListGet

mpiEventMgrControlNext

Declaration

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

Required Header

stdmpi.h

Description

EventMgrControlNext 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 conjuntion 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 ***control*** in the list of motion controllers associated with an EventMgr (***eventMgr***)
MPIHandleVOID	if ***eventMgr*** is invalid if ***control*** is the last motion controller
MPIMessageHANDLE_INVALID	Either ***eventMgr*** or ***control*** is an invalid handle.

See Also

mpiEventMgrControlPrevious

mpiEventMgrControlPrevious

Declaration

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

Required Header

stdmpi.h

Description

EventMgrControlPrevious 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 conjuntion 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 ***control*** in the list of motion controllers associated with an EventMgr (***eventMgr***)
MPIHandleVOID	if ***eventMgr*** is invalid if ***control*** is the first motion controller
MPIMessageHANDLE_INVALID	Either ***eventMgr*** or ***control*** is an invalid handle.

See Also

mpiEventMgrControlNext

mpiEventMgrControlRemove

Declaration

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

Required Header stdmpi.h

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

Return Values	
MPIMessageOK	if *EventMgrControlRemove* successfully removes a Control object from the list of Control objects associated with an EventMgr (***eventMgr***)

See Also mpiEventMgrControl

mpiEventMgrNotify

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotify 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 ***index***th Notify object 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	***index*** is a negative number.
MEIListMessageELEMENT_NOT_FOUND	***index*** is greater than or equal to the number of elements in the list.
MPIMessageHANDLE_INVALID	***eventMgr*** is an invalid handle.

See Also

mpiEventMgrNotifyAppend

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotifyAppend 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	if *EventMgrNotifyAppend* successfully appends a Notify object to the list of Notify objects maintained by an EventMgr object
MPIMessageHANDLE_INVALID	Either ***eventMgr*** or ***notify*** is an invalid handle.
MPIMessageOBJECT_ON_LIST	if ***notify*** is already on the list.
MPIMessageNO_MEMORY	if not enough memory was available.

See Also

mpiEventMgrNotifyCount

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotifyCount 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 (***eventMgr***)
-1	if ***eventMgr*** is invalid
0	if the list (of Notify objects) is empty

See Also

mpiEventMgrNotifyFirst

Declaration `MPINotify mpiEventMgrNotifyFirst(MPIEventMgr eventMgr)`

Required Header stdmpi.h

Description **EventMgrNotifyFirst** returns the first element in the list. This function can be used in conjuntion 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 (***eventMgr***)
MPIHandleVOID	if ***eventMgr*** is invalid if the list (of Notify objects) is empty
MPIMessageHANDLE_INVALID	if ***eventMgr*** is an invalid handle.

See Also mpiEventMgrNotifyLast

mpiEventMgrNotifyIndex

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotifyIndex 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 ***notify*** in the list (of Notify objects) maintained by an EventMgr (***eventMgr***)
-1	if eventMgr is invalid if ***notify*** was not found in the list

See Also

mpiEventMgrNotifyInsert

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotifyInsert 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	if *EventMgrNotifyInsert* successfully places a Notify object after another Notify object in the list of Notify objects

See Also

mpiEventMgrNotifyLast

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotifyLast returns the last element in the list. This function can be used in conjuntion with mpiEventMgrNotifyPrevious() in order to iterate through the list backwards.

eventMgr	a handle to the EventMgr object.

Return Values

handle	to the last Notify object in the list maintained by an EventMgr (***eventMgr***)
MPIHandleVOID	if ***eventMgr*** is invalid if the list (of Notify objects) is empty
MPIMessageHANDLE_INVALID	if ***eventMgr*** is an invalid handle.

See Also

mpiEventMgrNotifyFirst

mpiEventMgrNotifyListGet

Declaration

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

Required Header stdmpi.h

Description

EventMgrNotifyListGet 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	if *EventMgrNotifyListGet* successfully returns the list of Notify objects for an EventMgr object

See Also mpiEventMgrNotifyListSet

mpiEventMgrNotifyListSet

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotifyListSet 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	if *EventMgrNotifyListSet* successfully creates a list of Notify objects using the Notify handles specified by ***notifyList***

See Also

mpiEventMgrNotifyListGet

mpiEventMgrNotifyNext

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotifyNext returns the next element following "notify" on the list. This function can be used in conjuntion 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 Notify object after another Notify object (***notify***) in the list (of Notify objects) maintained by an EventMgr (***eventMgr***)
MPIHandleVOID	if ***eventMgr*** is invalid if ***notify*** is the last notify object in the list
MPIMessageHANDLE_INVALID	Either ***eventMgr*** or ***notify*** is an invalid handle.

See Also

mpiEventMgrNotifyPrevious

mpiEventMgrNotifyPrevious

Declaration

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

Required Header

stdmpi.h

Description

EventMgrNotifyPrevious returns the previous element prior to "notify" on the list. This function can be used in conjuntion 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

mpiEventMgrNotifyRemove

Declaration

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

Required Header

stdmpi.h

Description

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

Return Values

MPIMessageOK	if *EventMgrNotifyRemove* successfully removes a Notify object from the list of Notify objects maintained by an EventMgr object

See Also

MPIEventMgrMessage

MPIEventMgrMessage

```
typedef enum {

    MPIEventMgrMessageEVENTMGR_INVALID,
} MPIEventMgrMessage;
```

Description

MPIEventMgrMessageEVENTMGR_INVALID	
Meaning	The MPIEventMgr handle passed to an MPIEventMgr method is invalid.
Possible Causes	Either the handle was never initialized or the mpiEventMgrCreate method failed.
Recommendations	Use mpiEventMgrValidate after mpiEventMgrCreate to see if the returned handle is valid.

Sample Code

```
MPIControl      control;
MPIEventMgr     eventMgr;
long            returnValue;
…

eventMgr =
    mpiEventMgrCreate(control);
returnValue =
    mpiEventMgrValidate(eventMgr);
```

See Also

MPIEventMgr | mpiEventMgrCreate | mpiEventMgrValidate

MEIEventMgrServiceConfig

MEIEventMgrServiceConfig

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

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