



RtCreateSharedMemory(...)

RtCreateSharedMemory creates a named or unnamed region of physical memory that can be mapped by any process.

Prototype

HANDLE

```
RtCreateSharedMemory (  
    DWORD    flProtect,  
    DWORD    MaximumSizeHigh,  
    DWORD    MaximumSizeLow,  
    LPCTSTR  lpName,  
    VOID     **location  
);
```



RtCreateSharedMemory(...)

Parameters

`flProtect`

(ignored by RTSS). The protection desired for the shared memory view. This parameter can be one of the following values:

PAGE_READONLY

Gives read-only access to the committed region of pages. An attempt to write to or execute the committed region results in an access violation.

PAGE_READWRITE

Gives read-write access to the committed region of pages.

`MaximumSizeHigh`

the high-order 32 bits of the size of the shared memory object.

`MaximumSizeLow`

the low-order 32 bits of the size of the shared memory object.



RtCreateSharedMemory(...)

Parameters

- lpName* a pointer to a null-terminated string specifying the name of the shared memory object. The name can contain any character except the backslash (\).
If this parameter matches the name of an existing named shared memory object, the function requests access to the shared memory object with the protection specified by *flProtect*.
If *lpName* matches the name of an existing event, mutex, or semaphore object, the function fails and `GetLastError` returns ***ERROR_INVALID_HANDLE***. This occurs because event, mutex, semaphore, and shared memory objects share the same namespace.
If *lpName* is ***NULL***, the mapping object is created without a name.
- Location* a pointer to a location where the virtual address of the shared memory will be stored.



RtCreateSharedMemory(...)

Return Values

If the function succeeds, the return value is a handle to the shared memory object. If the object existed before the function call, `GetLastError` returns *ERROR_ALREADY_EXISTS*, and the return value is a valid handle to the existing shared memory object (with its current size, not the new specified size). If the mapping object did not exist, `GetLastError` returns zero and the location is set.

If the function fails, the return value is *NULL*. To get extended error information, call `GetLastError`.

Comments

The handle that `RtCreateSharedMemory` returns has full access to the new shared memory object. Shared memory objects can be shared by name. For information on opening a shared memory object by name, see `RtOpenSharedMemory`.

To close a shared memory object, an application must close its handle by calling `RtCloseHandle`.

When all handles to the shared memory object representing the physical memory are closed, the object is destroyed and physical memory is returned to the system.



RtOpenSharedMemory(...)

RtOpenSharedMemory opens a named physical-mapping object.

Prototyp

HANDLE

```
RtOpenSharedMemory (  
    DWORD    DesiredAccess,  
    BOOL     bInheritHandle,  
    LPCTSTR  lpName,  
    VOID     **location  
);
```



RtOpenSharedMemory(...)

Parameters

`DesiredAccess` the access mode. The RTSS environment always grants read and write access. This parameter can be one of the following values:

`SHM_MAP_WRITE`

Read-write access. The target shared memory object must have been created with `PAGE_READWRITE` protection. A read-write view of the shared memory is mapped.

`SHM_MAP_READ`

Read-only access. The target shared memory object must have been created with `PAGE_READWRITE` or `PAGE_READ` protection. A read-only view of the shared memory is mapped.



RtOpenSharedMemory(...)

Parameters

bInheritHandle (ignored)

lpName A pointer to a string that names the shared memory object to be opened. If there is an open handle to a shared memory object by this name, the open operation succeeds.

location A pointer to a location where the virtual address of the mapping will be stored.



RtOpenSharedMemory(...)

Return Values

If the function succeeds, the return value is an open handle to the specified shared memory object.

If the function fails, the return value is `NULL`. To get extended error information, call `GetLastError`.

Comments

The handle that `RtOpenSharedMemory` returns can be used with `RtCloseHandle` to decrement the reference count to the shared memory object. When the reference count is zero, the object is removed from the system.

In the same process, different calls to `RtOpenSharedMemory` may return different locations because they are mapped into different virtual addresses.



Example

RTX

```
typedef struct {
    unsigned int Win32_index;
    unsigned int RTSS_index;
} STGSTR, *PSTGSTR;

HANDLE hShm;
PSTGSTR pStrg;
DWORD dwMaximumSizeHigh = 0; /*64 KB MAX*/
#define STGSTR_NAME "Storage.Name"

hShm = RtCreateSharedMemory(    0, /*PAGE_READWRITE, PAGE_READONLY WIN32 */
                              dwMaximumSizeHigh,
                              sizeof(STGSTR),
                              TEXT(STGSTR_NAME),
                              (LPVOID) &pStrg);

if(GetLastError()==ERROR_ALREADY_EXISTS) {
    RtWprintf(L"Sdilena memory already exists!\n");
    ExitProcess(1);
}
if (hShm==NULL) {
    RtWprintf(L"Can not create shared memory object!");
    ExitProcess(1);
}
RtWprintf(L"Shared memory object was created. HANDLE = 0x%X.\n", hShm);
```

WIN32

```
hShm = RtOpenSharedMemory( SHM_MAP_READ, FALSE, STGSTR_NAME, (LPVOID *) &pStrg);
```