**SetWindowsHookEx function**

Installs an application-defined hook procedure into a hook chain. You would install a hook procedure to monitor the system for certain types of events. These events are associated either with a specific thread or with all threads in the same desktop as the calling thread.

**Syntax**

C++

HHOOK WINAPI SetWindowsHookEx(

\_In\_ int       idHook,

\_In\_ HOOKPROC  lpfn,

\_In\_ HINSTANCE hMod,

\_In\_ DWORD     dwThreadId

);

**Parameters**

*idHook* [in]

Type: **int**

The type of hook procedure to be installed. This parameter can be one of the following values.

|  |  |
| --- | --- |
| **Value** | **Meaning** |
| **WH\_CALLWNDPROC**  4 | Installs a hook procedure that monitors messages before the system sends them to the destination window procedure. For more information, see the [**CallWndProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644975%28v=vs.85%29.aspx) hook procedure. |
| **WH\_CALLWNDPROCRET**  12 | Installs a hook procedure that monitors messages after they have been processed by the destination window procedure. For more information, see the [**CallWndRetProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644976%28v=vs.85%29.aspx) hook procedure. |
| **WH\_CBT**  5 | Installs a hook procedure that receives notifications useful to a CBT application. For more information, see the [**CBTProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644977%28v=vs.85%29.aspx) hook procedure. |
| **WH\_DEBUG**  9 | Installs a hook procedure useful for debugging other hook procedures. For more information, see the [**DebugProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644978%28v=vs.85%29.aspx) hook procedure. |
| **WH\_FOREGROUNDIDLE**  11 | Installs a hook procedure that will be called when the application's foreground thread is about to become idle. This hook is useful for performing low priority tasks during idle time. For more information, see the [**ForegroundIdleProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644980%28v=vs.85%29.aspx) hook procedure. |
| **WH\_GETMESSAGE**  3 | Installs a hook procedure that monitors messages posted to a message queue. For more information, see the [**GetMsgProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644981%28v=vs.85%29.aspx) hook procedure. |
| **WH\_JOURNALPLAYBACK**  1 | Installs a hook procedure that posts messages previously recorded by a [**WH\_JOURNALRECORD**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644959%28v=vs.85%29.aspx#wh_journalrecordhook) hook procedure. For more information, see the [**JournalPlaybackProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644982%28v=vs.85%29.aspx) hook procedure. |
| **WH\_JOURNALRECORD**  0 | Installs a hook procedure that records input messages posted to the system message queue. This hook is useful for recording macros. For more information, see the [**JournalRecordProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644983%28v=vs.85%29.aspx) hook procedure. |
| **WH\_KEYBOARD**  2 | Installs a hook procedure that monitors keystroke messages. For more information, see the [**KeyboardProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644984%28v=vs.85%29.aspx) hook procedure. |
| **WH\_KEYBOARD\_LL**  13 | Installs a hook procedure that monitors low-level keyboard input events. For more information, see the [**LowLevelKeyboardProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644985%28v=vs.85%29.aspx) hook procedure. |
| **WH\_MOUSE**  7 | Installs a hook procedure that monitors mouse messages. For more information, see the [**MouseProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644988%28v=vs.85%29.aspx) hook procedure. |
| **WH\_MOUSE\_LL**  14 | Installs a hook procedure that monitors low-level mouse input events. For more information, see the [**LowLevelMouseProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644986%28v=vs.85%29.aspx) hook procedure. |
| **WH\_MSGFILTER**  -1 | Installs a hook procedure that monitors messages generated as a result of an input event in a dialog box, message box, menu, or scroll bar. For more information, see the [**MessageProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644987%28v=vs.85%29.aspx) hook procedure. |
| **WH\_SHELL**  10 | Installs a hook procedure that receives notifications useful to shell applications. For more information, see the [**ShellProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644991%28v=vs.85%29.aspx) hook procedure. |
| **WH\_SYSMSGFILTER**  6 | Installs a hook procedure that monitors messages generated as a result of an input event in a dialog box, message box, menu, or scroll bar. The hook procedure monitors these messages for all applications in the same desktop as the calling thread. For more information, see the [**SysMsgProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644992%28v=vs.85%29.aspx) hook procedure. |

*lpfn* [in]

Type: **HOOKPROC**

A pointer to the hook procedure. If the *dwThreadId* parameter is zero or specifies the identifier of a thread created by a different process, the *lpfn* parameter must point to a hook procedure in a DLL. Otherwise, *lpfn* can point to a hook procedure in the code associated with the current process.

*hMod* [in]

Type: **HINSTANCE**

A handle to the DLL containing the hook procedure pointed to by the *lpfn* parameter. The *hMod* parameter must be set to **NULL** if the *dwThreadId* parameter specifies a thread created by the current process and if the hook procedure is within the code associated with the current process.

*dwThreadId* [in]

Type: **DWORD**

The identifier of the thread with which the hook procedure is to be associated. For desktop apps, if this parameter is zero, the hook procedure is associated with all existing threads running in the same desktop as the calling thread. For Windows Store apps, see the Remarks section.

**Return value**

Type:

**Type: HHOOK**

If the function succeeds, the return value is the handle to the hook procedure.

If the function fails, the return value is **NULL**. To get extended error information, call [**GetLastError**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms679360%28v=vs.85%29.aspx).

**Remarks**

**SetWindowsHookEx** can be used to inject a DLL into another process. A 32-bit DLL cannot be injected into a 64-bit process, and a 64-bit DLL cannot be injected into a 32-bit process. If an application requires the use of hooks in other processes, it is required that a 32-bit application call **SetWindowsHookEx** to inject a 32-bit DLL into 32-bit processes, and a 64-bit application call **SetWindowsHookEx** to inject a 64-bit DLL into 64-bit processes. The 32-bit and 64-bit DLLs must have different names.

Because hooks run in the context of an application, they must match the "bitness" of the application. If a 32-bit application installs a global hook on 64-bit Windows, the 32-bit hook is injected into each 32-bit process (the usual security boundaries apply). In a 64-bit process, the threads are still marked as "hooked." However, because a 32-bit application must run the hook code, the system executes the hook in the hooking app's context; specifically, on the thread that called **SetWindowsHookEx**. This means that the hooking application must continue to pump messages or it might block the normal functioning of the 64-bit processes.

If a 64-bit application installs a global hook on 64-bit Windows, the 64-bit hook is injected into each 64-bit process, while all 32-bit processes use a callback to the hooking application.

To hook all applications on the desktop of a 64-bit Windows installation, install a 32-bit global hook and a 64-bit global hook, each from appropriate processes, and be sure to keep pumping messages in the hooking application to avoid blocking normal functioning. If you already have a 32-bit global hooking application and it doesn't need to run in each application's context, you may not need to create a 64-bit version.

An error may occur if the *hMod* parameter is **NULL** and the *dwThreadId* parameter is zero or specifies the identifier of a thread created by another process.

Calling the [**CallNextHookEx**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644974%28v=vs.85%29.aspx) function to chain to the next hook procedure is optional, but it is highly recommended; otherwise, other applications that have installed hooks will not receive hook notifications and may behave incorrectly as a result. You should call **CallNextHookEx** unless you absolutely need to prevent the notification from being seen by other applications.

Before terminating, an application must call the [**UnhookWindowsHookEx**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644993%28v=vs.85%29.aspx) function to free system resources associated with the hook.

The scope of a hook depends on the hook type. Some hooks can be set only with global scope; others can also be set for only a specific thread, as shown in the following table.

|  |  |
| --- | --- |
| **Hook** | **Scope** |
| **WH\_CALLWNDPROC** | Thread or global |
| **WH\_CALLWNDPROCRET** | Thread or global |
| **WH\_CBT** | Thread or global |
| **WH\_DEBUG** | Thread or global |
| **WH\_FOREGROUNDIDLE** | Thread or global |
| **WH\_GETMESSAGE** | Thread or global |
| **WH\_JOURNALPLAYBACK** | Global only |
| **WH\_JOURNALRECORD** | Global only |
| **WH\_KEYBOARD** | Thread or global |
| **WH\_KEYBOARD\_LL** | Global only |
| **WH\_MOUSE** | Thread or global |
| **WH\_MOUSE\_LL** | Global only |
| **WH\_MSGFILTER** | Thread or global |
| **WH\_SHELL** | Thread or global |
| **WH\_SYSMSGFILTER** | Global only |

For a specified hook type, thread hooks are called first, then global hooks. Be aware that the WH\_MOUSE, WH\_KEYBOARD, WH\_JOURNAL\*, WH\_SHELL, and low-level hooks can be called on the thread that installed the hook rather than the thread processing the hook. For these hooks, it is possible that both the 32-bit and 64-bit hooks will be called if a 32-bit hook is ahead of a 64-bit hook in the hook chain.

The global hooks are a shared resource, and installing one affects all applications in the same desktop as the calling thread. All global hook functions must be in libraries. Global hooks should be restricted to special-purpose applications or to use as a development aid during application debugging. Libraries that no longer need a hook should remove its hook procedure.

**Windows Store app development** If dwThreadId is zero, then window hook DLLs are not loaded in-process for the Windows Store app processes and the Windows Runtime broker process unless they are installed by either UIAccess processes (accessibility tools). The notification is delivered on the installer's thread for these hooks:

* WH\_JOURNALPLAYBACK
* WH\_JOURNALRECORD
* WH\_KEYBOARD
* WH\_KEYBOARD\_LL
* WH\_MOUSE
* WH\_MOUSE\_LL

This behavior is similar to what happens when there is an architecture mismatch between the hook DLL and the target application process, for example, when the hook DLL is 32-bit and the application process 64-bit.

**Examples**

For an example, see [Installing and Releasing Hook Procedures](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644960%28v=vs.85%29.aspx#installing_releasing).

**Requirements**

|  |  |
| --- | --- |
| **Minimum supported client** | Windows 2000 Professional [desktop apps only] |
| **Minimum supported server** | Windows 2000 Server [desktop apps only] |
| **Header** | Winuser.h (include Windows.h) |
| **Library** | User32.lib |
| **DLL** | User32.dll |
| **Unicode and ANSI names** | **SetWindowsHookExW** (Unicode) and **SetWindowsHookExA** (ANSI) |

**See also**

**Reference**

[**CallNextHookEx**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644974%28v=vs.85%29.aspx)

[**CallWndProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644975%28v=vs.85%29.aspx)

[**CallWndRetProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644976%28v=vs.85%29.aspx)

[**CBTProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644977%28v=vs.85%29.aspx)

[**DebugProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644978%28v=vs.85%29.aspx)

[**ForegroundIdleProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644980%28v=vs.85%29.aspx)

[**GetMsgProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644981%28v=vs.85%29.aspx)

[**JournalPlaybackProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644982%28v=vs.85%29.aspx)

[**JournalRecordProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644983%28v=vs.85%29.aspx)

[**LowLevelKeyboardProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644985%28v=vs.85%29.aspx)

[**LowLevelMouseProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644986%28v=vs.85%29.aspx)

[**KeyboardProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644984%28v=vs.85%29.aspx)

[**MouseProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644988%28v=vs.85%29.aspx)

[**MessageProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644987%28v=vs.85%29.aspx)

[**ShellProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644991%28v=vs.85%29.aspx)

[**SysMsgProc**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644992%28v=vs.85%29.aspx)

[**UnhookWindowsHookEx**](https://msdn.microsoft.com/en-us/library/windows/desktop/ms644993%28v=vs.85%29.aspx)

**Conceptual**

[Hooks](https://msdn.microsoft.com/en-us/library/windows/desktop/ms632589%28v=vs.85%29.aspx)