CreateEvent function

Creates or opens a named or unnamed event object.

To specify an access mask for the object, use the **CreateEventEx** function.

Syntax

C++

```
HANDLE WINAPI CreateEvent(
  _In_opt_ LPSECURITY_ATTRIBUTES lpEventAttributes,
  _In_     BOOL                  bManualReset,
  _In_     BOOL                  bInitialState,
  _In_opt_ LPCTSTR               lpName
);
```

Parameters

lpEventAttributes [in, optional]

A pointer to a **SECURITY_ATTRIBUTES** structure. If this parameter is **NULL**, the 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.

bManualReset [in]

If this parameter is **TRUE**, the function creates a manual-reset event object, which requires the use of the **ResetEvent** function to set the event state to nonsignaled. If this parameter is **FALSE**, the function creates an auto-reset event object, and system automatically resets the event state to nonsignaled after a single waiting thread has been released.

bInitialState [in]

If this parameter is **TRUE**, the initial state of the event object is signaled; otherwise, it is nonsignaled.

lpName [in, optional]

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

If *lpName* matches the name of an existing named event object, this function requests the

EVENT_ALL_ACCESS access right. In this case, the *bManualReset* and *bInitialState* parameters are ignored because they have already been set by the creating process. If the *lpEventAttributes* parameter is not **NULL**, it determines whether the handle can be inherited, but its security-descriptor member is ignored.

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**.

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

The handle returned by **CreateEvent** has the **EVENT_ALL_ACCESS** access right; it can be used in any function that requires a handle to an event object, provided that the caller has been granted access. If an event is created from a service or a thread that is impersonating a different user, you can either apply a security descriptor to the event when you create it, or change the default security descriptor for the creating process by changing its default DACL. For more information, see **Synchronization Object Security and Access Rights**.

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 *bInitialState* 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.

When the state of an auto-reset event object is signaled, it remains signaled until a single waiting thread is released; the system then automatically resets the state to nonsignaled. If no threads are waiting, the event object's state remains 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.

Examples

For an example that uses **CreateEvent**, see Using Event Objects.

Requirements

Minimum supported client	Windows XP [desktop apps | Windows Store apps]
Minimum supported server	Windows Server 2003 [desktop apps | Windows Store apps]
Header	WinBase.h on Windows XP, Windows Server 2003, 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	**CreateEventW** (Unicode) and **CreateEventA** (ANSI)

See also

CloseHandle
CreateEventEx
CreateProcess
DuplicateHandle
Event Objects
Object Names
OpenEvent
ResetEvent
SECURITY_ATTRIBUTES
SetEvent
Synchronization Functions