CloseHandle function

Closes an open object handle.

Syntax

C++

```
BOOL WINAPI CloseHandle(
  _In_ HANDLE hObject
);
```

Parameters

hObject [in]

A valid handle to an open object.

Return value

If the function succeeds, the return value is nonzero.

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

If the application is running under a debugger, the function will throw an exception if it receives either a handle value that is not valid or a pseudo-handle value. This can happen if you close a handle twice, or if you call **CloseHandle** on a handle returned by the **FindFirstFile** function instead of calling the **FindClose** function.

Remarks

The **CloseHandle** function closes handles to the following objects:

- Access token
- Communications device
- Console input
- Console screen buffer
- Event
- File
- File mapping
- I/O completion port
- Job

- Mailslot
- Memory resource notification
- Mutex
- Named pipe
- Pipe
- Process
- Semaphore
- Thread
- Transaction
- Waitable timer

The documentation for the functions that create these objects indicates that **CloseHandle** should be used when you are finished with the object, and what happens to pending operations on the object after the handle is closed. In general, **CloseHandle** invalidates the specified object handle, decrements the object's handle count, and performs object retention checks. After the last handle to an object is closed, the object is removed from the system. For a summary of the creator functions for these objects, see Kernel Objects.

Generally, an application should call **CloseHandle** once for each handle it opens. It is usually not necessary to call **CloseHandle** if a function that uses a handle fails with ERROR_INVALID_HANDLE, because this error usually indicates that the handle is already invalidated. However, some functions use ERROR_INVALID_HANDLE to indicate that the object itself is no longer valid. For example, a function that attempts to use a handle to a file on a network might fail with ERROR_INVALID_HANDLE if the network connection is severed, because the file object is no longer available. In this case, the application should close the handle.

If a handle is transacted, all handles bound to a transaction should be closed before the transaction is committed. If a transacted handle was opened by calling CreateFileTransacted with the FILE_FLAG_DELETE_ON_CLOSE flag, the file is not deleted until the application closes the handle and calls CommitTransaction. For more information about transacted objects, see Working With Transactions.

Closing a thread handle does not terminate the associated thread or remove the thread object. Closing a process handle does not terminate the associated process or remove the process object. To remove a thread object, you must terminate the thread, then close all handles to the thread. For more information, see Terminating a Thread. To remove a process object, you must terminate the process, then close all handles to the process. For more information, see Terminating a Process.

Closing a handle to a file mapping can succeed even when there are file views that are still open. For more information, see Closing a File Mapping Object.

Do not use the **CloseHandle** function to close a socket. Instead, use the closesocket function, which releases all resources associated with the socket including the handle to the socket object. For more information, see Socket Closure.

Examples

For an example, see Taking a Snapshot and Viewing Processes.

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps | Windows Store apps]
Minimum supported server	Windows 2000 Server [desktop apps | Windows Store apps]
Minimum supported phone	Windows Phone 8
Header	Winbase.h (include Windows.h)
Library	Kernel32.lib
DLL	Kernel32.dll

See also

CreateFile
CreateFileTransacted
DeleteFile
FindClose
FindFirstFile
Handle and Object Functions
Kernel Objects
Object Interface