

# VirtualFree function

Releases, decommits, or releases and decommits a region of pages within the virtual address space of the calling process.

To free memory allocated in another process by the [VirtualAllocEx](#) function, use the [VirtualFreeEx](#) function.

## Syntax

**C++**

```
BOOL WINAPI VirtualFree(  
    _In_ LPVOID lpAddress,  
    _In_ SIZE_T dwSize,  
    _In_ DWORD  dwFreeType  
);
```

## Parameters

*lpAddress* [in]

A pointer to the base address of the region of pages to be freed.

If the *dwFreeType* parameter is **MEM\_RELEASE**, this parameter must be the base address returned by the [VirtualAlloc](#) function when the region of pages is reserved.

*dwSize* [in]

The size of the region of memory to be freed, in bytes.

If the *dwFreeType* parameter is **MEM\_RELEASE**, this parameter must be 0 (zero). The function frees the entire region that is reserved in the initial allocation call to [VirtualAlloc](#).

If the *dwFreeType* parameter is **MEM\_DECOMMIT**, the function decommits all memory pages that contain one or more bytes in the range from the *lpAddress* parameter to (*lpAddress*+*dwSize*). This means, for example, that a 2-byte region of memory that straddles a page boundary causes both pages to be decommitted. If *lpAddress* is the base address returned by [VirtualAlloc](#) and *dwSize* is 0 (zero), the function decommits the entire region that is allocated by **VirtualAlloc**. After that, the entire region is in the reserved state.

*dwFreeType* [in]

The type of free operation. This parameter can be one of the following values.

Value	Meaning
<b>MEM_DECOMMIT</b> 0x4000	<p>Decommits the specified region of committed pages. After the operation, the pages are in the reserved state.</p> <p>The function does not fail if you attempt to decommit an uncommitted page. This means that you can decommit a range of pages without first determining the current commitment state.</p> <p>Do not use this value with <b>MEM_RELEASE</b>.</p> <p>The <b>MEM_DECOMMIT</b> value is not supported when the <i>lpAddress</i> parameter provides the base address for an enclave.</p>
<b>MEM_RELEASE</b> 0x8000	<p>Releases the specified region of pages. After this operation, the pages are in the free state.</p> <p>If you specify this value, <i>dwSize</i> must be 0 (zero), and <i>lpAddress</i> must point to the base address returned by the <a href="#">VirtualAlloc</a> function when the region is reserved. The function fails if either of these conditions is not met.</p> <p>If any pages in the region are committed currently, the function first decommits, and then releases them.</p> <p>The function does not fail if you attempt to release pages that are in different states, some reserved and some committed. This means that you can release a range of pages without first determining the current commitment state.</p> <p>Do not use this value with <b>MEM_DECOMMIT</b>.</p>

## Return value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is 0 (zero). To get extended error information, call [GetLastError](#).

## Remarks

Each page of memory in a process virtual address space has a [Page State](#). The **VirtualFree** function can decommit a range of pages that are in different states, some committed and some uncommitted. This means that you can decommit a range of pages without first determining the current commitment state of each page. Decommitting a page releases its physical storage, either in memory or in the paging file on disk.

If a page is decommitted but not released, its state changes to reserved. Subsequently, you can call [VirtualAlloc](#)

to commit it, or **VirtualFree** to release it. Attempts to read from or write to a reserved page results in an access violation exception.

The **VirtualFree** function can release a range of pages that are in different states, some reserved and some committed. This means that you can release a range of pages without first determining the current commitment state of each page. The entire range of pages originally reserved by the **VirtualAlloc** function must be released at the same time.

If a page is released, its state changes to free, and it is available for subsequent allocation operations. After memory is released or decommitted, you can never refer to the memory again. Any information that may have been in that memory is gone forever. Attempting to read from or write to a free page results in an access violation exception. If you need to keep information, do not decommit or free memory that contains the information.

The **VirtualFree** function can be used on an AWE region of memory, and it invalidates any physical page mappings in the region when freeing the address space. However, the physical page is not deleted, and the application can use them. The application must explicitly call **FreeUserPhysicalPages** to free the physical pages. When the process is terminated, all resources are cleaned up automatically.

To delete an enclave when you finish using it, specify the following values:

- The base address of the enclave for the *lpAddress* parameter.
- 0 for the *dwSize* parameter.
- **MEM\_RELEASE** for the *dwFreeType* parameter. The **MEM\_DECOMMIT** value is not supported for enclaves.

## Examples

For an example, see [Reserving and Committing Memory](#).

## Requirements

<b>Minimum supported client</b>	Windows XP [desktop apps   Windows Store apps]
<b>Minimum supported server</b>	Windows Server 2003 [desktop apps   Windows Store apps]
<b>Header</b>	WinBase.h (include Windows.h)
<b>Library</b>	Kernel32.lib
<b>DLL</b>	Kernel32.dll

## See also

[Memory Management Functions](#)

[Virtual Memory Functions](#)

[VirtualFreeEx](#)

© 2016 Microsoft