SystemParametersInfoA function

In this article

- 08/17/2018
- 55 minutes to read

Retrieves or sets the value of one of the system-wide parameters. This function can also update the user profile while setting a parameter.

Syntax

```
BOOL SystemParametersInfoA(
  UINT  uiAction,
  UINT  uiParam,
  PVOID pvParam,
  UINT  fWinIni
);
```

Parameters

`uiAction`

Type: **UINT**

The system-wide parameter to be retrieved or set. The possible values are organized in the following tables of related parameters:

- Accessibility parameters
- Desktop parameters
- Icon parameters
- Input parameters
- Menu parameters
- Power parameters
- Screen saver parameters
- Time-out parameters
- UI effect parameters
- Window parameters

The following are the accessibility parameters.

Accessibility parameter	Meaning

SPI_GETACCESSTIMEOUT **0x003C**	Retrieves information about the time-out period associated with the accessibility features. The *pvParam* parameter must point to an ACCESSTIMEOUT structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(ACCESSTIMEOUT)`.
SPI_GETAUDIODESCRIPTION **0x0074**	Determines whether audio descriptions are enabled or disabled. The *pvParam* parameter is a pointer to an AUDIODESCRIPTION structure. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(AUDIODESCRIPTION)`. While it is possible for users who have visual impairments to hear the audio in video content, there is a lot of action in video that does not have corresponding audio. Specific audio description of what is happening in a video helps these users understand the content better. This flag enables you to determine whether audio descriptions have been enabled and in which language. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETCLIENTAREAANIMATION **0x1042**	Determines whether animations are enabled or disabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if animations are enabled, or **FALSE** otherwise. Display features such as flashing, blinking, flickering, and moving content can cause seizures in users with photo-sensitive epilepsy. This flag enables you to determine whether such animations have been disabled in the client area. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETDISABLEOVERLAPPEDCONTENT **0x1040**	Determines whether overlapped content is enabled or disabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** otherwise. Display features such as background images, textured backgrounds, water marks on documents, alpha blending, and transparency can reduce the contrast between the foreground and background, making it harder for users with low vision to see objects on the screen. This flag enables you to determine whether such overlapped content has been disabled. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETFILTERKEYS **0x0032**	Retrieves information about the FilterKeys accessibility feature. The *pvParam* parameter must point to a FILTERKEYS structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(FILTERKEYS)`.

SPI_GETFOCUSBORDERHEIGHT **0x2010**	Retrieves the height, in pixels, of the top and bottom edges of the focus rectangle drawn with DrawFocusRect. The *pvParam* parameter must point to a **UINT** value. **Windows 2000:** This parameter is not supported.
SPI_GETFOCUSBORDERWIDTH **0x200E**	Retrieves the width, in pixels, of the left and right edges of the focus rectangle drawn with DrawFocusRect. The *pvParam* parameter must point to a **UINT**. **Windows 2000:** This parameter is not supported.
SPI_GETHIGHCONTRAST **0x0042**	Retrieves information about the HighContrast accessibility feature. The *pvParam* parameter must point to a HIGHCONTRAST structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(HIGHCONTRAST)`. For a general discussion, see Remarks.
SPI_GETLOGICALDPIOVERRIDE **0x009E**	Retrieves a value that determines whether Windows 8 is displaying apps using the default scaling plateau for the hardware or going to the next higher plateau. This value is based on the current "Make everything on your screen bigger" setting, found in the **Ease of Access** section of **PC settings**: 1 is on, 0 is off. Apps can provide text and image resources for each of several scaling plateaus: 100%, 140%, and 180%. Providing separate resources optimized for a particular scale avoids distortion due to resizing. Windows 8 determines the appropriate scaling plateau based on a number of factors, including screen size and pixel density. When "Make everything on your screen bigger" is selected (SPI_GETLOGICALDPIOVERRIDE returns a value of 1), Windows uses resources from the next higher plateau. For example, in the case of hardware that Windows determines should use a scale of SCALE_100_PERCENT, this override causes Windows to use the SCALE_140_PERCENT scale value, assuming that it does not violate other constraints. **Note** You should not use this value. It might be altered or unavailable in subsequent versions of Windows. Instead, use the GetScaleFactorForDevice function or the DisplayProperties class to retrieve the preferred scaling factor. Desktop applications should use desktop logical DPI rather than scale factor. Desktop logical DPI can be retrieved through the GetDeviceCaps function.
SPI_GETMESSAGEDURATION **0x2016**	Retrieves the time that notification pop-ups should be displayed, in seconds. The *pvParam* parameter must point to a **ULONG** that receives the message duration. Users with visual impairments or cognitive conditions such as ADHD and dyslexia might need a longer time to read the text in notification messages. This flag enables you to retrieve the message duration. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

SPI_GETMOUSECLICKLOCK **0x101E**	Retrieves the state of the Mouse ClickLock feature. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** otherwise. For more information, see About Mouse Input. **Windows 2000:** This parameter is not supported.
SPI_GETMOUSECLICKLOCKTIME **0x2008**	Retrieves the time delay before the primary mouse button is locked. The *pvParam* parameter must point to **DWORD** that receives the time delay, in milliseconds. This is only enabled if **SPI_SETMOUSECLICKLOCK** is set to **TRUE**. For more information, see About Mouse Input. **Windows 2000:** This parameter is not supported.
SPI_GETMOUSEKEYS **0x0036**	Retrieves information about the MouseKeys accessibility feature. The *pvParam* parameter must point to a MOUSEKEYS structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(MOUSEKEYS)`.
SPI_GETMOUSESONAR **0x101C**	Retrieves the state of the Mouse Sonar feature. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled or **FALSE** otherwise. For more information, see About Mouse Input. **Windows 2000:** This parameter is not supported.
SPI_GETMOUSEVANISH **0x1020**	Retrieves the state of the Mouse Vanish feature. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled or **FALSE** otherwise. For more information, see About Mouse Input. **Windows 2000:** This parameter is not supported.
SPI_GETSCREENREADER **0x0046**	Determines whether a screen reviewer utility is running. A screen reviewer utility directs textual information to an output device, such as a speech synthesizer or Braille display. When this flag is set, an application should provide textual information in situations where it would otherwise present the information graphically. The *pvParam* parameter is a pointer to a **BOOL**variable that receives **TRUE** if a screen reviewer utility is running, or **FALSE** otherwise. **Note** Narrator, the screen reader that is included with Windows, does not set the **SPI_SETSCREENREADER** or **SPI_GETSCREENREADER** flags.
SPI_GETSERIALKEYS **0x003E**	This parameter is not supported. **Windows Server 2003 and Windows XP/2000:** The user should control this setting through the Control Panel.

SPI_GETSHOWSOUNDS **0x0038**	Determines whether the Show Sounds accessibility flag is on or off. If it is on, the user requires an application to present information visually in situations where it would otherwise present the information only in audible form. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the feature is on, or **FALSE** if it is off. Using this value is equivalent to calling GetSystemMetrics with **SM_SHOWSOUNDS**. That is the recommended call.
SPI_GETSOUNDSENTRY **0x0040**	Retrieves information about the SoundSentry accessibility feature. The *pvParam* parameter must point to a SOUNDSENTRY structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(SOUNDSENTRY)`.
SPI_GETSTICKYKEYS **0x003A**	Retrieves information about the StickyKeys accessibility feature. The *pvParam* parameter must point to a STICKYKEYS structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(STICKYKEYS)`.
SPI_GETTOGGLEKEYS **0x0034**	Retrieves information about the ToggleKeys accessibility feature. The *pvParam* parameter must point to a TOGGLEKEYS structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(TOGGLEKEYS)`.
SPI_SETACCESSTIMEOUT **0x003D**	Sets the time-out period associated with the accessibility features. The *pvParam* parameter must point to an ACCESSTIMEOUT structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(ACCESSTIMEOUT)`.
SPI_SETAUDIODESCRIPTION **0x0075**	Turns the audio descriptions feature on or off. The *pvParam* parameter is a pointer to an AUDIODESCRIPTION structure. While it is possible for users who are visually impaired to hear the audio in video content, there is a lot of action in video that does not have corresponding audio. Specific audio description of what is happening in a video helps these users understand the content better. This flag enables you to enable or disable audio descriptions in the languages they are provided in. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

SPI_SETCLIENTAREAANIMATION **0x1043**	Turns client area animations on or off. The *pvParam* parameter is a **BOOL** variable. Set *pvParam* to **TRUE** to enable animations and other transient effects in the client area, or **FALSE** to disable them. Display features such as flashing, blinking, flickering, and moving content can cause seizures in users with photo-sensitive epilepsy. This flag enables you to enable or disable all such animations. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETDISABLEOVERLAPPEDCONTENT **0x1041**	Turns overlapped content (such as background images and watermarks) on or off. The *pvParam* parameter is a **BOOL** variable. Set *pvParam* to **TRUE** to disable overlapped content, or **FALSE** to enable overlapped content. Display features such as background images, textured backgrounds, water marks on documents, alpha blending, and transparency can reduce the contrast between the foreground and background, making it harder for users with low vision to see objects on the screen. This flag enables you to enable or disable all such overlapped content. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETFILTERKEYS **0x0033**	Sets the parameters of the FilterKeys accessibility feature. The *pvParam* parameter must point to a FILTERKEYS structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(FILTERKEYS)`.
SPI_SETFOCUSBORDERHEIGHT **0x2011**	Sets the height of the top and bottom edges of the focus rectangle drawn with DrawFocusRect to the value of the *pvParam* parameter. **Windows 2000:** This parameter is not supported.
SPI_SETFOCUSBORDERWIDTH **0x200F**	Sets the height of the left and right edges of the focus rectangle drawn with DrawFocusRect to the value of the *pvParam* parameter. **Windows 2000:** This parameter is not supported.
SPI_SETHIGHCONTRAST **0x0043**	Sets the parameters of the HighContrast accessibility feature. The *pvParam* parameter must point to a HIGHCONTRAST structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(HIGHCONTRAST)`.
SPI_SETLOGICALDPIOVERRIDE **0x009F**	Do not use.

SPI_SETMESSAGEDURATION **0x2017**	Sets the time that notification pop-ups should be displayed, in seconds. The *pvParam* parameter specifies the message duration. Users with visual impairments or cognitive conditions such as ADHD and dyslexia might need a longer time to read the text in notification messages. This flag enables you to set the message duration. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETMOUSECLICKLOCK **0x101F**	Turns the Mouse ClickLock accessibility feature on or off. This feature temporarily locks down the primary mouse button when that button is clicked and held down for the time specified by **SPI_SETMOUSECLICKLOCKTIME**. The *pvParam* parameter specifies **TRUE** for on, or **FALSE** for off. The default is off. For more information, see Remarks and AboutMouse Input. **Windows 2000:** This parameter is not supported.
SPI_SETMOUSECLICKLOCKTIME **0x2009**	Adjusts the time delay before the primary mouse button is locked. The *uiParam* parameter should be set to 0. The *pvParam* parameter points to a **DWORD** that specifies the time delay in milliseconds. For example, specify 1000 for a 1 second delay. The default is 1200. For more information, see About Mouse Input. **Windows 2000:** This parameter is not supported.
SPI_SETMOUSEKEYS **0x0037**	Sets the parameters of the MouseKeys accessibility feature. The *pvParam* parameter must point to a MOUSEKEYS structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(MOUSEKEYS)`.
SPI_SETMOUSESONAR **0x101D**	Turns the Sonar accessibility feature on or off. This feature briefly shows several concentric circles around the mouse pointer when the user presses and releases the CTRL key. The *pvParam* parameter specifies **TRUE** for on and **FALSE** for off. The default is off. For more information, see About Mouse Input. **Windows 2000:** This parameter is not supported.
SPI_SETMOUSEVANISH **0x1021**	Turns the Vanish feature on or off. This feature hides the mouse pointer when the user types; the pointer reappears when the user moves the mouse. The *pvParam* parameter specifies **TRUE** for on and **FALSE** for off. The default is off. For more information, see About Mouse Input. **Windows 2000:** This parameter is not supported.
SPI_SETSCREENREADER **0x0047**	Determines whether a screen review utility is running. The *uiParam* parameter specifies **TRUE** for on, or **FALSE** for off. **Note** Narrator, the screen reader that is included with Windows, does not set the **SPI_SETSCREENREADER** or **SPI_GETSCREENREADER** flags.

SPI_SETSERIALKEYS **0x003F**	This parameter is not supported. **Windows Server 2003 and Windows XP/2000:** The user should control this setting through the Control Panel.
SPI_SETSHOWSOUNDS **0x0039**	Turns the ShowSounds accessibility feature on or off. The *uiParam* parameter specifies **TRUE** for on, or **FALSE** for off.
SPI_SETSOUNDSENTRY **0x0041**	Sets the parameters of the **SoundSentry** accessibility feature. The *pvParam* parameter must point to a SOUNDSENTRY structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(SOUNDSENTRY)`.
SPI_SETSTICKYKEYS **0x003B**	Sets the parameters of the StickyKeys accessibility feature. The *pvParam* parameter must point to a STICKYKEYS structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(STICKYKEYS)`.
SPI_SETTOGGLEKEYS **0x0035**	Sets the parameters of the ToggleKeys accessibility feature. The *pvParam* parameter must point to a TOGGLEKEYS structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(TOGGLEKEYS)`.

The following are the desktop parameters.

Desktop parameter	Meaning
SPI_GETCLEARTYPE **0x1048**	Determines whether ClearType is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if ClearType is enabled, or **FALSE** otherwise. ClearType is a software technology that improves the readability of text on liquid crystal display (LCD) monitors. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETDESKWALLPAPER **0x0073**	Retrieves the full path of the bitmap file for the desktop wallpaper. The *pvParam* parameter must point to a buffer to receive the null-terminated path string. Set the *uiParam* parameter to the size, in characters, of the *pvParam* buffer. The returned string will not exceed **MAX_PATH** characters. If there is no desktop wallpaper, the returned string is empty.
SPI_GETDROPSHADOW **0x1024**	Determines whether the drop shadow effect is enabled. The *pvParam* parameter must point to a **BOOL** variable that returns **TRUE** if enabled or **FALSE** if disabled. **Windows 2000:** This parameter is not supported.

SPI_GETFLATMENU **0x1022**	Determines whether native User menus have flat menu appearance. The *pvParam* parameter must point to a **BOOL** variable that returns **TRUE** if the flat menu appearance is set, or **FALSE** otherwise. **Windows 2000:** This parameter is not supported.
SPI_GETFONTSMOOTHING **0x004A**	Determines whether the font smoothing feature is enabled. This feature uses font antialiasing to make font curves appear smoother by painting pixels at different gray levels. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the feature is enabled, or **FALSE** if it is not.
SPI_GETFONTSMOOTHINGCONTRAST **0x200C**	Retrieves a contrast value that is used in ClearType smoothing. The *pvParam* parameter must point to a **UINT** that receives the information. Valid contrast values are from 1000 to 2200. The default value is 1400. **Windows 2000:** This parameter is not supported.
SPI_GETFONTSMOOTHINGORIENTATION **0x2012**	Retrieves the font smoothing orientation. The *pvParam* parameter must point to a **UINT** that receives the information. The possible values are **FE_FONTSMOOTHINGORIENTATIONBGR** (blue-green-red) and **FE_FONTSMOOTHINGORIENTATIONRGB** (red-green-blue). **Windows XP/2000:** This parameter is not supported until Windows XP with SP2.
SPI_GETFONTSMOOTHINGTYPE **0x200A**	Retrieves the type of font smoothing. The *pvParam* parameter must point to a **UINT** that receives the information. The possible values are **FE_FONTSMOOTHINGSTANDARD** and **FE_FONTSMOOTHINGCLEARTYPE**. **Windows 2000:** This parameter is not supported.
SPI_GETWORKAREA **0x0030**	Retrieves the size of the work area on the primary display monitor. The work area is the portion of the screen not obscured by the system taskbar or by application desktop toolbars. The *pvParam* parameter must point to a RECT structure that receives the coordinates of the work area, expressed in physical pixel size. Any DPI virtualization mode of the caller has no effect on this output. To get the work area of a monitor other than the primary display monitor, call the GetMonitorInfo function.
SPI_SETCLEARTYPE **0x1049**	Turns ClearType on or off. The *pvParam* parameter is a **BOOL** variable. Set *pvParam* to **TRUE** to enable ClearType, or **FALSE** to disable it. ClearType is a software technology that improves the readability of text on LCD monitors. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

SPI_SETCURSORS **0x0057**	Reloads the system cursors. Set the *uiParam* parameter to zero and the *pvParam* parameter to **NULL**.
SPI_SETDESKPATTERN **0x0015**	Sets the current desktop pattern by causing Windows to read the **Pattern=** setting from the WIN.INI file.
SPI_SETDESKWALLPAPER **0x0014**	**Note** When the **SPI_SETDESKWALLPAPER** flag is used, **SystemParametersInfo** returns **TRUE** unless there is an error (like when the specified file doesn't exist).
SPI_SETDROPSHADOW **0x1025**	Enables or disables the drop shadow effect. Set *pvParam* to **TRUE** to enable the drop shadow effect or **FALSE** to disable it. You must also have **CS_DROPSHADOW** in the window class style. **Windows 2000:** This parameter is not supported.
SPI_SETFLATMENU **0x1023**	Enables or disables flat menu appearance for native User menus. Set *pvParam* to **TRUE** to enable flat menu appearance or **FALSE** to disable it. When enabled, the menu bar uses **COLOR_MENUBAR** for the menubar background, **COLOR_MENU** for the menu-popup background, **COLOR_MENUHILIGHT** for the fill of the current menu selection, and **COLOR_HILIGHT** for the outline of the current menu selection. If disabled, menus are drawn using the same metrics and colors as in Windows 2000. **Windows 2000:** This parameter is not supported.
SPI_SETFONTSMOOTHING **0x004B**	Enables or disables the font smoothing feature, which uses font antialiasing to make font curves appear smoother by painting pixels at different gray levels. To enable the feature, set the *uiParam* parameter to **TRUE**. To disable the feature, set *uiParam* to **FALSE**.
SPI_SETFONTSMOOTHINGCONTRAST **0x200D**	Sets the contrast value used in ClearType smoothing. The *pvParam* parameter is the contrast value. Valid contrast values are from 1000 to 2200. The default value is 1400. **SPI_SETFONTSMOOTHINGTYPE** must also be set to **FE_FONTSMOOTHINGCLEARTYPE**. **Windows 2000:** This parameter is not supported.
SPI_SETFONTSMOOTHINGORIENTATION **0x2013**	Sets the font smoothing orientation. The *pvParam* parameter is either **FE_FONTSMOOTHINGORIENTATIONBGR** (blue-green-red) or **FE_FONTSMOOTHINGORIENTATIONRGB** (red-green-blue). **Windows XP/2000:** This parameter is not supported until Windows XP with SP2.

SPI_SETFONTSMOOTHINGTYPE **0x200B**	Sets the font smoothing type. The *pvParam* parameter is either **FE_FONTSMOOTHINGSTANDARD**, if standard anti-aliasing is used, or **FE_FONTSMOOTHINGCLEARTYPE**, if ClearType is used. The default is **FE_FONTSMOOTHINGSTANDARD**. **SPI_SETFONTSMOOTHING** must also be set. **Windows 2000:** This parameter is not supported.
SPI_SETWORKAREA **0x002F**	Sets the size of the work area. The work area is the portion of the screen not obscured by the system taskbar or by application desktop toolbars. The *pvParam* parameter is a pointer to a RECT structure that specifies the new work area rectangle, expressed in virtual screen coordinates. In a system with multiple display monitors, the function sets the work area of the monitor that contains the specified rectangle.

The following are the icon parameters.

Icon parameter	Meaning
SPI_GETICONMETRICS **0x002D**	Retrieves the metrics associated with icons. The *pvParam* parameter must point to an ICONMETRICS structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(ICONMETRICS)`.
SPI_GETICONTITLELOGFONT **0x001F**	Retrieves the logical font information for the current icon-title font. The *uiParam* parameter specifies the size of a LOGFONT structure, and the *pvParam* parameter must point to the **LOGFONT** structure to fill in.
SPI_GETICONTITLEWRAP **0x0019**	Determines whether icon-title wrapping is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** otherwise.
SPI_ICONHORIZONTALSPACING **0x000D**	Sets or retrieves the width, in pixels, of an icon cell. The system uses this rectangle to arrange icons in large icon view. To set this value, set *uiParam* to the new value and set *pvParam* to **NULL**. You cannot set this value to less than **SM_CXICON**. To retrieve this value, *pvParam* must point to an integer that receives the current value.
SPI_ICONVERTICALSPACING **0x0018**	Sets or retrieves the height, in pixels, of an icon cell. To set this value, set *uiParam* to the new value and set *pvParam* to **NULL**. You cannot set this value to less than **SM_CYICON**. To retrieve this value, *pvParam* must point to an integer that receives the current value.
SPI_SETICONMETRICS **0x002E**	Sets the metrics associated with icons. The *pvParam* parameter must point to an ICONMETRICS structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(ICONMETRICS)`.
SPI_SETICONS **0x0058**	Reloads the system icons. Set the *uiParam* parameter to zero and the *pvParam* parameter to **NULL**.

SPI_SETICONTITLELOGFONT **0x0022**	Sets the font that is used for icon titles. The *uiParam* parameter specifies the size of a LOGFONT structure, and the *pvParam* parameter must point to a **LOGFONT** structure.
SPI_SETICONTITLEWRAP **0x001A**	Turns icon-title wrapping on or off. The *uiParam* parameter specifies **TRUE** for on, or **FALSE** for off.

The following are the input parameters. They include parameters related to the keyboard, mouse, pen, input language, and the warning beeper.

Input parameter	Meaning
SPI_GETBEEP **0x0001**	Determines whether the warning beeper is on. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the beeper is on, or **FALSE** if it is off.
SPI_GETBLOCKSENDINPUTRESETS **0x1026**	Retrieves a **BOOL** indicating whether an application can reset the screensaver's timer by calling the SendInput function to simulate keyboard or mouse input. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the simulated input will be blocked, or **FALSE** otherwise.
SPI_GETCONTACTVISUALIZATION **0x2018**	Retrieves the current contact visualization setting. The *pvParam* parameter must point to a **ULONG** variable that receives the setting. For more information, see Contact Visualization.
SPI_GETDEFAULTINPUTLANG **0x0059**	Retrieves the input locale identifier for the system default input language. The *pvParam* parameter must point to an **HKL** variable that receives this value. For more information, see Languages, Locales, and Keyboard Layouts.
SPI_GETGESTUREVISUALIZATION **0x201A**	Retrieves the current gesture visualization setting. The *pvParam* parameter must point to a **ULONG** variable that receives the setting. For more information, see Gesture Visualization.
SPI_GETKEYBOARDCUES **0x100A**	Determines whether menu access keys are always underlined. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if menu access keys are always underlined, and **FALSE** if they are underlined only when the menu is activated by the keyboard.
SPI_GETKEYBOARDDELAY **0x0016**	Retrieves the keyboard repeat-delay setting, which is a value in the range from 0 (approximately 250 ms delay) through 3 (approximately 1 second delay). The actual delay associated with each value may vary depending on the hardware. The *pvParam* parameter must point to an integer variable that receives the setting.
SPI_GETKEYBOARDPREF **0x0044**	Determines whether the user relies on the keyboard instead of the mouse, and wants applications to display keyboard interfaces that would otherwise be hidden. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the user relies on the keyboard; or **FALSE** otherwise.

SPI_GETKEYBOARDSPEED **0x000A**	Retrieves the keyboard repeat-speed setting, which is a value in the range from 0 (approximately 2.5 repetitions per second) through 31 (approximately 30 repetitions per second). The actual repeat rates are hardware-dependent and may vary from a linear scale by as much as 20%. The *pvParam* parameter must point to a **DWORD** variable that receives the setting.
SPI_GETMOUSE **0x0003**	Retrieves the two mouse threshold values and the mouse acceleration. The *pvParam* parameter must point to an array of three integers that receives these values. See mouse_event for further information.
SPI_GETMOUSEHOVERHEIGHT **0x0064**	Retrieves the height, in pixels, of the rectangle within which the mouse pointer has to stay for TrackMouseEvent to generate a WM_MOUSEHOVER message. The *pvParam* parameter must point to a **UINT** variable that receives the height.
SPI_GETMOUSEHOVERTIME **0x0066**	Retrieves the time, in milliseconds, that the mouse pointer has to stay in the hover rectangle for TrackMouseEvent to generate a WM_MOUSEHOVER message. The *pvParam* parameter must point to a **UINT** variable that receives the time.
SPI_GETMOUSEHOVERWIDTH **0x0062**	Retrieves the width, in pixels, of the rectangle within which the mouse pointer has to stay for TrackMouseEvent to generate a WM_MOUSEHOVER message. The *pvParam* parameter must point to a **UINT** variable that receives the width.
SPI_GETMOUSESPEED **0x0070**	Retrieves the current mouse speed. The mouse speed determines how far the pointer will move based on the distance the mouse moves. The *pvParam* parameter must point to an integer that receives a value which ranges between 1 (slowest) and 20 (fastest). A value of 10 is the default. The value can be set by an end-user using the mouse control panel application or by an application using **SPI_SETMOUSESPEED**.
SPI_GETMOUSETRAILS **0x005E**	Determines whether the Mouse Trails feature is enabled. This feature improves the visibility of mouse cursor movements by briefly showing a trail of cursors and quickly erasing them. The *pvParam* parameter must point to an integer variable that receives a value. if the value is zero or 1, the feature is disabled. If the value is greater than 1, the feature is enabled and the value indicates the number of cursors drawn in the trail. The *uiParam* parameter is not used. **Windows 2000:** This parameter is not supported.

SPI_GETMOUSEWHEELROUTING **0x201C**	Retrieves the routing setting for wheel button input. The routing setting determines whether wheel button input is sent to the app with focus (foreground) or the app under the mouse cursor. The *pvParam* parameter must point to a **DWORD** variable that receives the routing option. If the value is zero or MOUSEWHEEL_ROUTING_FOCUS, mouse wheel input is delivered to the app with focus. If the value is 1 or MOUSEWHEEL_ROUTING_HYBRID (default), mouse wheel input is delivered to the app with focus (desktop apps) or the app under the mouse cursor (Windows Store apps). The *uiParam* parameter is not used.
SPI_GETPENVISUALIZATION **0x201E**	Retrieves the current pen gesture visualization setting. The *pvParam* parameter must point to a **ULONG** variable that receives the setting. For more information, see Pen Visualization.
SPI_GETSNAPTODEFBUTTON **0x005F**	Determines whether the snap-to-default-button feature is enabled. If enabled, the mouse cursor automatically moves to the default button, such as **OK** or **Apply**, of a dialog box. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the feature is on, or **FALSE** if it is off.
SPI_GETSYSTEMLANGUAGEBAR **0x1050**	**Starting with Windows 8:** Determines whether the system language bar is enabled or disabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the language bar is enabled, or **FALSE** otherwise.
SPI_GETTHREADLOCALINPUTSETTINGS **0x104E**	**Starting with Windows 8:** Determines whether the active input settings have Local (per-thread, **TRUE**) or Global (session, **FALSE**) scope. The *pvParam* parameter must point to a **BOOL** variable.
SPI_GETWHEELSCROLLCHARS **0x006C**	Retrieves the number of characters to scroll when the horizontal mouse wheel is moved. The *pvParam* parameter must point to a **UINT** variable that receives the number of lines. The default value is 3.
SPI_GETWHEELSCROLLLINES **0x0068**	Retrieves the number of lines to scroll when the vertical mouse wheel is moved. The *pvParam* parameter must point to a **UINT** variable that receives the number of lines. The default value is 3.
SPI_SETBEEP **0x0002**	Turns the warning beeper on or off. The *uiParam* parameter specifies **TRUE** for on, or **FALSE** for off.
SPI_SETBLOCKSENDINPUTRESETS **0x1027**	Determines whether an application can reset the screensaver's timer by calling the SendInput function to simulate keyboard or mouse input. The *uiParam* parameter specifies **TRUE** if the screensaver will not be deactivated by simulated input, or **FALSE** if the screensaver will be deactivated by simulated input.
SPI_SETCONTACTVISUALIZATION **0x2019**	Sets the current contact visualization setting. The *pvParam* parameter must point to a **ULONG** variable that identifies the setting. For more information, see Contact Visualization. **Note** If contact visualizations are disabled, gesture visualizations cannot be enabled.

SPI_SETDEFAULTINPUTLANG **0x005A**	Sets the default input language for the system shell and applications. The specified language must be displayable using the current system character set. The *pvParam* parameter must point to an **HKL** variable that contains the input locale identifier for the default language. For more information, see Languages, Locales, and Keyboard Layouts.
SPI_SETDOUBLECLICKTIME **0x0020**	Sets the double-click time for the mouse to the value of the *uiParam* parameter. If the *uiParam* value is greater than 5000 milliseconds, the system sets the double-click time to 5000 milliseconds. The double-click time is the maximum number of milliseconds that can occur between the first and second clicks of a double-click. You can also call the SetDoubleClickTime function to set the double-click time. To get the current double-click time, call the GetDoubleClickTime function.
SPI_SETDOUBLECLKHEIGHT **0x001E**	Sets the height of the double-click rectangle to the value of the *uiParam* parameter. The double-click rectangle is the rectangle within which the second click of a double-click must fall for it to be registered as a double-click. To retrieve the height of the double-click rectangle, call GetSystemMetrics with the **SM_CYDOUBLECLK** flag.
SPI_SETDOUBLECLKWIDTH **0x001D**	Sets the width of the double-click rectangle to the value of the *uiParam* parameter. The double-click rectangle is the rectangle within which the second click of a double-click must fall for it to be registered as a double-click. To retrieve the width of the double-click rectangle, call GetSystemMetrics with the **SM_CXDOUBLECLK** flag.
SPI_SETGESTUREVISUALIZATION **0x201B**	Sets the current gesture visualization setting. The *pvParam* parameter must point to a **ULONG** variable that identifies the setting. For more information, see Gesture Visualization. **Note** If contact visualizations are disabled, gesture visualizations cannot be enabled.
SPI_SETKEYBOARDCUES **0x100B**	Sets the underlining of menu access key letters. The *pvParam* parameter is a **BOOL** variable. Set *pvParam* to **TRUE** to always underline menu access keys, or **FALSE** to underline menu access keys only when the menu is activated from the keyboard.
SPI_SETKEYBOARDDELAY **0x0017**	Sets the keyboard repeat-delay setting. The *uiParam* parameter must specify 0, 1, 2, or 3, where zero sets the shortest delay approximately 250 ms) and 3 sets the longest delay (approximately 1 second). The actual delay associated with each value may vary depending on the hardware.
SPI_SETKEYBOARDPREF **0x0045**	Sets the keyboard preference. The *uiParam* parameter specifies **TRUE** if the user relies on the keyboard instead of the mouse, and wants applications to display keyboard interfaces that would otherwise be hidden; *uiParam* is **FALSE** otherwise.

SPI_SETKEYBOARDSPEED **0x000B**	Sets the keyboard repeat-speed setting. The *uiParam* parameter must specify a value in the range from 0 (approximately 2.5 repetitions per second) through 31 (approximately 30 repetitions per second). The actual repeat rates are hardware-dependent and may vary from a linear scale by as much as 20%. If *uiParam* is greater than 31, the parameter is set to 31.
SPI_SETLANGTOGGLE **0x005B**	Sets the hot key set for switching between input languages. The *uiParam* and *pvParam* parameters are not used. The value sets the shortcut keys in the keyboard property sheets by reading the registry again. The registry must be set before this flag is used. the path in the registry is **HKEY_CURRENT_USER\Keyboard Layout\Toggle** . Valid values are "1" = ALT+SHIFT, "2" = CTRL+SHIFT, and "3" = none.
SPI_SETMOUSE **0x0004**	Sets the two mouse threshold values and the mouse acceleration. The *pvParam* parameter must point to an array of three integers that specifies these values. See mouse_event for further information.
SPI_SETMOUSEBUTTONSWAP **0x0021**	Swaps or restores the meaning of the left and right mouse buttons. The *uiParam* parameter specifies **TRUE** to swap the meanings of the buttons, or **FALSE** to restore their original meanings. To retrieve the current setting, call GetSystemMetrics with the **SM_SWAPBUTTON** flag.
SPI_SETMOUSEHOVERHEIGHT **0x0065**	Sets the height, in pixels, of the rectangle within which the mouse pointer has to stay for TrackMouseEvent to generate a WM_MOUSEHOVER message. Set the *uiParam* parameter to the new height.
SPI_SETMOUSEHOVERTIME **0x0067**	Sets the time, in milliseconds, that the mouse pointer has to stay in the hover rectangle for TrackMouseEvent to generate a WM_MOUSEHOVER message. This is used only if you pass **HOVER_DEFAULT** in the *dwHoverTime* parameter in the call to **TrackMouseEvent**. Set the *uiParam*parameter to the new time. The time specified should be between **USER_TIMER_MAXIMUM** and **USER_TIMER_MINIMUM**. If *uiParam* is less than **USER_TIMER_MINIMUM**, the function will use **USER_TIMER_MINIMUM**. If *uiParam* is greater than **USER_TIMER_MAXIMUM**, the function will be **USER_TIMER_MAXIMUM**. `<b>Windows Server 2003 and Windows XP:  </b>The operating system does not enforce the use of <b>USER_TIMER_MAXIMUM</b> and <b>USER_TIMER_MINIMUM</b> until Windows Server 2003 with SP1 and Windows XP with SP2.`
SPI_SETMOUSEHOVERWIDTH **0x0063**	Sets the width, in pixels, of the rectangle within which the mouse pointer has to stay for TrackMouseEvent to generate a WM_MOUSEHOVER message. Set the *uiParam* parameter to the new width.
SPI_SETMOUSESPEED **0x0071**	Sets the current mouse speed. The *pvParam* parameter is an integer between 1 (slowest) and 20 (fastest). A value of 10 is the default. This value is typically set using the mouse control panel application.

SPI_SETMOUSETRAILS **0x005D**	Enables or disables the Mouse Trails feature, which improves the visibility of mouse cursor movements by briefly showing a trail of cursors and quickly erasing them. To disable the feature, set the *uiParam* parameter to zero or 1. To enable the feature, set *uiParam* to a value greater than 1 to indicate the number of cursors drawn in the trail. **Windows 2000:** This parameter is not supported.
SPI_SETMOUSEWHEELROUTING **0x201D**	Sets the routing setting for wheel button input. The routing setting determines whether wheel button input is sent to the app with focus (foreground) or the app under the mouse cursor. The *pvParam* parameter must point to a **DWORD** variable that receives the routing option. If the value is zero or MOUSEWHEEL_ROUTING_FOCUS, mouse wheel input is delivered to the app with focus. If the value is 1 or MOUSEWHEEL_ROUTING_HYBRID (default), mouse wheel input is delivered to the app with focus (desktop apps) or the app under the mouse cursor (Windows Store apps). Set the *uiParam* parameter to zero.
SPI_SETPENVISUALIZATION **0x201F**	Sets the current pen gesture visualization setting. The *pvParam* parameter must point to a **ULONG** variable that identifies the setting. For more information, see Pen Visualization.
SPI_SETSNAPTODEFBUTTON **0x0060**	Enables or disables the snap-to-default-button feature. If enabled, the mouse cursor automatically moves to the default button, such as **OK** or **Apply**, of a dialog box. Set the *uiParam* parameter to **TRUE** to enable the feature, or **FALSE** to disable it. Applications should use the ShowWindow function when displaying a dialog box so the dialog manager can position the mouse cursor.
SPI_SETSYSTEMLANGUAGEBAR **0x1051**	**Starting with Windows 8:** Turns the legacy language bar feature on or off. The *pvParam* parameter is a pointer to a **BOOL** variable. Set *pvParam* to **TRUE** to enable the legacy language bar, or **FALSE** to disable it. The flag is supported on Windows 8 where the legacy language bar is replaced by Input Switcher and therefore turned off by default. Turning the legacy language bar on is provided for compatibility reasons and has no effect on the Input Switcher.
SPI_SETTHREADLOCALINPUTSETTINGS **0x104F**	**Starting with Windows 8:** Determines whether the active input settings have Local (per-thread, **TRUE**) or Global (session, **FALSE**) scope. The *pvParam* parameter must point to a **BOOL** variable, casted by PVOID.
SPI_SETWHEELSCROLLCHARS **0x006D**	Sets the number of characters to scroll when the horizontal mouse wheel is moved. The number of characters is set from the *uiParam* parameter.

SPI_SETWHEELSCROLLLINES **0x0069**	Sets the number of lines to scroll when the vertical mouse wheel is moved. The number of lines is set from the *uiParam* parameter. The number of lines is the suggested number of lines to scroll when the mouse wheel is rolled without using modifier keys. If the number is 0, then no scrolling should occur. If the number of lines to scroll is greater than the number of lines viewable, and in particular if it is **WHEEL_PAGESCROLL** (#defined as **UINT_MAX**), the scroll operation should be interpreted as clicking once in the page down or page up regions of the scroll bar.

The following are the menu parameters.

Menu parameter	Meaning
SPI_GETMENUDROPALIGNMENT **0x001B**	Determines whether pop-up menus are left-aligned or right-aligned, relative to the corresponding menu-bar item. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if right-aligned, or **FALSE** otherwise.
SPI_GETMENUFADE **0x1012**	Determines whether menu fade animation is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** when fade animation is enabled and **FALSE** when it isdisabled. If fade animation is disabled, menus use slide animation. This flag is ignored unless menu animation is enabled, which you can do using the **SPI_SETMENUANIMATION** flag. For more information, see AnimateWindow.
SPI_GETMENUSHOWDELAY **0x006A**	Retrieves the time, in milliseconds, that the system waits before displaying a shortcut menu when the mouse cursor is over a submenu item. The *pvParam* parameter must point to a **DWORD** variable that receives the time of the delay.
SPI_SETMENUDROPALIGNMENT **0x001C**	Sets the alignment value of pop-up menus. The *uiParam* parameter specifies **TRUE** for right alignment, or **FALSE** for left alignment.
SPI_SETMENUFADE **0x1013**	Enables or disables menu fade animation. Set *pvParam* to **TRUE** to enable the menu fade effect or **FALSE** to disable it. If fade animation is disabled, menus use slide animation. he The menu fade effect is possible only if the system has a color depth of more than 256 colors. This flag is ignored unless **SPI_MENUANIMATION** is also set. For more information, see AnimateWindow.
SPI_SETMENUSHOWDELAY **0x006B**	Sets *uiParam* to the time, in milliseconds, that the system waits before displaying a shortcut menu when the mouse cursor is over a submenu item.

The following are the power parameters.

Beginning with Windows Server 2008 and Windows Vista, these power parameters are not supported. Instead, to determine the current display power state, an application should register for **GUID_MONITOR_POWER_STATE** notifications. To determine the current

display power down time-out, an application should register for notification of changes to the **GUID_VIDEO_POWERDOWN_TIMEOUT** power setting. For more information, see Registering for Power Events.

Windows Server 2003 and Windows XP/2000: To determine the current display power state, use the following power parameters.

Power parameter	Meaning
SPI_GETLOWPOWERACTIVE 0x0053	This parameter is not supported. **Windows Server 2003 and Windows XP/2000:** Determines whether the low-power phase of screen saving is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** if disabled. This flag is supported for 32-bit applications only.
SPI_GETLOWPOWERTIMEOUT 0x004F	This parameter is not supported. **Windows Server 2003 and Windows XP/2000:** Retrieves the time-out value for the low-power phase of screen saving. The *pvParam* parameter must point to an integer variable that receives the value. This flag is supported for 32-bit applications only.
SPI_GETPOWEROFFACTIVE 0x0054	This parameter is not supported. When the power-off phase of screen saving is enabled, the **GUID_VIDEO_POWERDOWN_TIMEOUT** power setting is greater than zero. **Windows Server 2003 and Windows XP/2000:** Determines whether the power-off phase of screen saving is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** if disabled. This flag is supported for 32-bit applications only.
SPI_GETPOWEROFFTIMEOUT 0x0050	This parameter is not supported. Instead, check the **GUID_VIDEO_POWERDOWN_TIMEOUT** power setting. **Windows Server 2003 and Windows XP/2000:** Retrieves the time-out value for the power-off phase of screen saving. The *pvParam* parameter must point to an integer variable that receives the value. This flag is supported for 32-bit applications only.
SPI_SETLOWPOWERACTIVE 0x0055	This parameter is not supported. **Windows Server 2003 and Windows XP/2000:** Activates or deactivates the low-power phase of screen saving. Set *uiParam* to 1 to activate, or zero to deactivate. The *pvParam* parameter must be **NULL**. This flag is supported for 32-bit applications only.
SPI_SETLOWPOWERTIMEOUT 0x0051	This parameter is not supported. **Windows Server 2003 and Windows XP/2000:** Sets the time-out value, in seconds, for the low-power phase of screen saving. The *uiParam* parameter specifies the new value. The *pvParam* parameter must be **NULL**. This flag is supported for 32-bit applications only.
SPI_SETPOWEROFFACTIVE 0x0056	This parameter is not supported. Instead, set the **GUID_VIDEO_POWERDOWN_TIMEOUT** power setting. **Windows Server 2003 and Windows XP/2000:** Activates or deactivates the power-off phase of screen saving. Set *uiParam* to 1 to activate, or zero to deactivate. The *pvParam* parameter must be **NULL**. This flag is supported for 32-bit applications only.

SPI_SETPOWEROFFTIMEOUT 0x0052	This parameter is not supported. Instead, set the **GUID_VIDEO_POWERDOWN_TIMEOUT** power setting to a time-out value. **Windows Server 2003 and Windows XP/2000:** Sets the time-out value, in seconds, for the power-off phase of screen saving. The *uiParam* parameter specifies the new value. The *pvParam* parameter must be **NULL**. This flag is supported for 32-bit applications only.

The following are the screen saver parameters.

Screen saver parameter	Meaning
SPI_GETSCREENSAVEACTIVE 0x0010	Determines whether screen saving is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if screen saving is enabled, or **FALSE** otherwise. **Windows 7, Windows Server 2008 R2 and Windows 2000:** The function returns **TRUE** even when screen saving is not enabled. For more information and a workaround, see KB318781.
SPI_GETSCREENSAVERRUNNING 0x0072	Determines whether a screen saver is currently running on the window station of the calling process. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if a screen saver is currently running, or **FALSE** otherwise. Note that only the interactive window station, WinSta0, can have a screen saver running.
SPI_GETSCREENSAVESECURE 0x0076	Determines whether the screen saver requires a password to display the Windows desktop. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the screen saver requires a password, or **FALSE** otherwise. The *uiParam* parameter is ignored. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETSCREENSAVETIMEOUT 0x000E	Retrieves the screen saver time-out value, in seconds. The *pvParam* parameter must point to an integer variable that receives the value.
SPI_SETSCREENSAVEACTIVE 0x0011	Sets the state of the screen saver. The *uiParam* parameter specifies **TRUE** to activate screen saving, or **FALSE** to deactivate it. If the machine has entered power saving mode or system lock state, an ERROR_OPERATION_IN_PROGRESS exception occurs.
SPI_SETSCREENSAVESECURE 0x0077	Sets whether the screen saver requires the user to enter a password to display the Windows desktop. The *uiParam* parameter is a **BOOL** variable. The *pvParam* parameter is ignored. Set *uiParam* to **TRUE** to require a password, or **FALSE** to not require a password. If the machine has entered power saving mode or system lock state, an ERROR_OPERATION_IN_PROGRESS exception occurs. **Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

SPI_SETSCREENSAVETIMEOUT 0x000F	Sets the screen saver time-out value to the value of the *uiParam* parameter. This value is the amount of time, in seconds, that the system must be idle before the screen saver activates. If the machine has entered power saving mode or system lock state, an ERROR_OPERATION_IN_PROGRESS exception occurs.

The following are the time-out parameters for applications and services.

Time-out parameter	Meaning
SPI_GETHUNGAPPTIMEOUT 0x0078	Retrieves the number of milliseconds that a thread can go without dispatching a message before the system considers it unresponsive. The *pvParam* parameter must point to an integer variable that receives the value. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETWAITTOKILLTIMEOUT 0x007A	Retrieves the number of milliseconds that the system waits before terminating an application that does not respond to a shutdown request. The *pvParam* parameter must point to an integer variable that receives the value. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETWAITTOKILLSERVICETIMEOUT 0x007C	Retrieves the number of milliseconds that the service control manager waits before terminating a service that does not respond to a shutdown request. The *pvParam* parameter must point to an integer variable that receives the value. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETHUNGAPPTIMEOUT 0x0079	Sets the hung application time-out to the value of the *uiParam* parameter. This value is the number of milliseconds that a thread can go without dispatching a message before the system considers it unresponsive. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETWAITTOKILLTIMEOUT 0x007B	Sets the application shutdown request time-out to the value of the *uiParam* parameter. This value is the number of milliseconds that the system waits before terminating an application that does not respond to a shutdown request. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

SPI_SETWAITTOKILLSERVICETIMEOUT 0x007D	Sets the service shutdown request time-out to the value of the *uiParam* parameter. This value is the number of milliseconds that the system waits before terminating a service that does not respond to a shutdown request. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

The following are the UI effects. The **SPI_SETUIEFFECTS** value is used to enable or disable all UI effects at once. This table contains the complete list of UI effect values.

UI effects parameter	Meaning
SPI_GETCOMBOBOXANIMATION 0x1004	Determines whether the slide-open effect for combo boxes is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** for enabled, or **FALSE** for disabled.
SPI_GETCURSORSHADOW 0x101A	Determines whether the cursor has a shadow around it. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the shadow is enabled, **FALSE** if it is disabled. This effect appears only if the system has a color depth of more than 256 colors.
SPI_GETGRADIENTCAPTIONS 0x1008	Determines whether the gradient effect for window title bars is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** for enabled, or **FALSE** for disabled. For more information about the gradient effect, see the GetSysColor function.
SPI_GETHOTTRACKING 0x100E	Determines whether hot tracking of user-interface elements, such as menu names on menu bars, is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** for enabled, or **FALSE** for disabled. Hot tracking means that when the cursor moves over an item, it is highlighted but not selected. You can query this value to decide whether to use hot tracking in the user interface of your application.
SPI_GETLISTBOXSMOOTHSCROLLING 0x1006	Determines whether the smooth-scrolling effect for list boxes is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** for enabled, or **FALSE** for disabled.
SPI_GETMENUANIMATION 0x1002	Determines whether the menu animation feature is enabled. This master switch must be on to enable menu animation effects. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if animation is enabled and **FALSE** if it is disabled. If animation is enabled, **SPI_GETMENUFADE** indicates whether menus use fade or slide animation.
SPI_GETMENUUNDERLINES 0x100A	Same as **SPI_GETKEYBOARDCUES**.

SPI_GETSELECTIONFADE **0x1014**	Determines whether the selection fade effect is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled or **FALSE** if disabled. The selection fade effect causes the menu item selected by the user to remain on the screen briefly while fading out after the menu is dismissed.
SPI_GETTOOLTIPANIMATION **0x1016**	Determines whether ToolTip animation is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled or **FALSE** if disabled. If ToolTip animation is enabled, **SPI_GETTOOLTIPFADE** indicates whether ToolTips use fade or slide animation.
SPI_GETTOOLTIPFADE **0x1018**	If **SPI_SETTOOLTIPANIMATION** is enabled, **SPI_GETTOOLTIPFADE** indicates whether ToolTip animation uses a fade effect or a slide effect. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** for fade animation or **FALSE** for slide animation. For more information on slide and fade effects, see AnimateWindow.
SPI_GETUIEFFECTS **0x103E**	Determines whether UI effects are enabled or disabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if all UI effects are enabled, or **FALSE** if they are disabled.
SPI_SETCOMBOBOXANIMATION **0x1005**	Enables or disables the slide-open effect for combo boxes. Set the *pvParam* parameter to **TRUE** to enable the gradient effect, or **FALSE** to disable it.
SPI_SETCURSORSHADOW **0x101B**	Enables or disables a shadow around the cursor. The *pvParam* parameter is a **BOOL** variable. Set *pvParam* to **TRUE** to enable the shadow or **FALSE** to disable the shadow. This effect appears only if the system has a color depth of more than 256 colors.
SPI_SETGRADIENTCAPTIONS **0x1009**	Enables or disables the gradient effect for window title bars. Set the *pvParam* parameter to **TRUE** to enable it, or **FALSE** to disable it. The gradient effect is possible only if the system has a color depth of more than 256 colors. For more information about the gradient effect, see the GetSysColor function.
SPI_SETHOTTRACKING **0x100F**	Enables or disables hot tracking of user-interface elements such as menu names on menu bars. Set the *pvParam* parameter to **TRUE** to enable it, or **FALSE** to disable it. Hot-tracking means that when the cursor moves over an item, it is highlighted but not selected.
SPI_SETLISTBOXSMOOTHSCROLLING **0x1007**	Enables or disables the smooth-scrolling effect for list boxes. Set the *pvParam* parameter to **TRUE** to enable the smooth-scrolling effect, or **FALSE** to disable it.
SPI_SETMENUANIMATION **0x1003**	Enables or disables menu animation. This master switch must be on for any menu animation to occur. The *pvParam* parameter is a **BOOL** variable; set *pvParam* to **TRUE** to enable animation and **FALSE** to disable animation. If animation is enabled, **SPI_GETMENUFADE** indicates whether menus use fade or slide animation.

SPI_SETMENUUNDERLINES **0x100B**	Same as **SPI_SETKEYBOARDCUES**.
SPI_SETSELECTIONFADE **0x1015**	Set *pvParam* to **TRUE** to enable the selection fade effect or **FALSE** to disable it. The selection fade effect causes the menu item selected by the user to remain on the screen briefly while fading out after the menu is dismissed. The selection fade effect is possible only if the system has a color depth of more than 256 colors.
SPI_SETTOOLTIPANIMATION **0x1017**	Set *pvParam* to **TRUE** to enable ToolTip animation or **FALSE** to disable it. If enabled, you can use **SPI_SETTOOLTIPFADE** to specify fade or slide animation.
SPI_SETTOOLTIPFADE **0x1019**	If the **SPI_SETTOOLTIPANIMATION** flag is enabled, use **SPI_SETTOOLTIPFADE** to indicate whether ToolTip animation uses a fade effect or a slide effect. Set *pvParam* to **TRUE** for fade animation or **FALSE** for slide animation. The tooltip fade effect is possible only if the system has a color depth of more than 256 colors. For more information on the slide and fade effects, see the AnimateWindowfunction.
SPI_SETUIEFFECTS **0x103F**	Enables or disables UI effects. Set the *pvParam* parameter to **TRUE** to enable all UI effects or **FALSE** to disable all UI effects.

The following are the window parameters.

Window parameter	Meaning
SPI_GETACTIVEWINDOWTRACKING **0x1000**	Determines whether active window tracking (activating the window the mouse is on) is on or off. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** for on, or **FALSE** for off.
SPI_GETACTIVEWNDTRKZORDER **0x100C**	Determines whether windows activated through active window tracking will be brought to the top. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** for on, or **FALSE** for off.
SPI_GETACTIVEWNDTRKTIMEOUT **0x2002**	Retrieves the active window tracking delay, in milliseconds. The *pvParam* parameter must point to a **DWORD** variable that receives the time.
SPI_GETANIMATION **0x0048**	Retrieves the animation effects associated with user actions. The *pvParam* parameter must point to an ANIMATIONINFO structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(ANIMATIONINFO)`.
SPI_GETBORDER **0x0005**	Retrieves the border multiplier factor that determines the width of a window's sizing border. The *pvParam*parameter must point to an integer variable that receives this value.
SPI_GETCARETWIDTH **0x2006**	Retrieves the caret width in edit controls, in pixels. The *pvParam* parameter must point to a **DWORD** variable that receives this value.

SPI_GETDOCKMOVING **0x0090**	Determines whether a window is docked when it is moved to the top, left, or right edges of a monitor or monitor array. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** otherwise. Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETDRAGFROMMAXIMIZE **0x008C**	Determines whether a maximized window is restored when its caption bar is dragged. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** otherwise. Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETDRAGFULLWINDOWS **0x0026**	Determines whether dragging of full windows is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** otherwise.
SPI_GETFOREGROUNDFLASHCOUNT **0x2004**	Retrieves the number of times SetForegroundWindow will flash the taskbar button when rejecting a foreground switch request. The *pvParam* parameter must point to a **DWORD** variable that receives the value.
SPI_GETFOREGROUNDLOCKTIMEOUT **0x2000**	Retrieves the amount of time following user input, in milliseconds, during which the system will not allow applications to force themselves into the foreground. The *pvParam* parameter must point to a **DWORD** variable that receives the time.
SPI_GETMINIMIZEDMETRICS **0x002B**	Retrieves the metrics associated with minimized windows. The *pvParam* parameter must point to a MINIMIZEDMETRICS structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(MINIMIZEDMETRICS)`.
SPI_GETMOUSEDOCKTHRESHOLD **0x007E**	Retrieves the threshold in pixels where docking behavior is triggered by using a mouse to drag a window to the edge of a monitor or monitor array. The default threshold is 1. The *pvParam* parameter must point to a **DWORD** variable that receives the value. Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

SPI_GETMOUSEDRAGOUTTHRESHOLD 0x0084	Retrieves the threshold in pixels where undocking behavior is triggered by using a mouse to drag a window from the edge of a monitor or a monitor array toward the center. The default threshold is 20. Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETMOUSESIDEMOVETHRESHOLD 0x0088	Retrieves the threshold in pixels from the top of a monitor or a monitor array where a vertically maximized window is restored when dragged with the mouse. The default threshold is 50. Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETNONCLIENTMETRICS 0x0029	Retrieves the metrics associated with the nonclient area of nonminimized windows. The *pvParam* parameter must point to a NONCLIENTMETRICS structure that receives the information. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(NONCLIENTMETRICS)`.
SPI_GETPENDOCKTHRESHOLD 0x0080	**Retrieves the threshold in pixels where docking behavior is triggered by using a pen to drag a window to the edge of a monitor or monitor array. The default is 30.** Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETPENDRAGOUTTHRESHOLD 0x0086	Retrieves the threshold in pixels where undocking behavior is triggered by using a pen to drag a window from the edge of a monitor or monitor array toward its center. The default threshold is 30. Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETPENSIDEMOVETHRESHOLD 0x008A	Retrieves the threshold in pixels from the top of a monitor or monitor array where a vertically maximized window is restored when dragged with the mouse. The default threshold is 50. Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

SPI_GETSHOWIMEUI **0x006E**	Determines whether the IME status window is visible (on a per-user basis). The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if the status window is visible, or **FALSE** if it is not.
SPI_GETSNAPSIZING **0x008E**	Determines whether a window is vertically maximized when it is sized to the top or bottom of a monitor or monitor array. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** otherwise. Use **SPI_GETWINARRANGING** to determine whether this behavior is enabled. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_GETWINARRANGING **0x0082**	Determines whether window arrangement is enabled. The *pvParam* parameter must point to a **BOOL** variable that receives **TRUE** if enabled, or **FALSE** otherwise. Window arrangement reduces the number of mouse, pen, or touch interactions needed to move and size top-level windows by simplifying the default behavior of a window when it is dragged or sized. The following parameters retrieve individual window arrangement settings: **SPI_GETDOCKMOVING** **SPI_GETMOUSEDOCKTHRESHOLD** **SPI_GETMOUSEDRAGOUTTHRESHOLD** **SPI_GETMOUSESIDEMOVETHRESHOLD** **SPI_GETPENDOCKTHRESHOLD** **SPI_GETPENDRAGOUTTHRESHOLD** **SPI_GETPENSIDEMOVETHRESHOLD** **SPI_GETSNAPSIZING** **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETACTIVEWINDOWTRACKING **0x1001**	Sets active window tracking (activating the window the mouse is on) either on or off. Set *pvParam* to **TRUE** for on or **FALSE** for off.
SPI_SETACTIVEWNDTRKZORDER **0x100D**	Determines whether or not windows activated through active window tracking should be brought to the top. Set *pvParam* to **TRUE** for on or **FALSE** for off.
SPI_SETACTIVEWNDTRKTIMEOUT **0x2003**	Sets the active window tracking delay. Set *pvParam* to the number of milliseconds to delay before activating the window under the mouse pointer.
SPI_SETANIMATION **0x0049**	Sets the animation effects associated with user actions. The *pvParam* parameter must point to an ANIMATIONINFO structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(ANIMATIONINFO)`.

SPI_SETBORDER **0x0006**	Sets the border multiplier factor that determines the width of a window's sizing border. The *uiParam* parameter specifies the new value.
SPI_SETCARETWIDTH **0x2007**	Sets the caret width in edit controls. Set *pvParam* to the desired width, in pixels. The default and minimum value is 1.
SPI_SETDOCKMOVING **0x0091**	Sets whether a window is docked when it is moved to the top, left, or right docking targets on a monitor or monitor array. Set *pvParam* to **TRUE** for on or **FALSE** for off. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETDRAGFROMMAXIMIZE **0x008D**	Sets whether a maximized window is restored when its caption bar is dragged. Set *pvParam* to **TRUE** for on or **FALSE** for off. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETDRAGFULLWINDOWS **0x0025**	Sets dragging of full windows either on or off. The *uiParam* parameter specifies **TRUE** for on, or **FALSE** for off.
SPI_SETDRAGHEIGHT **0x004D**	Sets the height, in pixels, of the rectangle used to detect the start of a drag operation. Set *uiParam* to the new value. To retrieve the drag height, call GetSystemMetrics with the **SM_CYDRAG** flag.
SPI_SETDRAGWIDTH **0x004C**	Sets the width, in pixels, of the rectangle used to detect the start of a drag operation. Set *uiParam* to the new value. To retrieve the drag width, call GetSystemMetrics with the **SM_CXDRAG** flag.
SPI_SETFOREGROUNDFLASHCOUNT **0x2005**	Sets the number of times SetForegroundWindow will flash the taskbar button when rejecting a foreground switch request. Set *pvParam* to the number of times to flash.
SPI_SETFOREGROUNDLOCKTIMEOUT **0x2001**	Sets the amount of time following user input, in milliseconds, during which the system does not allow applications to force themselves into the foreground. Set *pvParam* to the new time-out value. The calling thread must be able to change the foreground window, otherwise the call fails.
SPI_SETMINIMIZEDMETRICS **0x002C**	Sets the metrics associated with minimized windows. The *pvParam* parameter must point to a MINIMIZEDMETRICS structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(MINIMIZEDMETRICS)`.

SPI_SETMOUSEDOCKTHRESHOLD 0x007F	Sets the threshold in pixels where docking behavior is triggered by using a mouse to drag a window to the edge of a monitor or monitor array. The default threshold is 1. The *pvParam* parameter must point to a **DWORD** variable that contains the new value. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETMOUSEDRAGOUTTHRESHOLD 0x0085	Sets the threshold in pixels where undocking behavior is triggered by using a mouse to drag a window from the edge of a monitor or monitor array to its center. The default threshold is 20. The *pvParam* parameter must point to a **DWORD** variable that contains the new value. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETMOUSESIDEMOVETHRESHOLD 0x0089	Sets the threshold in pixels from the top of the monitor where a vertically maximized window is restored when dragged with the mouse. The default threshold is 50. The *pvParam* parameter must point to a **DWORD** variable that contains the new value. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETNONCLIENTMETRICS 0x002A	Sets the metrics associated with the nonclient area of nonminimized windows. The *pvParam* parameter must point to a NONCLIENTMETRICS structure that contains the new parameters. Set the **cbSize** member of this structure and the *uiParam* parameter to `sizeof(NONCLIENTMETRICS)`. Also, the **lfHeight** member of the LOGFONT structure must be a negative value.
SPI_SETPENDOCKTHRESHOLD 0x0081	Sets the threshold in pixels where docking behavior is triggered by using a pen to drag a window to the edge of a monitor or monitor array. The default threshold is 30. The *pvParam* parameter must point to a **DWORD** variable that contains the new value. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

SPI_SETPENDRAGOUTTHRESHOLD **0x0087**	Sets the threshold in pixels where undocking behavior is triggered by using a pen to drag a window from the edge of a monitor or monitor array to its center. The default threshold is 30. The *pvParam* parameter must point to a **DWORD** variable that contains the new value. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETPENSIDEMOVETHRESHOLD **0x008B**	Sets the threshold in pixels from the top of the monitor where a vertically maximized window is restored when dragged with a pen. The default threshold is 50. The *pvParam* parameter must point to a **DWORD** variable that contains the new value. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETSHOWIMEUI **0x006F**	Sets whether the IME status window is visible or not on a per-user basis. The *uiParam* parameter specifies **TRUE** for on or **FALSE** for off.
SPI_SETSNAPSIZING **0x008F**	Sets whether a window is vertically maximized when it is sized to the top or bottom of the monitor. Set *pvParam* to **TRUE** for on or **FALSE** for off. **SPI_GETWINARRANGING** must be **TRUE** to enable this behavior. **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.
SPI_SETWINARRANGING **0x0083**	Sets whether window arrangement is enabled. Set *pvParam* to **TRUE** for on or **FALSE** for off. Window arrangement reduces the number of mouse, pen, or touch interactions needed to move and size top-level windows by simplifying the default behavior of a window when it is dragged or sized. The following parameters set individual window arrangement settings: **SPI_SETDOCKMOVING** **SPI_SETMOUSEDOCKTHRESHOLD** **SPI_SETMOUSEDRAGOUTTHRESHOLD** **SPI_SETMOUSESIDEMOVETHRESHOLD** **SPI_SETPENDOCKTHRESHOLD** **SPI_SETPENDRAGOUTTHRESHOLD** **SPI_SETPENSIDEMOVETHRESHOLD** **SPI_SETSNAPSIZING** **Windows Server 2008, Windows Vista, Windows Server 2003 and Windows XP/2000:** This parameter is not supported.

`uiParam`

Type: **UINT**

A parameter whose usage and format depends on the system parameter being queried or set. For more information about system-wide parameters, see the *uiAction* parameter. If not otherwise indicated, you must specify zero for this parameter.

`pvParam`

Type: **PVOID**

A parameter whose usage and format depends on the system parameter being queried or set. For more information about system-wide parameters, see the *uiAction* parameter. If not otherwise indicated, you must specify **NULL** for this parameter. For information on the **PVOID** datatype, see Windows Data Types.

`fWinIni`

Type: **UINT**

If a system parameter is being set, specifies whether the user profile is to be updated, and if so, whether the WM_SETTINGCHANGE message is to be broadcast to all top-level windows to notify them of the change.

This parameter can be zero if you do not want to update the user profile or broadcast the WM_SETTINGCHANGE message, or it can be one or more of the following values.

Value	Meaning
SPIF_UPDATEINIFILE	Writes the new system-wide parameter setting to the user profile.
SPIF_SENDCHANGE	Broadcasts the WM_SETTINGCHANGE message after updating the user profile.
SPIF_SENDWININICHANGE	Same as **SPIF_SENDCHANGE**.

Return Value

Type: **Type: BOOL**

If the function succeeds, the return value is a nonzero value.

If the function fails, the return value is zero. To get extended error information, call GetLastError.

This function is intended for use with applications that allow the user to customize the environment.

A keyboard layout name should be derived from the hexadecimal value of the language identifier corresponding to the layout. For example, U.S. English has a language identifier of 0x0409, so the primary U.S. English layout is named "00000409". Variants of U.S.

English layout, such as the Dvorak layout, are named "00010409", "00020409" and so on. For a list of the primary language identifiers and sublanguage identifiers that make up a language identifier, see the **MAKELANGID** macro.

There is a difference between the High Contrast color scheme and the High Contrast Mode. The High Contrast color scheme changes the system colors to colors that have obvious contrast; you switch to this color scheme by using the Display Options in the control panel. The High Contrast Mode, which uses **SPI_GETHIGHCONTRAST** and **SPI_SETHIGHCONTRAST**, advises applications to modify their appearance for visually-impaired users. It involves such things as audible warning to users and customized color scheme (using the Accessibility Options in the control panel). For more information, see HIGHCONTRAST. For more information on general accessibility features, see Accessibility.

During the time that the primary button is held down to activate the Mouse ClickLock feature, the user can move the mouse. After the primary button is locked down, releasing the primary button does not result in a **WM_LBUTTONUP** message. Thus, it will appear to an application that the primary button is still down. Any subsequent button message releases the primary button, sending a **WM_LBUTTONUP** message to the application, thus the button can be unlocked programmatically or through the user clicking any button.

This API is not DPI aware, and should not be used if the calling thread is per-monitor DPI aware. For the DPI-aware version of this API, see SystemParametersInfoForDPI. For more information on DPI awareness, see the Windows High DPI documentation.

Examples

The following example uses **SystemParametersInfo** to double the mouse speed.

C++

```
#include <windows.h>
#include <stdio.h>
#pragma comment(lib, "user32.lib")

void main()
{
    BOOL fResult;
    int aMouseInfo[3];


    fResult = SystemParametersInfo(SPI_GETMOUSE,
                                   0,
                                   &aMouseInfo,
                                   0);


    if( fResult )
    {
        aMouseInfo[2] = 2 * aMouseInfo[2];


        SystemParametersInfo(SPI_SETMOUSE,
                             0,
                             aMouseInfo,
                             SPIF_SENDCHANGE);
    }
}
```

Requirements

Minimum supported client	Windows 2000 Professional [desktop apps only]
Minimum supported server	Windows 2000 Server [desktop apps only]
Target Platform	Windows
Header	winuser.h (include Windows.h)
Library	User32.lib
DLL	User32.dll

See Also

ACCESSTIMEOUT

ANIMATIONINFO

AUDIODESCRIPTION

FILTERKEYS

HIGHCONTRAST

ICONMETRICS

LOGFONT

MAKELANGID

MINIMIZEDMETRICS

MOUSEKEYS

NONCLIENTMETRICS

RECT

SERIALKEYS

SOUNDSENTRY

STICKYKEYS

SystemParametersInfoForDPI

TOGGLEKEYS

WM_SETTINGCHANGE

Windows Data Types

Note

The feedback system for this content will be changing soon. Old comments will not be carried over. If content within a comment thread is important to you, please save a copy. For more information on the upcoming change, we invite you to read our blog post.