Service Objects

Introduction

A **Service** object creates and handles threads that help event managers dispatch events. Typically, one will use a Service Object to create threads that will call **mpiEventMgrService(…)** whenever an XMP interrupt occurs. They are a convenient way to have a program automatically deal with event managers and events. Thread handling is something that is different on every operating system. Service objects may therefore have different behaviors on different operating systems. Programmers that are experienced in multi-threaded application programming will probably want to program their own threads that will call **mpiEventMgrService(…)**.

Methods

Create, Delete, Validate Methods

Service**Create**	Create a Service for EventMgr and the threads necessary for it to run.
Service**Delete**	Stop all threads belonging to the Service and deletes the Service.

Configuration and Information Methods

Service**Enable**	Enable or disables the Service.

ServiceCreate

Declaration

```
const Service serviceCreate(MPIEventMgr eventMgr,
                            long        priority,
                            long        sleep)
```

Required Header

service.h

Description

ServiceCreate creates threads for each control associated with ***eventMgr***, flushes ***eventMgr***, and starts threads with priority that call mpiEventMgrService(eventMgr, …) every ***sleep*** milliseconds.

priority is a platform specific variable.

If "priority" is	***Then***
-1	The operating system will choose some default priority for the service’s threads.
>0	*ServiceCreate* will attempt to assign the priority to all of the service’s threads.

If "sleep" is	***Then***
-1	*ServiceCreate* will attempt to create interrupt driven threads.
0	*ServiceCreate* will create threads that call **mpiEventMgrService(eventMgr,...)** as quickly as possible.
>0	*ServiceCreate* will create threads that attempt to call **mpiEventMgrService(eventMgr,...)** every ***sleep*** milliseconds.

Return Values

handle	to a Service object
MPIHandleVOID	if the Service could not be created

See Also

mpiEventMgrService | ServiceDelete

ServiceDelete

Declaration	`long serviceDelete(Service service)`
Required Header	service.h
Description	**ServiceDelete** alerts all threads that they should end, waits for all threads to end, and frees the memory allocated to ***service***.

Return Values

MPIMessageOK	if *serviceDelete* successfully deletes a Service object and invalidates its handle

See Also ServiceCreate

ServiceEnable

Declaration

```
long serviceEnable(Service service,
                   long    enabled)
```

Required Header

service.h

Description

ServiceEnable enables or disables all threads belonging to Service.

If "enabled" is	***Then***
FALSE	*serviceEnable* will disable ***service***.
TRUE	*serviceEnable* will enable ***service***.

Return Values	
handle	to a Service object
MPIHandleVOID	if the Service could not be created

See Also