CreateEventEx function

Creates or opens a named or unnamed event object and returns a handle to the object.

Syntax

C++

```
HANDLE WINAPI CreateEventEx(
  _In_opt_ LPSECURITY_ATTRIBUTES lpEventAttributes,
  _In_opt_ LPCTSTR               lpName,
  _In_     DWORD                 dwFlags,
  _In_     DWORD                 dwDesiredAccess
);
```

Parameters

lpEventAttributes [in, optional]

A pointer to a SECURITY_ATTRIBUTES structure. If *lpEventAttributes* is **NULL**, the event handle cannot be inherited by child processes.

The **lpSecurityDescriptor** member of the structure specifies a security descriptor for the new event. If *lpEventAttributes* is **NULL**, the event gets a default security descriptor. The ACLs in the default security descriptor for an event come from the primary or impersonation token of the creator.

lpName [in, optional]

The name of the event object. The name is limited to **MAX_PATH** characters. Name comparison is case sensitive.

If *lpName* is **NULL**, the event object is created without a name.

If *lpName* matches the name of another kind of object in the same namespace (such as an existing semaphore, mutex, waitable timer, job, or file-mapping object), the function fails and the **GetLastError** function returns **ERROR_INVALID_HANDLE**. This occurs because these objects share the same namespace.

The name can have a "Global\" or "Local\" prefix to explicitly create the object in the global or session namespace. The remainder of the name can contain any character except the backslash character (\). For more information, see Kernel Object Namespaces. Fast user switching is implemented using Terminal Services sessions. Kernel object names must follow the guidelines outlined for Terminal Services so that applications can support multiple users.

The object can be created in a private namespace. For more information, see Object Namespaces.

dwFlags [in]

This parameter can be one or more of the following values.

Value	Meaning
CREATE_EVENT_INITIAL_SET 0x00000002	The initial state of the event object is signaled; otherwise, it is nonsignaled.
CREATE_EVENT_MANUAL_RESET 0x00000001	The event must be manually reset using the ResetEvent function. Any number of waiting threads, or threads that subsequently begin wait operations for the specified event object, can be released while the object's state is signaled. If this flag is not specified, the system automatically resets the event after releasing a single waiting thread.

dwDesiredAccess [in]

The access mask for the event object. For a list of access rights, see Synchronization Object Security and Access Rights.

Return value

If the function succeeds, the return value is a handle to the event object. If the named event object existed before the function call, the function returns a handle to the existing object and GetLastError returns **ERROR_ALREADY_EXISTS**.

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

Remarks

Any thread of the calling process can specify the event-object handle in a call to one of the wait functions. The single-object wait functions return when the state of the specified object is signaled. The multiple-object wait functions can be instructed to return either when any one or when all of the specified objects are signaled. When a wait function returns, the waiting thread is released to continue its execution.

The initial state of the event object is specified by the *dwFlags* parameter. Use the SetEvent function to set the state of an event object to signaled. Use the ResetEvent function to reset the state of an event object to nonsignaled.

When the state of a manual-reset event object is signaled, it remains signaled until it is explicitly reset to nonsignaled by the ResetEvent function. Any number of waiting threads, or threads that subsequently begin wait operations for the specified event object, can be released while the object's state is signaled.

Multiple processes can have handles of the same event object, enabling use of the object for interprocess synchronization. The following object-sharing mechanisms are available:

- A child process created by the CreateProcess function can inherit a handle to an event object if the *lpEventAttributes* parameter of **CreateEvent** enabled inheritance.
- A process can specify the event-object handle in a call to the DuplicateHandle function to create a duplicate handle that can be used by another process.
- A process can specify the name of an event object in a call to the OpenEvent or **CreateEvent** function.

Use the CloseHandle function to close the handle. The system closes the handle automatically when the process terminates. The event object is destroyed when its last handle has been closed.

Requirements

Minimum supported client	Windows Vista [desktop apps | Windows Store apps]
Minimum supported server	Windows Server 2008 [desktop apps | Windows Store apps]
Minimum supported phone	Windows Phone 8
Header	WinBase.h on Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 (include Windows.h); Synchapi.h on Windows 8 and Windows Server 2012
Library	Kernel32.lib
DLL	Kernel32.dll
Unicode and ANSI names	**CreateEventExW** (Unicode) and **CreateEventExA** (ANSI)

See also

CloseHandle
Event Objects
Object Names
Synchronization Functions

© 2016 Microsoft