CopyFileEx function

Copies an existing file to a new file, notifying the application of its progress through a callback function.

To perform this operation as a transacted operation, use the CopyFileTransacted function.

Syntax

C++

```
BOOL WINAPI CopyFileEx(
  _In_     LPCTSTR            lpExistingFileName,
  _In_     LPCTSTR            lpNewFileName,
  _In_opt_ LPPROGRESS_ROUTINE lpProgressRoutine,
  _In_opt_ LPVOID             lpData,
  _In_opt_ LPBOOL             pbCancel,
  _In_     DWORD              dwCopyFlags
);
```

Parameters

lpExistingFileName [in]
The name of an existing file.

In the ANSI version of this function, the name is limited to **MAX_PATH** characters. To extend this limit to 32,767 wide characters, call the Unicode version of the function and prepend "\\?\" to the path. For more information, see Naming a File.

If *lpExistingFileName* does not exist, the **CopyFileEx** function fails, and the GetLastError function returns **ERROR_FILE_NOT_FOUND**.

lpNewFileName [in]
The name of the new file.

In the ANSI version of this function, the name is limited to **MAX_PATH** characters. To extend this limit to 32,767 wide characters, call the Unicode version of the function and prepend "\\?\" to the path. For more information, see Naming a File.

lpProgressRoutine [in, optional]
The address of a callback function of type **LPPROGRESS_ROUTINE** that is called each time another portion of the file has been copied. This parameter can be **NULL**. For more information on the progress

callback function, see the CopyProgressRoutine function.

lpData [in, optional]
The argument to be passed to the callback function. This parameter can be **NULL**.

pbCancel [in, optional]
If this flag is set to **TRUE** during the copy operation, the operation is canceled. Otherwise, the copy operation will continue to completion.

dwCopyFlags [in]
Flags that specify how the file is to be copied. This parameter can be a combination of the following values.

Value	Meaning
COPY_FILE_ALLOW_DECRYPTED_DESTINATION 0x00000008	An attempt to copy an encrypted file will succeed even if the destination copy cannot be encrypted.
COPY_FILE_COPY_SYMLINK 0x00000800	If the source file is a symbolic link, the destination file is also a symbolic link pointing to the same file that the source symbolic link is pointing to. **Windows Server 2003 and Windows XP:** This value is not supported.
COPY_FILE_FAIL_IF_EXISTS 0x00000001	The copy operation fails immediately if the target file already exists.
COPY_FILE_NO_BUFFERING 0x00001000	The copy operation is performed using unbuffered I/O, bypassing system I/O cache resources. Recommended for very large file transfers. **Windows Server 2003 and Windows XP:** This value is not supported.
COPY_FILE_	The file is copied and the original file is opened for write access.

OPEN_SOURCE_FOR_WRITE 0x00000004	
COPY_FILE_RESTARTABLE 0x00000002	Progress of the copy is tracked in the target file in case the copy fails. The failed copy can be restarted at a later time by specifying the same values for *lpExistingFileName* and *lpNewFileName* as those used in the call that failed. This can significantly slow down the copy operation as the new file may be flushed multiple times during the copy operation.

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 *lpProgressRoutine* returns **PROGRESS_CANCEL** due to the user canceling the operation, **CopyFileEx** will return zero and GetLastError will return **ERROR_REQUEST_ABORTED**. In this case, the partially copied destination file is deleted.

If *lpProgressRoutine* returns **PROGRESS_STOP** due to the user stopping the operation, **CopyFileEx** will return zero and GetLastError will return **ERROR_REQUEST_ABORTED**. In this case, the partially copied destination file is left intact.

Remarks

This function preserves extended attributes, OLE structured storage, NTFS file system alternate data streams, security resource attributes, and file attributes.

Windows 7, Windows Server 2008 R2, Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP: Security resource attributes (**ATTRIBUTE_SECURITY_INFORMATION**) for the existing file are not copied to the new file until Windows 8 and Windows Server 2012.

The security resource properties (**ATTRIBUTE_SECURITY_INFORMATION**) for the existing file are copied to the new file.

Windows 7, Windows Server 2008 R2, Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP: Security resource properties for the existing file are not copied to the new file until Windows 8 and Windows Server 2012.

This function fails with **ERROR_ACCESS_DENIED** if the destination file already exists and has the **FILE_ATTRIBUTE_HIDDEN** or **FILE_ATTRIBUTE_READONLY** attribute set.

When encrypted files are copied using **CopyFileEx**, the function attempts to encrypt the destination file with the keys used in the encryption of the source file. If this cannot be done, this function attempts to encrypt the destination file with default keys. If both of these methods cannot be done, **CopyFileEx** fails with an **ERROR_ENCRYPTION_FAILED** error code. If you want **CopyFileEx** to complete the copy operation even if the destination file cannot be encrypted, include the **COPY_FILE_ALLOW_DECRYPTED_DESTINATION** as the value of the *dwCopyFlags* parameter in your call to **CopyFileEx**.

If **COPY_FILE_COPY_SYMLINK** is specified, the following rules apply:

- If the source file is a symbolic link, the symbolic link is copied, not the target file.
- If the source file is not a symbolic link, there is no change in behavior.
- If the destination file is an existing symbolic link, the symbolic link is overwritten, not the target file.
- If **COPY_FILE_FAIL_IF_EXISTS** is also specified, and the destination file is an existing symbolic link, the operation fails in all cases.

If **COPY_FILE_COPY_SYMLINK** is not specified, the following rules apply:

- If **COPY_FILE_FAIL_IF_EXISTS** is also specified, and the destination file is an existing symbolic link, the operation fails only if the target of the symbolic link exists.
- If **COPY_FILE_FAIL_IF_EXISTS** is not specified, there is no change in behavior.

Windows 7, Windows Server 2008 R2, Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP: If you are writing an application that is optimizing file copy operations across a LAN, consider using the TransmitFile function from Windows Sockets (Winsock). **TransmitFile** supports high-performance network transfers and provides a simple interface to send the contents of a file to a remote computer. To use **TransmitFile**, you must write a Winsock client application that sends the file from the source computer as well as a Winsock server application that uses other Winsock functions to receive the file on the remote computer.

In Windows 8 and Windows Server 2012, this function is supported by the following technologies.

Technology	Supported
Server Message Block (SMB) 3.0 protocol	Yes
SMB 3.0 Transparent Failover (TFO)	Yes
SMB 3.0 with Scale-out File Shares (SO)	Yes
Cluster Shared Volume File System (CsvFS)	Yes
Resilient File System (ReFS)	Yes

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]
Header	WinBase.h (include Windows.h)
Library	Kernel32.lib
DLL	Kernel32.dll
Unicode and ANSI names	**CopyFileExW** (Unicode) and **CopyFileExA** (ANSI)

See also

CopyFile
CopyFileTransacted
CopyProgressRoutine
CreateFile
File Attribute Constants
File Management Functions
MoveFile
MoveFileWithProgress
Symbolic Links
TransmitFile