SendInput Function

The **SendInput** function synthesizes keystrokes, mouse motions, and button clicks.

Syntax

```
UINT SendInput(
    UINT nInputs,
    LPINPUT pInputs,
    int cbSize
);
```

Parameters

nInputs
[in] Number of structures in the *pInputs* array.

pInputs
[in] Pointer to an array of INPUT [http://msdn.microsoft.com/en-us/library/ms646270(VS.85).aspx] structures. Each structure represents an event to be inserted into the keyboard or mouse input stream.

cbSize
[in] Specifies the size, in bytes, of an **INPUT** structure. If *cbSize* is not the size of an **INPUT** structure, the function fails.

Return Value

The function returns the number of events that it successfully inserted into the keyboard or mouse input stream. If the function returns zero, the input was already blocked by another thread. To get extended error information, call GetLastError [http://msdn.microsoft.com/en-us/library/cc428944.aspx] .

Microsoft Windows Vista. This function fails when it is blocked by User Interface Privilege Isolation (UIPI). Note that neither **GetLastError** nor the return value will indicate the failure was caused by UIPI blocking.

Remarks

Microsoft Windows Vista. This function is subject to UIPI. Applications are permitted to inject input only into applications that are at an equal or lesser integrity level.

The **SendInput** function inserts the events in the **INPUT** structures serially into the keyboard or mouse input stream. These events are not interspersed with other keyboard or mouse input events inserted either by the user (with the keyboard or mouse) or by calls to keybd_event [http://msdn.microsoft.com/en-us/library/ms646304(VS.85).aspx] , mouse_event [http://msdn.microsoft.com/en-us/library/ms646260(VS.85).aspx] , or other calls to **SendInput**.

This function does not reset the keyboard's current state. Any keys that are already pressed when the function is called might interfere with the events that this function generates. To avoid this problem, check the keyboard's state with the GetAsyncKeyState [http://msdn.microsoft.com/en-us/library/ms646293(VS.85).aspx] function and correct as necessary.

Function Information

Minimum DLL Version	user32.dll
Header	Declared in Winuser.h, include Windows.h
Import library	User32.lib
Minimum operating systems	Windows XP, Windows NT 4.0 Service Pack 3

See Also

INPUT Structure

The **INPUT** structure is used by SendInput [http://msdn.microsoft.com/en-us/library/ms646310(VS.85).aspx] to store information for synthesizing input events such as keystrokes, mouse movement, and mouse clicks.

Syntax

```
typedef struct tagINPUT {
  DWORD type;
  union {MOUSEINPUT mi;
         KEYBDINPUT ki;
         HARDWAREINPUT hi;
        };
  }INPUT, *PINPUT;
```

Members

type

Specifies the type of the input event. This member can be one of the following values.

`INPUT_MOUSE`

The event is a mouse event. Use the **mi** structure of the union.

`INPUT_KEYBOARD`

The event is a keyboard event. Use the **ki** structure of the union.

`INPUT_HARDWARE`

Windows 95/98/Me: The event is from input hardware other than a keyboard or mouse. Use the **hi** structure of the union.

mi

A MOUSEINPUT [http://msdn.microsoft.com/en-us/library/ms646273(VS.85).aspx] structure that contains information about a simulated mouse event.

ki

A KEYBDINPUT [http://msdn.microsoft.com/en-us/library/ms646271(VS.85).aspx] structure that contains information about a simulated keyboard event.

hi

Windows 95/98/Me: A HARDWAREINPUT [http://msdn.microsoft.com/en-us/library/ms646269(VS.85).aspx] structure that contains information about a simulated event from input hardware other than a keyboard or mouse.

Remarks

This structure contains information identical to that used in the parameter list of the keybd_event [http://msdn.microsoft.com/en-us/library/ms646304(VS.85).aspx] or mouse_event [http://msdn.microsoft.com/en-us/library/ms646260(VS.85).aspx] function.

Windows 2000/XP: INPUT_KEYBOARD supports nonkeyboard input methods, such as handwriting recognition or voice recognition, as if it were text input by using the KEYEVENTF_UNICODE flag. For more information, see the remarks section of **KEYBDINPUT**.

Structure Information

Header	Declared in Winuser.h, include Windows.h
Minimum operating systems	Windows 98, Windows NT 4.0 Service Pack 3

See Also

Keyboard Input [http://msdn.microsoft.com/en-us/library/ms645530(VS.85).aspx] , GetMessageExtraInfo [http://msdn.microsoft.com/en-us/library/ms644937(VS.85).aspx] , SendInput [http://msdn.microsoft.com/en-us/library/ms646310(VS.85).aspx] , keybd_event

KEYBDINPUT Structure

The **KEYBDINPUT** structure contains information about a simulated keyboard event.

Syntax

```
typedef struct tagKEYBDINPUT {
    WORD wVk;
    WORD wScan;
    DWORD dwFlags;
    DWORD time;
    ULONG_PTR dwExtraInfo;
} KEYBDINPUT, *PKEYBDINPUT;
```

Members

wVk
Specifies a virtual-key code. The code must be a value in the range 1 to 254. The Winuser.h header file provides macro definitions (VK_*) for each value. If the **dwFlags** member specifies KEYEVENTF_UNICODE, **wVk** must be 0.

wScan
Specifies a hardware scan code for the key. If **dwFlags** specifies KEYEVENTF_UNICODE, **wScan** specifies a Unicode character which is to be sent to the foreground application.

dwFlags
Specifies various aspects of a keystroke. This member can be certain combinations of the following values.

KEYEVENTF_EXTENDEDKEY
If specified, the scan code was preceded by a prefix byte that has the value 0xE0 (224).

KEYEVENTF_KEYUP
If specified, the key is being released. If not specified, the key is being pressed.

KEYEVENTF_SCANCODE
If specified, **wScan** identifies the key and **wVk** is ignored.

KEYEVENTF_UNICODE
Windows 2000/XP: If specified, the system synthesizes a VK_PACKET keystroke. The **wVk** parameter must be zero. This flag can only be combined with the KEYEVENTF_KEYUP flag. For more information, see the Remarks section.

time
Time stamp for the event, in milliseconds. If this parameter is zero, the system will provide its own time stamp.

dwExtraInfo
Specifies an additional value associated with the keystroke. Use the GetMessageExtraInfo [http://msdn.microsoft.com/en-us/library/ms644937(VS.85).aspx] function to obtain this information.

Remarks

Windows 2000/XP: INPUT_KEYBOARD supports nonkeyboard-input methods—such as handwriting recognition or voice recognition—as if it were text input by using the KEYEVENTF_UNICODE flag. If KEYEVENTF_UNICODE is specified, SendInput [http://msdn.microsoft.com/en-us/library/ms646310(VS.85).aspx] sends a WM_KEYDOWN [http://msdn.microsoft.com/en-us/library/ms646280(VS.85).aspx] or WM_KEYUP [http://msdn.microsoft.com/en-us/library/ms646281(VS.85).aspx] message to the foreground thread's message queue with *wParam* equal to VK_PACKET. Once GetMessage [http://msdn.microsoft.com/en-us/library/ms644936(VS.85).aspx] or PeekMessage [http://msdn.microsoft.com/en-us/library/ms644943(VS.85).aspx] obtains this message, passing the message to TranslateMessage [http://msdn.microsoft.com/en-us/library/ms644955(VS.85).aspx] posts a WM_CHAR [http://msdn.microsoft.com/en-us/library/ms646276(VS.85).aspx] message with the Unicode character originally specified by **wScan**. This Unicode character will automatically be converted to the appropriate ANSI value if it is posted to an ANSI window.

Windows 2000/XP: Set the KEYEVENTF_SCANCODE flag to define keyboard input in terms of the scan code.

This is useful to simulate a physical keystroke regardless of which keyboard is currently being used. The virtual key value of a key may alter depending on the current keyboard layout or what other keys were pressed, but the scan code will always be the same.

Structure Information

Header	Declared in Winuser.h, include Windows.h
Minimum operating systems	Windows 98, Windows NT 4.0 Service Pack 3

See Also

Keyboard Input [http://msdn.microsoft.com/en-us/library/ms645530(VS.85).aspx] , GetMessageExtraInfo [http://msdn.microsoft.com/en-us/library/ms644937(VS.85).aspx] , INPUT [http://msdn.microsoft.com/en-us/library/ms646270(VS.85).aspx] , SendInput [http://msdn.microsoft.com/en-us/library/ms646310(VS.85).aspx]

Tags:

Community Content

MOUSEINPUT Structure

The **MOUSEINPUT** structure contains information about a simulated mouse event.

Syntax

```
typedef struct tagMOUSEINPUT {
    LONG dx;
    LONG dy;
    DWORD mouseData;
    DWORD dwFlags;
    DWORD time;
    ULONG_PTR dwExtraInfo;
} MOUSEINPUT, *PMOUSEINPUT;
```

Members

dx
Specifies the absolute position of the mouse, or the amount of motion since the last mouse event was generated, depending on the value of the **dwFlags** member. Absolute data is specified as the x coordinate of the mouse; relative data is specified as the number of pixels moved.

dy
Specifies the absolute position of the mouse, or the amount of motion since the last mouse event was generated, depending on the value of the **dwFlags** member. Absolute data is specified as the y coordinate of the mouse; relative data is specified as the number of pixels moved.

mouseData
If **dwFlags** contains MOUSEEVENTF_WHEEL, then **mouseData** specifies the amount of wheel movement. A positive value indicates that the wheel was rotated forward, away from the user; a negative value indicates that the wheel was rotated backward, toward the user. One wheel click is defined as WHEEL_DELTA, which is 120.

Windows Vista: If *dwFlags* contains MOUSEEVENTF_HWHEEL, then *dwData* specifies the amount of wheel movement. A positive value indicates that the wheel was rotated to the right; a negative value indicates that the wheel was rotated to the left. One wheel click is defined as WHEEL_DELTA, which is 120.

Windows 2000/XP: If**dwFlags** does not contain MOUSEEVENTF_WHEEL, MOUSEEVENTF_XDOWN, or MOUSEEVENTF_XUP, then **mouseData** should be zero.

If **dwFlags** contains MOUSEEVENTF_XDOWN or MOUSEEVENTF_XUP, then **mouseData** specifies which X buttons were pressed or released. This value may be any combination of the following flags.

XBUTTON1
Set if the first X button is pressed or released.

XBUTTON2
Set if the second X button is pressed or released.

dwFlags
A set of bit flags that specify various aspects of mouse motion and button clicks. The bits in this member can be any reasonable combination of the following values.

The bit flags that specify mouse button status are set to indicate changes in status, not ongoing conditions. For example, if the left mouse button is pressed and held down, MOUSEEVENTF_LEFTDOWN is set when the left button is first pressed, but not for subsequent motions. Similarly, MOUSEEVENTF_LEFTUP is set only when the button is first released.

You cannot specify both the MOUSEEVENTF_WHEEL flag and either MOUSEEVENTF_XDOWN or MOUSEEVENTF_XUP flags simultaneously in the **dwFlags** parameter, because they both require use of the **mouseData** field.

MOUSEEVENTF_ABSOLUTE
Specifies that the **dx** and **dy** members contain normalized absolute coordinates. If the flag is not set, **dx**and **dy** contain relative data (the change in position since the last reported position). This flag can be set, or not set, regardless of what kind of mouse or other pointing device, if any, is connected to the system. For further information about relative mouse

motion, see the following Remarks section.

MOUSEEVENTF_MOVE

Specifies that movement occurred.

MOUSEEVENTF_MOVE_NOCOALESCE

Windows Vista: Specifies that WM_MOUSEMOVE [http://msdn.microsoft.com/en-us/library/ms645616(VS.85).aspx] messages will not be coalesced. The default behavior is to coalesce **WM_MOUSEMOVE** messages.

MOUSEEVENTF_LEFTDOWN

Specifies that the left button was pressed.

MOUSEEVENTF_LEFTUP

Specifies that the left button was released.

MOUSEEVENTF_RIGHTDOWN

Specifies that the right button was pressed.

MOUSEEVENTF_RIGHTUP

Specifies that the right button was released.

MOUSEEVENTF_MIDDLEDOWN

Specifies that the middle button was pressed.

MOUSEEVENTF_MIDDLEUP

Specifies that the middle button was released.

MOUSEEVENTF_VIRTUALDESK

Windows 2000/XP: Maps coordinates to the entire desktop. Must be used with MOUSEEVENTF_ABSOLUTE

.

MOUSEEVENTF_WHEEL

Windows NT/2000/XP: Specifies that the wheel was moved, if the mouse has a wheel. The amount of movement is specified in **mouseData**.

MOUSEEVENTF_HWHEEL

Windows Vista: Specifies that the wheel was moved horizontally, if the mouse has a wheel. The amount of movement is specified in **mouseData**.

MOUSEEVENTF_XDOWN

Windows 2000/XP: Specifies that an X button was pressed.

MOUSEEVENTF_XUP

Windows 2000/XP: Specifies that an X button was released.

time

Time stamp for the event, in milliseconds. If this parameter is 0, the system will provide its own time stamp.

dwExtraInfo

Specifies an additional value associated with the mouse event. An application calls GetMessageExtraInfo [http://msdn.microsoft.com/en-us/library/ms644937(VS.85).aspx] to obtain this extra information.

Remarks

If the mouse has moved, indicated by MOUSEEVENTF_MOVE, **dx**and **dy** specify information about that movement. The information is specified as absolute or relative integer values.

If MOUSEEVENTF_ABSOLUTE value is specified, **dx** and **dy** contain normalized absolute coordinates between 0 and 65,535. The event procedure maps these coordinates onto the display surface. Coordinate (0,0) maps onto the upper-left corner of the display surface; coordinate (65535,65535) maps onto the lower-right corner. In a multimonitor system, the coordinates map to the primary monitor.

Windows 2000/XP: If MOUSEEVENTF_VIRTUALDESK is specified, the coordinates map to the entire virtual desktop.

If the MOUSEEVENTF_ABSOLUTE value is not specified, **dx**and **dy** specify movement relative to the previous mouse event (the last reported position). Positive values mean the mouse moved right (or down); negative values mean the mouse moved left (or up).

Relative mouse motion is subject to the effects of the mouse speed and the two-mouse threshold values. A user sets these three values with the **Pointer Speed** slider of the Control Panel's **Mouse Properties** sheet. You can obtain and set these values using the SystemParametersInfo [http://msdn.microsoft.com/en-

us/library/cc429946.aspx] function.

The system applies two tests to the specified relative mouse movement. If the specified distance along either the x or y axis is greater than the first mouse threshold value, and the mouse speed is not zero, the system doubles the distance. If the specified distance along either the x or y axis is greater than the second mouse threshold value, and the mouse speed is equal to two, the system doubles the distance that resulted from applying the first threshold test. It is thus possible for the system to multiply specified relative mouse movement along the x or y axis by up to four times.

Structure Information

Header	Declared in Winuser.h, include Windows.h
Minimum operating systems	Windows 98, Windows NT 4.0 Service Pack 3

See Also

Keyboard Input [http://msdn.microsoft.com/en-us/library/ms645530(VS.85).aspx] , GetMessageExtraInfo [http://msdn.microsoft.com/en-us/library/ms644937(VS.85).aspx] , INPUT [http://msdn.microsoft.com/en-us/library/ms646270(VS.85).aspx] , SendInput [http://msdn.microsoft.com/en-us/library/ms646310(VS.85).aspx] , SystemParametersInfo [http://msdn.microsoft.com/en-us/library/cc429946.aspx]

Tags:

Community Content

dx and dy are not pixels when using MOUSEEVENTF_ABSOLUTE Last Edit 12:53 AM by win32 sucks

It's not obvious unless you read this entire page, but dx and dy are not pixel values when using MOUSEEVENTF_ABSOLUTE. To convert from pixels, do something like this:

dx = x * (65335/ScreenWidth)
dy = y * (65335/ScreenHeight)

This isn't mentioned in the description of dx, dy, or MOUSEEVENTF_ABSOLUTE.

Tags: