QueueUserAPC function

Adds a user-mode asynchronous procedure call (APC) object to the APC queue of the specified thread.

Syntax

C++

```
DWORD WINAPI QueueUserAPC(
  _In_ PAPCFUNC  pfnAPC,
  _In_ HANDLE    hThread,
  _In_ ULONG_PTR dwData
);
```

Parameters

pfnAPC [in]
A pointer to the application-supplied APC function to be called when the specified thread performs an alertable wait operation. For more information, see APCProc.

hThread [in]
A handle to the thread. The handle must have the **THREAD_SET_CONTEXT** access right. For more information, see Synchronization Object Security and Access Rights.

dwData [in]
A single value that is passed to the APC function pointed to by the *pfnAPC* parameter.

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**.

Windows Server 2003 and Windows XP: There are no error values defined for this function that can be retrieved by calling **GetLastError**.

Remarks

The APC support provided in the operating system allows an application to queue an APC object to a thread. To ensure successful execution of functions used by the APC, APCs should be queued only to threads in the caller's

process.

> **Note** Queuing APCs to threads outside the caller's process is not recommended for a number of reasons. DLL rebasing can cause the addresses of functions used by the APC to be incorrect when the functions are executed outside the caller's process. Similarly, if a 64-bit process queues an APC to a 32-bit process or vice versa, addresses will be incorrect and the application will crash. Other factors can prevent successful function execution, even if the address is known.

Each thread has its own APC queue. The queuing of an APC is a request for the thread to call the APC function. The operating system issues a software interrupt to direct the thread to call the APC function.

When a user-mode APC is queued, the thread is not directed to call the APC function unless it is in an alertable state. After the thread is in an alertable state, the thread handles all pending APCs in first in, first out (FIFO) order, and the wait operation returns **WAIT_IO_COMPLETION**. A thread enters an alertable state by using SleepEx, SignalObjectAndWait, WaitForSingleObjectEx, WaitForMultipleObjectsEx, or MsgWaitForMultipleObjectsEx to perform an alertable wait operation.

If an application queues an APC before the thread begins running, the thread begins by calling the APC function. After the thread calls an APC function, it calls the APC functions for all APCs in its APC queue.

It is possible to sleep or wait for an object within the APC. If you perform an alertable wait inside an APC, it will recursively dispatch the APCs. This can cause a stack overflow.

When the thread is terminated using the ExitThread or TerminateThread function, the APCs in its APC queue are lost. The APC functions are not called.

When the thread is in the process of being terminated, calling QueueUserAPC to add to the thread's APC queue will fail with **(31) ERROR_GEN_FAILURE**.

Note that the ReadFileEx, SetWaitableTimer, and WriteFileEx functions are implemented using an APC as the completion notification callback mechanism.

To compile an application that uses this function, define **_WIN32_WINNT** as 0x0400 or later. For more information, see Using the Windows Headers.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]

Header	WinBase.h on Windows XP, Windows Server 2003, Windows Vista, Windows 7, Windows Server 2008, and Windows Server 2008 R2 (include Windows.h); Processthreadsapi.h on Windows 8 and Windows Server 2012
Library	Kernel32.lib
DLL	Kernel32.dll

See also

APCProc
Asynchronous Procedure Calls
Synchronization Functions