GetQueuedCompletionStatusEx function

Retrieves multiple completion port entries simultaneously. It waits for pending I/O operations that are associated with the specified completion port to complete.

To dequeue I/O completion packets one at a time, use the **GetQueuedCompletionStatus** function.

Syntax

C++

```
BOOL WINAPI GetQueuedCompletionStatusEx(
  _In_  HANDLE             CompletionPort,
  _Out_ LPOVERLAPPED_ENTRY lpCompletionPortEntries,
  _In_  ULONG              ulCount,
  _Out_ PULONG             ulNumEntriesRemoved,
  _In_  DWORD              dwMilliseconds,
  _In_  BOOL               fAlertable
);
```

Parameters

CompletionPort [in]
A handle to the completion port. To create a completion port, use the **CreateIoCompletionPort** function.

lpCompletionPortEntries [out]
On input, points to a pre-allocated array of **OVERLAPPED_ENTRY** structures.

On output, receives an array of **OVERLAPPED_ENTRY** structures that hold the entries. The number of array elements is provided by *ulNumEntriesRemoved*.

The number of bytes transferred during each I/O, the completion key that indicates on which file each I/O occurred, and the overlapped structure address used in each original I/O are all returned in the *lpCompletionPortEntries* array.

ulCount [in]
The maximum number of entries to remove.

ulNumEntriesRemoved [out]
A pointer to a variable that receives the number of entries actually removed.

dwMilliseconds [in]

The number of milliseconds that the caller is willing to wait for a completion packet to appear at the completion port. If a completion packet does not appear within the specified time, the function times out and returns **FALSE**.

If *dwMilliseconds* is **INFINITE** (0xFFFFFFFF), the function will never time out. If *dwMilliseconds* is zero and there is no I/O operation to dequeue, the function will time out immediately.

fAlertable [in]

If this parameter is **FALSE**, the function does not return until the time-out period has elapsed or an entry is retrieved.

If the parameter is **TRUE** and there are no available entries, the function performs an alertable wait. The thread returns when the system queues an I/O completion routine or APC to the thread and the thread executes the function.

A completion routine is queued when the ReadFileEx or **WriteFileEx** function in which it was specified has completed, and the calling thread is the thread that initiated the operation. An APC is queued when you call QueueUserAPC.

Return value

Returns nonzero (**TRUE**) if successful or zero (**FALSE**) otherwise.

To get extended error information, call **GetLastError**.

Remarks

This function associates a thread with the specified completion port. A thread can be associated with at most one completion port.

This function returns **TRUE** when at least one pending I/O is completed, but it is possible that one or more I/O operations failed. Note that it is up to the user of this function to check the list of returned entries in the *lpCompletionPortEntries* parameter to determine which of them correspond to any possible failed I/O operations by looking at the status contained in the **lpOverlapped** member in each OVERLAPPED_ENTRY.

This function returns **FALSE** when no I/O operation was dequeued. This typically means that an error occurred while processing the parameters to this call, or that the *CompletionPort* handle was closed or is otherwise invalid. The **GetLastError** function provides extended error information.

If a call to **GetQueuedCompletionStatusEx** fails because the handle associated with it is closed, the function returns **FALSE** and **GetLastError** will return **ERROR_ABANDONED_WAIT_0**.

Server applications may have several threads calling the **GetQueuedCompletionStatusEx** function for the same completion port. As I/O operations complete, they are queued to this port in first-in-first-out order. If a thread is actively waiting on this call, one or more queued requests complete the call for that thread only.

For more information on I/O completion port theory, usage, and associated functions, see **I/O Completion**

Ports.

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 Vista [desktop apps only]
Minimum supported server	Windows Server 2008 [desktop apps only]
Header	IoAPI.h (include Windows.h); WinBase.h on Windows Server 2008 R2, Windows 7, Windows Server 2008, and Windows Vista (include Windows.h)
Library	Kernel32.lib
DLL	Kernel32.dll

See also

Overview Topics

File Management Functions

I/O Completion Ports
Using the Windows Headers
Functions
ConnectNamedPipe
CreateIoCompletionPort
DeviceIoControl
GetQueuedCompletionStatusEx
LockFileEx
ReadFile
PostQueuedCompletionStatus
TransactNamedPipe
WaitCommEvent
WriteFile