Portable Systems Group

Windows NT Mutant Specification

Author: *David N. Cutler*

Original Draft 1.0, October 19, 1989
Revision 1.1, November 12, 1989
Revision 1.2, November 28, 1989
Revision 1.3, January 5, 1990

1. Introduction

This specification describes the **Windows NT** *mutant object* which is used to emulate OS/2 2.0 Semaphore Mutexes. Although **Windows NT** provides other, more straightforward, capabilites to synchronize access to critical sections, this object has been included to enable more efficient emulation of the OS/2 2.0 capabilities.

Threads acquire ownership of a mutant object using the **Windows NT** wait services. Only one thread can own a mutant object at a time; however, the owner thread can recursively acquire the mutant object after first gaining ownership. If a thread terminates without releasing ownership of a mutant object, then the mutant object enters the *abandoned* state. The next thread that gains ownership of the mutant object will receive a return status that indicates that the mutant object was previously abandoned. Assigning ownership of an abandoned mutant object to another thread also clears the abandoned state of the mutant object.

Waiting for a mutant object causes the execution of the subject thread to be suspended until the thread can gain ownership of the mutant object. Satisfying the wait for a mutant object assigns ownership to the subject thread.

The following APIs are supported for the mutant object:

NtCreateMutant - Create mutant object and open handle
NtOpenMutant - Open handle to existing mutant object
NtQueryMutant - Get information about mutant object
NtReleaseMutant - Release ownership of a mutant object

1.1 Create Mutant Object

A mutant object can be created and a handle opened for access to the object with the **NtCreateMutant** function:

NTSTATUS
NtCreateMutant (
    OUT PHANDLE *MutantHandle,*
    IN ULONG *DesiredAccess,*
    IN POBJECT_ATTRIBUTES *ObjectAttributes* **OPTIONAL**,
    IN BOOLEAN *InitialOwner*
    );

Parameters:

MutantHandle - A pointer to a variable that receives the mutant object handle value.

DesiredAccess - The desired types of access to the mutant object. The following object type specific access flags can be specified in addition to the *STANDARD_ACCESS_REQUIRED* flags described in the Object Management Specification.

DesiredAccess Flags

MUTANT_QUERY_STATE - Query access to the mutant object is desired.

SYNCHRONIZE - Synchronization access (wait or release) to the mutant object is desired.

MUTANT_ALL_ACCESS - All possible types of access to the mutant object are desired.

ObjectAttributes - An optional pointer to a structure that specifies the object attributes; refer to Object Management Specification for details.

InitialOwner - A boolean value that determines whether the creator of the object desires immediate ownership of the mutant object.

If the *OBJ_OPENIF* flag is specified and a mutant object with the specified name already exists, then a handle to the existing object is opened and the *InitialOwner* parameter is ignored, provided the desired access can be granted. Otherwise, a new mutant object is created and a handle opened to the object with ownership as determined by the *InitialOwner* parameter. The status of the newly created mutant object is set to not abandoned.

1.2 Open Mutant Object

A handle can be opened to an existing mutant object with the **NtOpenMutant** function:

NTSTATUS
NtOpenMutant *(*
    OUT PHANDLE *MutantHandle,*
    IN ULONG *DesiredAccess,*
    IN POBJECT_ATTRIBUTES *ObjectAttributes*
    );

Parameters:

MutantHandle - A pointer to a variable that receives the mutant object handle value.

DesiredAccess - The desired types of access to the mutant object. The following object type specific access flags can be specified in addition to the *STANDARD_ACCESS_REQUIRED* flags described in the Object Management Specification.

DesiredAccess Flags

MUTANT_QUERY_STATE - Query access to the mutant object is desired.

SYNCHRONIZE - Synchronization access (wait or release) to the mutant object is desired.

MUTANT_ALL_ACCESS - All possible types of access to the mutant object are desired.

ObjectAttributes - A pointer to a structure that specifies the object attributes; refer to Object Management Specification for details.

If the desired types of access can be granted, then a handle is opened to the specified mutant object.

1.3 Query Mutant Object

The state of a mutant object can be queried with the **NtQueryMutant** function:

NTSTATUS
NtQueryMutant *(*
  IN HANDLE *MutantHandle,*
  IN MUTANTINFOCLASS *MutantInformationClass,*
  OUT PVOID *MutantInformation,*
  IN ULONG *MutantInformationLength,*
  OUT PULONG *ReturnLength* **OPTIONAL**
  );

Parameters:

MutantHandle - An open handle to a mutant object.

MutantInformationClass - The mutant information class for which information is to be returned.

MutantInformation - A pointer to a buffer that receives the specified information. The format and content of the buffer depend on the specified information class.

MutantInformation Format by Information Class:

MutantBasicInformation - Data type is *MUTANTBASICINFO*.

MUTANTBASICINFO Structure

LONG *CurrentCount* - The current ownership count of the mutant object.

BOOLEAN *AbandonedState* - The current abandoned state of the mutant object.

MutantInformationLength - Specifies the length in bytes of the mutant information buffer.

ReturnLength - An optional pointer to a variable that receives the number of bytes placed in the mutant information buffer.

This function provides the capability to determine the ownership and abandoned state of a mutant object.

1.4 Release Mutant Object

Ownership of a mutant object can be released with the **NtReleaseMutant** function:

NTSTATUS
NtReleaseMutant *(*
    IN HANDLE *MutantHandle,*
    OUT PLONG *PreviousCount* **OPTIONAL**
    );

Parameters:

MutantHandle - An open handle to a mutant object.

PreviousCount - An optional pointer to a variable that receives the previous ownership count of the mutant object.

A mutant object can only be released by a thread that currently owns the mutant object. When the mutant is released, the current count of the mutant object is incremented by one. If the resultant count is one, then the mutant object is no longer owned. Any threads that are waiting for the mutant object are examined to see if their wait can be satisfied.

Revision History:

Original Draft 1.0, October 18, 1989

Revision 1.1, November 12, 1989

1. Added initial ownership parameter to NtCreateMutant.

Revision 1.2, November 28, 1989

1. Change access right required for wait access to be only *SYNCHRONIZE* access.

Revision 1.3, January 5, 1990

1. Change type name of object attributes parameter and refer to the Object Management Specification for the definition of this parameter.

2. Change the description of the desired access flags to include standard rights, object specific rights, and generic rights.

3. Delete the handle flags and object names parameters from the **NtOpenMutant** service and replace with a pointer to an object attributes structure.