Win32 Multilingual IME Application Programming Interface

Version 1.41

04-01-1999

This documentation contains the application programming interface reference for Input Method Editor (IME)development. The following functions are intended to be used by the IME.

IMM UI Functions

Following are the Input Method Manager (IMM) functions that can be accessed from the UI window. They are also used by applications to change IME status.

ImmGetCompositionWindow
ImmSetCompositionWindow
ImmGetCandidateWindow
ImmSetCandidateWindow
ImmGetCompositionString
ImmSetCompositionString
ImmGetCompositionFont
ImmSetCompositionFont
ImmGetNumCandidateList
ImmGetCandidateList
ImmGetGuideLine
ImmGetConversionStatus
ImmGetConversionList
ImmGetOpenStatus
ImmSetConversionStatus
ImmSetOpenStatus
ImmNotifyIME
ImmCreateSoftKeyboard
ImmDestroySoftkeyboard
ImmShowSoftKeyboard

Please refer to the Input Method Editor (IME) functions in the Platform SDK for information about these functions.

IMM Support Functions

The following topics contain IMM functions that support and are used by the IME.

ImmGenerateMessage

The IME uses the **ImmGenerateMessage** function to send messages to the **hWnd** of *hIMC*. The messages to be sent are stored in hMsgBuf of *hIMC*.

BOOL WINAPI
ImmGenerateMessage(
HIMC *hIMC*
)

Parameters

hIMC
Input context handle containing *hMsgBuf*.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

Comments

This is a general purpose function. Typically, an IME uses this function when it is notified about the context update through **ImmNotifyIME** from IMM. In this case, even if IME needs to provide messages to an application, there is no keystroke in the application's message queue.

An IME User Interface should not use this function when it only wants to update the UI appearance. The IME User Interface should have been updated when the IME is informed about the updated Input Context. It is recommended that you use this function from the IME only when the IME changes the Input Context without any keystroke given and needs to inform an application of the change.

ImmRequestMessage

The **ImmRequestMessage** function is used to send a **WM_IME_REQUEST** message to the application.

LRESULT WINAPI
ImmRequestMessage(
HIMC *hIMC***,**
WPARAM *wParam***,**
LPARAM *lParam*
)

Parameters

hIMC
Target input context handle.

wParam
wParam for the **WM_IME_REQUEST** message.

lParam
lParam for the **WM_IME_REQUEST** message.

Return Values

The return value is the return value of the **WM_IME_REQUEST** message.

Comments

This function is new for Windows® 98 and Window 2000, and is used by the IME to send a **WM_IME_REQUEST** message to the application. The IME may want to obtain some guidelines from the application in defining the position of the candidate or composition window. But in an IME fully aware (true in-line) application, the application usually does not set the composition window position. When the IME makes a request to the application, it receives the **WM_IME_REQUEST** message. The IME should make a request to the application by calling the **ImmRequestMessage** function and not by calling **SendMessage**

The following is a list of submessages that the IME can send to applications through the **ImmRequestMessage** function:

IMR_COMPOSITIONWINOW
IMR_CANDIDATEWINDOW
IMR_COMPOSITIONFONT
IMR_RECONVERTSTRING
IMR_CONFIRMRECONVERTSTRING
IMR_QUERYCHARPOSITION
IMR_DOCUMENTFEED

Please refer to the Input Method Editor (IME) functions in the Platform SDK for information about these messages.

HIMC and HIMCC Management Functions

The following topics contain the HIMC and HIMCC management functions.

ImmLockIMC

The **ImmLockIMC** function increases the lock count for the IMC. When the IME needs to see the **INPUTCONTEXT** structure, it calls this function to get the pointer of the **INPUTCONTEXT** structure.

LPINPUTCONTEXT WINAPI
ImmLockIMC(
HIMC *hIMC*
)

Parameters

hIMC
Input context handle.

Return Values

If the function is successful, it returns a pointer to the **INPUTCONTEXT** structure. Otherwise, it returns NULL.

ImmUnlockIMC

The **ImmUnlockIMC** function decrements the lock count for the IMC.

BOOL WINAPI
ImmUnlockIMC(
HIMC *hIMC*
)

Parameters

hIMC
Input context handle.

Return Values

If the lock count of the IMC is decremeted to zero, the return value is FALSE. Otherwise, the return value is TRUE.

ImmGetIMCLockCount

The **ImmGetIMCLockCount** is used to get the lock count of the IMC.

HIMCC WINAPI
ImmGetIMCLockCount(
HIMC *hIMC*
)

Parameters

hIMC
Input context handle

Return Values

If the function is successful, the return value is the lock count of the IMC. Otherwise, the return value is NULL.

ImmCreateIMCC

The **ImmCreateIMCC** function creates a new component as a member of the IMC.

HIMCC WINAPI
ImmCreateIMCC(
DWORD *dwSize*
)

Parameters

dwSize
Size of the new IMC component.

Return Values

If the function is successful, the return value is the IMC component handle (HIMCC). Otherwise, the return value is NULL.

Comments

The IMC component created by this function is initialized as zero.

ImmDestroyIMCC

The **ImmDestroyIMCC** function is used by the IME to destroy the IMC component that was created as a member of the IMC.

HIMCC WINAPI
ImmDestroyIMCC(
HIMCC *hIMCC*
)

Parameters

hIMCC
Handle of the IMC component.

Return Values

If the function is successful, the return value is NULL. Otherwise, the return value is equal to the HIMCC.

ImmLockIMCC

The **ImmLockIMCC** function is used by the IME to get the pointer for the IMC component that was created as a member of the IMC. The **ImmLockIMC** function increases the lock count for the IMCC.

LPVOID WINAPI
ImmLockIMCC(
HIMCC *hIMCC*
)

Parameters

hIMCC
Handle of the IMC component.

Return Values

If the function is successful, the return value is the pointer for the IMC component. Otherwise, the return value is NULL.

ImmUnlockIMCC

The **ImmUnlockIMC** function decrements the lock count for the IMCC.

BOOL WINAPI
ImmUnlockIMCC(
HIMCC *hIMCC*
)

Parameters

hIMCC
Handle of the IMC component.

Return Values

If the lock count of the IMCC is decremeted to zero, the return value is FALSE. Otherwise, the return value is TRUE.

ImmReSizeIMCC

The **ImmReSizeIMCC** function changes the size of the component.

```
HIMCC WINAPI
    ImmReSizeIMCC(
    HIMCC hIMCC,
    DWORD dwSize
   )
```

Parameters

hIMCC
Handle of the IMC component.

dwSize
New size of the IMC component.

Return Values

If the function is successful, the return value is the new HIMCC. Otherwise, the return value is NULL.

ImmGetIMCCSize

The **ImmGetIMCCLockCount** function is used to get the size of the IMCC.

```
DWORD WINAPI
    ImmGetIMCCSize(
    HIMCC hIMCC
   )
```

Parameters

hIMCC
Handle of the IMC component.

Return Values

Size of the IMCC.

ImmGetIMCCLockCount

The **ImmGetIMCCLockCount** function is used to get the lock count of the IMCC.

```
DWORD WINAPI
    ImmGetIMCCLockCount(
    HIMCC hIMCC
   )
```

Parameters

hIMCC
Handle of the IMC component.

Return Values

If the function is successful, the return value is the lock count of the IMCC. Otherwise, the return value is zero.

IME Hot Keys and Hot Key Functions

The IME hot key is used for changing the IME input mode and for switching the IME. The IME hot key used to switch directly to an IME is called a *direct switching hot key*.

The direct switching hot key ranges from IME_HOTKEY_DSWITCH_FIRST to IME_HOTKEY_DSWITCH_LAST. It is registered by an IME or Control Panel when the IME or an end user wants such a hot key. The IME hot key is effective in all IMEs, regardless which IME is active.

There are several predefined hot key functionalities in the IMM. The IMM itself provides the functionality (different handling routines) of those hot key functions. Every hot key funtionality has a different hot key ID in IMM and each ID has its own functionality according to the specific requirements of each country. Note that an application cannot add another predefined hot key ID into the system.

Following are the predefined hot key identifiers.

Hot Key ID	Description
IME_CHOTKEY_IME_NONIME_TOGGLE	Hot key for Simplified Chinese Edition. This hot key toggles between the IME and non-IME.
IME_CHOTKEY_SHAPE_TOGGLE	Hot key for Simplified Chinese Edition. This hot key toggles the shape conversion mode of the IME.
IME_CHOTKEY_SYMBOL_TOGGLE	Hot key for Simplified Chinese Edition. This hot key toggles the symbol conversion mode of the IME. The symbol mode indicates that the user can input Chinese punctuation and symbols (full shape characters) by mapping it to the punctuation and symbol keystrokes of the keyboard.
IME_JHOTKEY_CLOSE_OPEN	Hot key for Japanese Edition. This hot key toggles between closed and opened.
IME_THOTKEY_IME_NONIME_TOGGLE	Hot key for (Traditional) Chinese Edition. This hot key toggles between the IME and non-IME.
IME_THOTKEY_SHAPE_TOGGLE	Hot key for (Traditional) Chinese Edition. This hot key toggles the shape conversion mode of the IME.
IME_THOTKEY_SYMBOL_TOGGLE	Hot key for (Traditional) Chinese Edition. This hot key toggles the symbol conversion mode of the IME.

The other kind of hot key is the IME *private hot key,* but there is no functionality for this kind of hot key. It is just a placeholder for a hot key value. An IME can get this value by calling **ImmGetHotKey**. If an IME supports this functionality for one hot key ID, it will perform the functionality every time it finds this key input.

Following are the currently defined private IME hot key IDs.

Hot Key ID	Description

IME_ITHOTKEY_RESEND_RESULSTR	Hot key for (Traditional) Chinese Edition. This hot key should trigger the IME to resend the previous result string to the application. If the IME detects that this hot key is pressed, it needs to resend the previous result string to this application.
IME_ITHOTKEY_PREVIOUS_COMPOSITION	Hot key for (Traditional) Chinese Edition. This hot key should trigger the IME to bring up the previous composition string to the application.
IME_ITHOTKEY_UISTYLE_TOGGLE	Hot key for (Traditional) Chinese Edition. This hot key should trigger the IME UI to toggle the UI style between caret-related UI and the caret-unrelated UI.
IME_ITHOTKEY_RECONVERTSTRING	Hot key for (Traditional) Chinese Edition. This hot key should trigger the IME to make a reconversion. This is a new ID for Windows 98 and Windows 2000.

ImmGetHotKey

The **ImmGetHotKey** function gets the value of the IME hot key.

BOOL WINAPI
ImmGetHotKey(
DWORD *dwHotKeyID*,
LPUINT *lpuModifiers*,
LPUINT *lpuVKey*,
LPHKL *lphKL*
)

Parameters

dwHotKeyID
Hot key identifier.

lpuModifiers
Combination keys with the hot key. It includes ALT (MOD_ALT), CTRL (MOD_CONTROL), SHIFT (MOD_SHIFT), left-hand side (MOD_LEFT), and right-hand side (MOD_RIGHT).

The key up flag (MOD_ON_KEYUP) indicates that the hot key is effective when the key is up. The modifier ignore flag (MOD_IGNORE_ALL_MODIFIER) indicates that the combination of modifiers is ignored in hot key matching.

lpuVKey
Virtual key code of this hot key.

lphKL
HKL of the IME. If the return value of this parameter is not NULL, this hot key can switch to the IME with this HKL.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

Comments

This function is called by the Control Panel.

ImmSetHotKey

The **ImmSetHotKey** function sets the value of the IME hot key.

```
OOL WINAPI
    ImmSetHotKey(
    DWORD dwHotKeyID,
    UINT uModifiers,
    UINT uVKey,
    hKL hKL
   )
```

Parameters

dwHotKeyID
Hot key identifier.

uModifiers
Combination keys with the hot key. It includes ALT (MOD_ALT), CTRL (MOD_CONTROL), SHIFT (MOD_SHIFT), left-hand side (MOD_LEFT), and right-hand side (MOD_RIGHT).

The key up flag (MOD_ON_KEYUP) indicates that the hot key is effective when the key is up. The modifier ignore flag (MOD_IGNORE_ALL_MODIFIER) indicates that the combination of modifiers is ignored in hot key matching.

uVKey
Virtual key code of this hot key.

hKL
HKL of the IME. If this parameter is specified, this hot key can switch to the IME with this HKL.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

Comments

This function is called by the Control Panel. For a key that does not indicate a specific keyboard hand side, the *uModifiers* should specify both sides (MOD_LEFT|MODE_RIGHT).

IMM Soft Keyboard Functions

The following topics contain the IMM functions that are used by the IME to manipulate the soft keyboard.

ImmCreateSoftKeyboard

The **ImmCreateSoftKeyboard** function creates one type of soft keyboard window.

```
HWND WINAPI
    ImmCreateSoftKeyboard(
    UINT uType,
    UINT hOwner,
    int x,
    int y
   )
```

Parameters

uType
Specifies the type of the soft keyboard.

Utype	Description
SOFTKEYBOARD_TYPE_T1	Type T1 soft keyboard. This kind of soft keyboard should be updated by IMC_SETSOFTKBDDATA.
SOFTKEYBOARD_TYPE_C1	Type C1 soft keyboard. This kind of soft keyboard should be updated by IMC_SETSOFTKBDDATA with two sets of 256-word array data. The first set is for nonshift state, and the second is for shift state.

hOwner
Specifies the owner of the soft keyboard. It must be the UI window.

x
Specifies the initial horizontal position of the soft keyboard.

y
Specifies the initial vertical position of the soft keyboard.

Return Values

This function returns the window handle of the soft keyboard.

ImmDestroySoftKeyboard

The **ImmDestroySoftKeyboard** function destroys the soft keyboard window.

BOOL WINAPI
ImmDestroySoftKeyboard(
HWND *hSoftKbdWnd*
)

Parameters

hSoftKbdWnd
Window handle of the soft keyboard to destroy.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

ImmShowSoftKeyboard

The **ImmShowSoftKeyboard** function shows or hides the given soft keyboard.

BOOL WINAPI
ImmShowSoftKeyboard(
HWND *hSoftKbdWnd*,
int *nCmdShow*
)

Parameters

hSoftKbdWnd
Window handle of the soft keyboard.

nCmdShow
Shows the state of the window. The following values are provided.

NcmdShow	Meaning
SW_HIDE	Hides the soft keyboard.
SW_SHOWNOACTIVATE	Displays the soft keyboard

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

Messages

The following topics contain the messages that the UI window receives.

WM_IME_SETCONTEXT

The **WM_IME_SETCONTEXT** message is sent to an application when a window of the application is being activated. If the application does not have an application IME window, the application has to pass this message to the **DefWindowProc** and should return the return value of the **DefWindowProc**. If the application has an application IME window, the application should call **ImmIsUIMessage**.**WM_IME_SETCONTEXT**

fSet= (BOOL) wParam;

lISCBits = lParam;

Parameters

fSet
fSet is TRUE when the Input Context becomes active for the application. When it is FALSE, the Input Context becomes inactive for the application.

lISCBits
lISCBits consists of the following bit combinations.

Value	Description
ISC_SHOWUICOMPOSITIONWINDOW	Shows the composition window.
ISC_SHOWUIGUIDWINDOW	Shows the guide window.
ISC_SHOWUICANDIDATEWINDOW	Shows the candidate window of Index 0.
(ISC_SHOWUICANDIDATEWINDOW << 1)	Shows the candidate window of Index 1.
(ISC_SHOWUICANDIDATEWINDOW << 2)	Shows the candidate window of Index 2.
(ISC_SHOWUICANDIDATEWINDOW << 3)	Shows the candidate window of Index 3.

Return Values

The return value is the return value of **DefWindowProc** or **ImmIsUIMessage**.

Comments

After an application calls **DefWindowProc**(or **ImmIsUIMessage** with WM_IME_SETCONTEXT, the UI window receives WM_IME_SETCONTEXT. If the bit is on, the UI window shows the composition, guide, or candidate window as the bit status of *lParam*.If an application draws the composition window by itself, the UI window does not need to show its composition window. The application then has to clear the ISC_SHOWUICOMPOSITIONWINDOW bit of *lParam* and call **DefWindowProc** or **ImmIsUIMessage** with it.

WM_IME_CONTROL

The **WM_IME_CONTROL** message is a group of sub messages used to control the IME User Interface. An application uses this message to interact with the IME window created by the application.

WM_IME_CONTROL

wSubMessage= wParam;

lpData = (LPVOID) lParam;

Parameters

wSubMessage
Submessage value.

LpData
Dependent on each *wSubMessage.*

The following topics contain the submessages classified by the *wSubMessage* value.

Except for IMC_GETSOFTKBDSUBTYPE, IMC_SETSOFTKBDSUBTYPE, IMC_SETSOFTKBDDATA, IMC_GETSOFTKBDFONT, IMC_SETSOFTKBDFONT, IMC_GETSOFTKBDPOS and IMC_SETSOFTKBDPOS, it is recommended that applications use IMM APIs instead of the IMC messages to communicate with the IME window.

IMC_GETCANDIDATEPOS

The **IMC_GETCANDIDATEPOS** message is sent by an application to the IME window to get the position of the candidate window. The IME can adjust the position of a candidate window in respect to the screen boundary. In addition, an application can obtain the real position of a candidate window to determine whether to move it to another position.

WM_IME_CONTROL

wSubMessage= IMC_GETCANDIDATEPOS;

lpCANDIDATENFORM = (LPCANDIDATEFORM) lParam;

Parameters

lpCANDIDATENFORM
Buffer to retrieve the position of the candidate window.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

In return, the IME will fill the **CANDIDATEFORM** pointed to by *lpCANDIDATENFORM* with the client coordinates of the application's focus window. The UI window receives this message. An application should specify *lpCANIDATEFORM*->dwIndex to 0 ~ 3 to obtain a different candidate window position. (For example, index 0 is a top-level candidate window.)

IMC_GETCOMPOSITONFONT

The **IMC_GETCOMPOSITONFONT** message is sent by an application to the IME window to obtain the font to use in displaying intermediate characters in the composition window.

WM_IME_CONTROL

wSubMessage= IMC_GETCOMPOSITIONFONT;

lpLogFont= (LPLOGFONT) lParam;

Parameters

lpLogFont
Buffer to retrieve the LOGFONT.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

The UI window does not receive this message.

IMC_GETCOMPOSITONWINDOW

The **IMC_GETCOMPOSITONWINDOW** message is sent by an application to the IME window to get the position of the composition window. An IME can adjust the position of a composition window, and an application can obtain the real position of composition window to determine whether to move it to another position.

WM_IME_CONTROL

wSubMessage= IMC_GETCOMPOSITIONWINDOW;

lpCOMPOSITIONFORM = (LPCOMPOSITIONFORM) lParam;

Parameters

lpCOMPOSITIONFORM
Buffer to retrieve the position of the composition window.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

In return, the IME will fill the **CANDIDATEFORM** pointed to by *lpCANDIDATENFORM* with the client coordinates of the application's focus window. The UI window receives this message.

IMC_GETSOFTKBDFONT

The **IMC_GETSOFTKBDFONT** message is sent by the IME to the soft keyboard window to obtain the font to use for character display in the soft keyboard window.

WM_IME_CONTROL

wSubMessage= IMC_GETSOFTKBDFONT;

lpLogFont= (LPLOGFONT) lParam;

Parameters

lpLogFont
Buffer to retrieve the LOGFONT.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

IMC_GETSOFTKBDPOS

The **IMC_GETSOFTKBDPOS** message is sent by an IME to the soft keyboard window to obtain the position of the soft keyboard window.

WM_IME_CONTROL

wSubMessage= IMC_GETSOFTKBDPOS;

lParam = 0;

Parameters

lParam
Not used.

Return Values

The return value specifies a **POINTS** structure that contains the *x* and *y* coordinates of the position of the soft keyboard window, in screen coordinates.

Comments

The **POINTS** structure has the following form:

```
typedef struct tagPOINTS { /* pts */
SHORT x;
SHORT y;
} POINTS;
```

IMC_GETSOFTKBDSUBTYPE

The **IMC_GETSOFTKBDSUBTYPE** message is sent by an IME to the soft keyboard window to obtain the subtype of the soft keyboard window set by **IMC_SETSOFTKBDSUBTYPE**.

WM_IME_CONTROL

wSubMessage= IMC_GETSOFTKBDSUBTYPE;

lParam = 0;

Parameters

lParam
Not used.

Return Values

The return value is the subtype of the soft keyboard set by **IMC_SETSOFTKBDSUBTYPE**. A return value of -1 indicates failure.

IMC_GETSTATUSWINDOWPOS

The **IMC_GETSTATUSWINDOWPOS** message is sent by an application to the IME window to get the position of the status window.

WM_IME_CONTROL

wSubMessage= IMC_GETSTATUSWINDOWPOS;

lParam = 0;

Parameters

lParam
Not used.

Return Values

The return value specifies a **POINTS** structure that contains the *x* and *y* coordinates of the position of the status window, in screen coordinates.

Comments

The **POINTS** structure has the following form:

```
typedef struct tagPOINTS { /* pts */
SHORT x;
SHORT y;
} POINTS;
```

Comments

The UI window receives the message.

IMC_SETCANDIDATEPOS

The **IMC_SETCANDIDATEPOS** message is sent by an application to the IME window to specify the display position of a candidate window. In particular, this applies to an application that displays composition characters by itself, but uses the IME UI to display candidates.

WM_IME_CONTROL

wSubMessage= IMC_SETCANDIDATEPOS;

lpCANDIDATEFORM= (LPCANDIDATEFORM) lParam;

Parameters

lpCANDIDATEFORM
Buffer includes the candidate window position information.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

The UI window does not receive this message.

IMC_SETCOMPOSITONFONT

The **IMC_SETCOMPOSITONFONT** message is sent by an application to the IME window to specify the font to use in displaying intermediate characters in the composition window.

WM_IME_CONTROL

wSubMessage= IMC_SETCOMPOSITIONFONT;

lpLogFont= (LPLOGFONT) lParam;

Parameters

lpLogFont
Buffer includes the LOGFONT data to set.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

The UI window does not receive this message.

IMC_SETCOMPOSITONWINDOW

The **IMC_SETCOMPOSITONWINDOW** message is sent by an application to the IME window to set the style of the composition window in the current active Input Context. Once an application sets the style, the IME user interface then follows the style specified in the Input Context.

WM_IME_CONTROL

wSubMessage= IMC_SETCOMPOSITIONWINDOW;

lpCOMPOSITIONFORM= (LPCOMPOSITIONFORM) lParam;

Parameters

lpCOMPOSITIONFORM
COMPOSITIONFORM structure includes the new styles for the composition window.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

The IME user interface uses a default style for the composition window that is equal to the CFS_POINT style. If an application has not specified a composition style in its Input Context, the IME user interface retrieves the current caret position and window client area when it opens the composition window (in client coordinates). The UI window does not receive this message.

IMC_SETSOFTKBDDATA

The **IMC_SETSOFTKBDDATA** message is sent by the IME to the soft keyboard window to specify the character code to use for displaying characters in the soft keyboard window.

WM_IME_CONTROL

wSubMessage= IMC_SETSOFTKBDDATA;

lpSoftKbdData= (LPSOFTKBDDATA) lParam;

Parameters

lpSoftKbdData
Points to the buffer to specify the character code to use for displaying characters.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

The UI window does not receive this message.

IMC_SETSOFTKBDSUBTYPE

The **IMC_SETSOFTKBDSUBTYPE** message is sent by the IME to the soft keyboard window to specify the subtype to use for displaying characters in the soft keyboard window. It also can be used for IME-specific purposes.

WM_IME_CONTROL

wSubMessage= IMC_SETSOFTKBDSUBTYPE;

lSubType= lParam;

Parameters

lSubType
Specifies the subtype to set.

Return Values

The return value is the subtype. A return value of -1 indicates failure.

Comments

The UI window does not receive this message, and the SOFTKEYBOARD_TYPE_T1 does not use this information. The IME sends this message so the soft keyboard will not change the displayed reading characters. The IME can use the SOFTKEYBOARD_TYPE_T1 soft keyboard to define the meaning of this message and can obtain this data by using IMC_GETSOFTKBDSUBTYPE.

IMC_SETSOFTKBDFONT

The **IMC_SETSOFTKBDFONT** message is sent by the IME to the soft keyboard window to specify the font to use in displaying characters in the soft keyboard window.

WM_IME_CONTROL

wSubMessage= IMC_SETSOFTKBDFONT;

lpLogFont= (LPLOGFONT)lParam;

Parameters

lpLogFont
Points to the LOGFONT to be set.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

The UI window does not receive this message.

IMC_SETSOFTKBDPOS

The **IMC_SETSOFTKBDPOS** message is sent by the UI window to soft keyboard window to set the position of the soft keyboard window.

WM_IME_CONTROL

wSubMessage= IMC_SETSOFTKBDPOS;

ptsPt= (POINTS)lParam;

Parameters

ptsPt
Specifies a **POINTS** structure that contains the *x* and *y* coordinates of the position of the soft keyboard window, in screen coordinates.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

The **POINTS** structure has the following form:

```
typedef struct tagPOINTS { /* pts */
SHORT x;
SHORT y;
} POINTS;
```

IMC_SETSTATUSWINDOWPOS

The **IMC_SETSTATUSWINDOWPOS** message is sent by an application to the IME window to set the position of the status window.

WM_IME_CONTROL

wSubMessage= IMC_SETSTATUSWINDOWPOS;

ptsPt= (POINTS)lParam;

Parameters

ptsPt
Specifies a **POINTS** structure that contains the *x* and *y* coordinates of the position of the status window, in screen coordinates.

Return Values

If the message is successful, the return value is zero. Otherwise, the return value is nonzero.

Comments

The **POINTS** structure has the following form:

```
typedef struct tagPOINTS { /* pts */
SHORT x;
SHORT y;
} POINTS;
```

WM_IME_COMPOSITION

The **WM_IME_COMPOSITION** message is sent to an application when the IME composition status is changed (by the user). The message consists of two bytes of composition character. The IME user interface window changes its appearance when it processes this message. An application can call **ImmGetCompositionString** to obtain the new composition status.

WM_IME_COMPOSITION

wChar= wParam;

lAttribute= lParam;

Parameters

wChar

Consists of two bytes of DBCS character that is the latest change of composition character.

lAttribute

Contains the following flag combinations. Basically, the flag indicates how the composition string or character has changed. An application checks this to retrieve necessary information.

Value	Description
GCR_ERRORSTR	Updates the error string.
GCR_INFORMATIONSTR	Updates the information string.
GCS_COMPATTR	Updates the attribute of the composition string.
GCS_COMPCLAUSE	Updates clause information of the composition string.
GCS_COMPREADATTR	Updates the attributes of the reading string of the current composition.
GCS_COMPREADCLAUSE	Updates the clause information of the reading string of the composition string.
GCS_COMPREADSTR	Updates the reading string of the current composition.
GCS_COMPSTR	Updates the current composition string.
GCS_CURSORPOS	Updates the cursor position in composition string.
GCS_DELTASTART	Updates the starting position of any changes in composition string.
GCS_RESULTCLAUSE	Updates clause information of the result string.
GCS_RESULTREADCLAUSE	Updates clause information of the reading string.
GCS_RESULTREADSTR	Updates the reading string.
GCS_RESULTSTR	Updates the string of the composition result.

The following style bit values are provided for WM_IME_COMPOSITION.

Value	Description

CS_INSERTCHAR	An IME specifies this value when *wParam* shows a composition character that should be inserted into the current insertion point. An application should display a composition character if it processes this bit flag.
CS_NOMOVECARET	An IME specifies this value when it does not want an application to move the caret position as a result of processing WM_IME_COMPOSITION. For example, if an IME specifies a combination of CS_INSERTCHAR and CS_NOMOVECARET, it means that an application should insert a character given by *wParam* to the current caret position, but should not move the caret. Subsequent WM_IME_COMPOSITION messages containing the GCS_RESULTSTR flag will replace this character.

Return Values

None.

Comments

When an application wants to display composition characters by themselves, it should not pass this message to the application IME user interface window or to **DefWindowProc**. The **DefWindowProc** function processes this message to pass to the Default IME window. An IME should send this message to an application even when the IME only cancels the current composition. This message should also be used to notify an application or IME UI to erase the current composition string.

See Also

ImmGetCompositionString

WM_IME_COMPOSITIONFULL

The **WM_IME_COMPOSITIONFULL** message is sent to an application when the IME user interface window cannot increase the size of the composition window. An application should specify how to display the IME UI window when it receives this message.

WM_IME_COMPOSITIONFULL

wParam = 0

lParam= 0

Parameters

wParam
Not used.

lParam
Not used.

Return Values

None.

Comments

This message is a notification, which is sent to an application by the IME user interface window and not by the IME itself. The IME uses **SendMEssage** to send this notification.

See Also

IMC_SETCOMPOSITONWINDOW

WM_IME_ENDCOMPOSITION

The **WM_IME_ENDCOMPOSITION** message is sent to an application when the IME ends composition.

WM_IME_ENDCOMPOSITION

wParam = 0

lParam= 0

Parameters

wParam
Not used.

lParam
Not used.

Return Values

None.

Comments

When an application wants to display composition characters by themselves, it should not pass this message to the application IME UI window or to **DefWindowProc**. **DefWindowProc** processes this message to pass it to the default IME window.

WM_IME_SELECT

The **WM_IME_SELECT** message is sent to the UI window when the system is about to change the current IME.

WM_IME_SELECT

fSelect= (BOOL)wParam;

hKL= lParam;

Parameters

fSelect
TRUE if the IME is newly selected. Otherwise, it is FALSE if the IME is unselected.

hKL
Input language handle of the IME.

Return Values

None.

Comments

The system IME class uses this message to create a new UI window and destroy an old UI window for an application or system. **DefWindowProc** processes this message to pass the information to the default IME window. The default IME window then sends this message to its UI window.

WM_IME_STARTCOMPOSITION

The **WM_IME_STARTCOMPOSITION** message is sent immediately before an IME generates a composition string as a result of a user's keystroke. The UI window opens its composition window when it receives this message.

WM_IME_STARTCOMPOSITION

wParam = 0

lParam= 0

Parameters

wParam
Not used.

lParam
Not used.

Return Values

None.

Comments

When an application wants to display composition characters by themselves, it should not pass this message to the application IME window or to **DefWindowProc**. The **DefWindowProc** function processes this message to pass it to the default IME window.

WM_IME_NOTIFY

The **WM_IME_NOTIFY** message is a group of submessages that notifies an application or UI window of the IME status.

WM_IME_NOTIFY

wSubMessage= wParam; //submessage ID

lParam= lParam; // depends on the submessage

The following topics contain the submessages classified by the value of *wSubMessage*.

IMN_CLOSESTATUSWINDOW

The **IMN_CLOSESTATUSWINDOW** message is sent when an IME is about to close a status window.

WM_IME_NOTIFY

wSubMessage = IMN_CLOSESTATUSWINDOW;

lParam= 0;

Parameters

lParam
Not used.

Return Values

None.

Comments

The UI window closes the status window when it receives this message.

IMN_OPENSTATUSWINDOW

The **IMN_OPENSTATUSWINDOW** message is sent when an IME is about to create a status window. An application then processes this message and displays a system window for the IME itself.

An application can obtain information about the system window by calling the **ImmGetConversionStatus** function.

WM_IME_NOTIFY

wSubMessage = IMN_OPENSTATUSWINDOW;

lParam= 0;

Parameters

lParam
Not used.

Return Values

None.

Comments

The UI window creates a status window when it receives this message.

See Also

ImmGetConversionStatus

IMN_OPENCANDIDATE

The **IMN_OPENCANDIDATE** message is sent when an IME is about to open a candidate window. An application then processes this message and calls **ImmGetCandidateCount** and **ImmGetCandidateList** to display the candidate window itself.

WM_IME_NOTIFY

wSubMessage = IMN_OPENCANDIDATE;

lCandidateList= lParam;

Parameters

lCandidateList
Shows which candidate list should be updated. For example, if bit 0 is 1, the first candidate list should be updated. If bit 31 is 1, the 32nd candidate list should be updated.

Return Values

None.

Comments

The UI window creates a candidate window when it receives this message.

See Also

ImmGetCandidateListCount, ImmGetCandidateList, WM_IME_CHANGECANDIDATE

IMN_CHANGECANDIDATE

The **IMN_CHANGECANDIDATE** message is sent when an IME is about to change the content of a candidate window. An application then processes this message to display the candidate window itself.

WM_IME_NOTIFY

wSubMessage = IMN_CHANGECANDIDATE;

lCandidateList= lParam;

Parameters

lCandidateList
Shows which candidate list should be updated. For example, if bit 0 is 1, the first candidate list should be updated. If bit 31 is 1, the 32nd candidate list should be updated.

Return Values

None.

Comments

The UI window redraws a candidate window when it receives this message.

See Also

ImmGetCandidateCount, **ImmGetCandidateList**

IMN_CLOSECANDIDATE

The **IMN_CLOSECANDIDATE** message is sent when an IME is about to close a candidate window. An application processes this message to obtain information about the end of candidate processing.

WM_IME_NOTIFY

wSubMessage = IMN_CLOSECANDIDATE;

lCandidateList= lParam;

Parameters

lCandidateList
Shows which candidate list should be closed. For example, if bit 0 is 1, the first candidate list should be updated. If bit 31 is 1, the 32nd candidate list should be updated.

Return Values

None.

Comments

The UI window destroys a candidate window when it receives this message.

IMN_SETCONVERSIONMODE

The **IMN_SETCONVERSIONMODE** message is sent when the conversion mode of the Input Context is updated. When the application or UI window receives this message, either one can call **ImmGetConversionStatus** to obtain information about the status window.

WM_IME_NOTIFY

wSubMessage = IMN_SETCONVERSIONMODE;

lParam= 0;

Parameters

lParam
Not used.

Return Values

None.

Comments

The UI window redraws the status window if the status window shows the conversion mode.

IMN_SETSENTENCEMODE

The **IMN_SETSENTENCEMODE** message is sent when the sentence mode of the Input Context is updated. When the application or UI window receives this message, either one can call **ImmGetConversionStatus** to obtain information about the status window.

WM_IME_NOTIFY

wSubMessage = IMN_SETSENTENCEMODE;

lParam= 0;

Parameters

lParam
Not used.

Return Values

None.

Comments

The UI window redraws the status window if the status window shows the sentence mode.

IMN_SETOPENSTATUS

The **IMN_SETOPENSTATUS** message is sent when the open status of the Input Context is updated. When the application or UI window receives this message, either one can call **ImmGetOpenStatus** to obtain information.

WM_IME_NOTIFY

wSubMessage = IMN_SETOPENSTATUS;

lParam= 0;

Parameters

lParam
Not used.

Return Values

None.

Comments

The UI window redraws the status window if the status window shows the open/close status.

IMN_SETCANDIDATEPOS

The **IMN_SETCANDIDATEPOS** message is sent when an IME is about to move the candidate window. An application processes this message to obtain information about the end of candidate processing.

WM_IME_NOTIFY

wSubMessage = IMN_SETCANDIDATEPOS;

lCandidateList= lParam;

Parameters

lCandidateList
Shows which candidate list should be moved. For example, if bit 0 is 1, the first candidate list should be updated. If bit 31 is 1, the 32nd candidate list should be updated.

Return Values

None.

Comments

The UI window moves a candidate window when it receives this message.

IMN_SETCOMPOSITIONFONT

The **IMN_SETCOMPOSITIONFONT** message is sent when the font of the Input Context is updated. When the application or UI window receives this message, either one can call **ImmGetCompositionFont** to obtain information about the composition font.

WM_IME_NOTIFY

wSubMessage = IMN_SETCOMPOSITIONFONT;

lParam= 0;

Parameters

lParam
Not used.

Return Values

None.

Comments

The composition component of the UI window uses the font information by calling **ImmGetCompositionFont** to draw the text of the composition string.

IMN_SETCOMPOSITIONWINDOW

The **IMN_SETCOMPOSITIONWINDOW** message is sent when the composition form of the Input Context is updated. When the UI window receives this message, the *cfCompForm* of the Input Context can be referenced to obtain the new conversion mode.

WM_IME_NOTIFY

wSubMessage = IMN_SETCOMPOSITIONWINDOW;

lParam= 0;

Parameters

lParam
Not used.

Return Values

None.

Comments

The composition component of the UI window uses *cfCompForm* to show the composition window.

IMN_GUIDELINE

The **IMN_GUIDELINE** message is sent when an IME is about to show an error or information. When the application or UI window receives this message, either one can call **ImmGetGuideLine** to obtain information about the guideline.

WM_IME_NOTIFY

wSubMessage = IMN_GUIDELINE;

lParam= 0;

Parameters

lParam
Not used. Has to be zero.

Return Values

None.

Comments

The UI window can create an information window when it receives this message and show the information string.

See Also

ImmGetGuideLine, **GUIDELINE** structure

IMN_SOFTKBDDESTROYED

The **IMN_SOFTKBDDESTROYED** message is sent to the UI window when the soft keyboard is destroyed.

WM_IME_NOTIFY

wSubMessage = IMN_SOFTKBDDESTROYED;

lParam= 0;

Parameters

lParam
Not used. Has to be zero.

Return Values

None.

WM_IME_KEYDOWN and WM_IME_KEYUP

The **WM_IME_KEYDOWN** and **WM_IME_KEYUP** messages are sent to an application when an IME needs to generate a **WM_KEYDOWN** or **WM_KEYUP** message. The value sent is the same as the original Windows **WM_KEYDOWN** and **WM_KEYUP** value (English version).

WM_IME_KEYDOWN / WM_IME_KEYUP

nVirtKey = (int) wParam; // virtual-key code

lKeyData = lParam; // key data

Parameters

nVirtKey
Value of *wParam*. Specifies the virtual key code of the nonsystem key.

lKeyData
Value of *lParam*. Specifies the repeat count, scan code, extended key flag, context code, previous key state flag, and transition state flag. It is the same as for the original Windows **WM_KEYDOWN** and **WM_KEYUP** messages

Return Values

None.

Comments

An application can handle this message the same way as the **WM_KEYDOWN** and **WM_KEYUP** message. Otherwise, **DefWindowProc** processes this message to generate a **WM_KEYDOWN** or **WM_KEYUP** message with the same *wParam* and *lParam* parameters. This message is usually generated by the IME to maintain message order.

WM_IME_CHAR

The **WM_IME_CHAR** message is sent to an application when the IME gets a character of the conversion result. The value that is sent is similar to the original Windows **WM_CHAR** (English version). The difference is that *wParam* can include two bytes of character.

WM_IME_CHAR

wCharCode = wParam;

lKeyData = lParam;

Parameters

wCharCode
Includes two bytes for an FE character. For NT Unicode application, it includes one Unicode character.

lKeyData
Same as the original Windows **WM_CHAR** (English Version). Following are the available bits and their description.

Value	Description
0 – 15	Repeat count. Since the first byte and second byte are continuous, this is always 1.
16 – 23	Scan Code. Scan code for a complete FE character.
24 – 28	Not used.
29	Context code.
31	Conversion state.

Return Values

None.

Comments

If the application does not handle this message, the **DefWindowProc** function processes this message to generate **WM_CHAR** messages. If the application is not Unicode based and *wCharCode* includes 2 bytes of DBCS character, the **DefWindowProc** function will generate two WM_CHAR messages, each message containing 1 byte of the DBCS character. If the message just includes an SBCS character, **DefWindowProc** generates only one **WM_CHAR** message.

VK_PROCESSKEY

The **VK_PROCESSKEY** message is sent to an application as a *wParam* of **WM_KEYDOWN** or **WM_KEYUP**. When this virtual key is generated, either the real virtual key is saved in the Input Context or the messages that were generated by IME are stored in the Input Context. The system either restores the real virtual key or posts the messages that are stored in the message buffer of the Input Context.

WM_KEYDOWN /WM_KEYUP

wParam = VK_PROCESSKEY;

lParam= 1;

Parameters

lParam
Must be 1.

INDICM_SETIMEICON

This message is sent to the Indicator window when the IME wants to change the icon for System Pen icon. This message can be accepted when the selected *hKL* of the focused window is the same as the sender IME.

INDICM_SETIMEICON

nIconIndex = wParam;

hKL = lParam;

Parameters

nIconIdex
Index for the icon resource of the IME file. If this value is (-1), the Indicator restores the original icon provided by the system.

lKeyhKL that is the sender IME.

Return Values

A nonzero value indicates failure. Otherwise, zero is returned.

Comments

Due to the internal design requirement in the task bar manager, the IME must use **PostMessage** for INDICM_xxx messages.

INDICM_SETIMETOOLTIPS

This message is sent to the Indicator window when the IME wants to change the Tooltip string for the System Pen icon. This message can be accepted when the selected *hKL* of the focused window is the same as the sender IME.**INDICM_SETIMETOOLTIPS**

hAtom = wParam;

hKL = lParam;

Parameters

hAtom
Global ATOM value for the Tooltip string. If this value is (-1), the Indicator restores the original tips provided by the system.

lKeyhKL that is the sender IME.

Return Values

A nonzero indicates failure. Otherwise, zero is returned.

Comments

Due to the internal design requirement in the task bar manager, the IME must use **PostMessage** for INDICM_xxx messages. The global ATOM must be retrieved by **GlobalAddAtom** or **GlobalFindAtom**.

INDICM_REMOVEDEFAULTMENUITEMS

This message is sent to the Indicator window when the IME wants to remove the default menu items of the System Pen icon.**INDICM_REMOVEDEFAULTMENUITEMS**

wValue = wParam;

hKL = lParam;

Parameters

wValue
wValue is a combination of the following bits.

Value	Description
RDMI_LEFT	Removes the menu items of the left click menu.
RDMI_RIGHT	Removes the menu items of the right click menu.

If *wValue* is zero, all default menu items are restored.

lKeyhKL that is the sender IME.

Return Values

A nonzero indicates failure. Otherwise, zero is returned.

Comments

Due to the internal design requirement in the task bar manager, the IME must use **PostMessage** for INDICM_xxx messages.

IME Interface Functions

IMEs are provided as dynamic-link libraries (DLLs). The Input Method Manager (IMM) should handle all installed IMEs. Because IMEs are changeable at run time without rebooting, the IMM will have a structure to maintain all the entry points of each IME.

The following topics contain all the common IME functions. These functions should not be called by an application directly.

ImeInquire

For Windows 95, Windows 98, and Windows NT 3.51

The **ImeInquire** function handles initialization of the IME. It also returns an **IMEINFO** structure and the UI class name of the IME.

BOOL
ImeInquire(
LPIMEINFO *lpIMEInfo***,**
LPTSTR *lpszWndClass***,**
LPCTSTR *lpszData*
)

Parameters

lpIMEInfo
Pointer to the IME info structure.

lpszWndClass
Window class name that should be filled by the IME. This name is the IME's UI class.

lpszData
IME option block. NULL for this version.

For Windows NT 4.0 and Windows 2000

BOOL
ImeInquire(
LPIMEINFO *lpIMEInfo***,**
LPTSTR*lpszWndClass***,**
DWORD *dwSystemInfoFlags*
)

Parameters

lpIMEInfo
Pointer to the IME info structure.

lpszWndClass
Window class name that should be filled by the IME. This name is the IME's UI class.

dwSystemInfoFlags
Varying system information provided by the system. The following flags are provided.

Flag	Description
IME_SYSINFO_WINLOGON	Tells the IME that the client process is the Winlogon process. The IME should not allow users to configure the IME when this flag is specified.
IME_SYSINFO_WOW16	Tells the IME that the client process is a 16-bit application.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

ImeConversionList

The **ImeConversionList** function obtains a converted result list from another character or string.

DWORD
IMEConversionList(
HIMC *hIMC*,
LPCTSTR*lpSrc*,
LPCANDIDATELIST *lpDst*,
DWORD *dwBufLen*,
UINT *uFlag*
)

Parameters

hIMC
Input context handle.

lpSrc
Character string to be converted.

lpDst
Pointer to the destination buffer.

dwBufLen
Length of the destination buffer.

uFlag
Currently can be one of the following three flags.

Flag	Description
GCL_CONVERSION	Specifies the reading string to the *lpSrc* parameter. The IME returns the result string in the *lpDst* parameter.
GCL_REVERSECONVERSION	Specifies the result string in the *lpSrc* parameter. The IME returns the reading string in the *lpDst* parameter.
GCL_REVERSE_LENGTH	Specifies the result string in the *lpSrc* parameter. The IME returns the length that it can handle in GCL_REVERSECONVERSION. For example, an IME cannot convert a result string with a sentence period to a reading string. As a result, it returns the string length in bytes without the sentence period.

Return Values

The return value is the number of bytes of the result string list.

Comments

This function is intended to be called by an application or an IME without generating IME-related messages. Therefore, an IME should not generate any IME-related messages in this function.

ImeConfigure

The **ImeConfigure** function provides a dialog box to use to request optional information for an IME.

BOOL
ImeConfigure(
HKL *hKL*,
HWND *hWnd*,
DWORD *dwMode*,
LPVOID *lpData*
)

Parameters

hKL
Input language handle of this IME.

hWnd
Parent window handle.

dwMode
Mode of dialog. The following flags are provided.

Flag	Description
IME_CONFIG_GENERAL	Dialog for general purpose configuration.
IME_CONFIG_REGWORD	Dialog for register word.
IME_CONFIG_SELECTDICTIONARY	Dialog for selecting the IME dictionary.

lpData
Pointer to VOID, which will be a pointer to the **REGISTERWORD** structure only if dwMode==IME_CONFIG_REGISTERWORD. Otherwise, *lpData* should just be ignored.
This also can be NULL with the IME_CONFIG_REGISTER mode, if no initial string information is given.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

Comments

An IME checks *lpData* in the following way in the pseudo code.

```
if (dwmode != IME_CONFIG_REGISTERWORD)
    {
// Does original execution
    }
else if (IsBadReadPtr(lpdata, sizeof(REGISTERWORD))==FALSE)
    {

if (IsBadStringPtr(PREGISTERWORD(lpdata)->lpReading, (UINT)-1)==FALSE)
    {
// Set the reading string to word registering dialogbox
    }
if (IsBadStringPtr(PREGISTERWORD(lpdata)->lpWord, (UINT)-1)==FALSE)
    {
// Set the word string to word registering dialogbox
    }
    }
```

ImeDestroy

The **ImeDestroy** function terminates the IME itself.

BOOL
ImeDestroy(
UINT *uReserved*
)

Parameters

uReserved
Reserved. Currently, it should be zero. For this version, the IME should return FALSE if it is not zero.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

ImeEscape

The **ImeEscape** function allows an application to access capabilities of a particular IME not directly available though other IMM functions. This is necessary mainly for country-specific functions or private functions in the IME.

LRESULT
ImeEscape(
HIMC *hIMC*,
UINT *uEscape*,
LPVOID *lpData*
)

Parameters

hIMC
Input context handle

uEscape
Specifies the escape function to be performed.

lpData
Points to the data required for the specified escape.

The **ImeEscape** function supports the following escape functions.

uEscape	**Meaning**
IME_ESC_QUERY _SUPPORT	Checks for implementation. If this escape is not implemented, the return value is zero.
IME_ESC_RESERVED_FIRST	Escape that is between IME_ESC_RESERVED_FIRST and IME_ESC_RESERVED_LAST is reserved by the system.
IME_ESC_RESERVED_LAST	Escape that is between IME_ESC_RESERVED_FIRST and IME_ESC_RESERVED_LAST is reserved by the system.
IME_ESC_PRIVATE_FIRST	Escape that is between IME_ESC_PRIVATE_FIRST and IME_ESC_PRIVATE_LAST is reserved for the IME. The IME can freely use these escape functions for its own purposes.
IME_ESC_PRIVATE_LAST	Escape that is between IME_ESC_PRIVATE_FIRST and IME_ESC_PRIVATE_LAST is reserved for the IME. The IME can freely use these escape functions for its own purposes.

IME_ESC_SEQUENCE_TO_INTERNAL	Escape that is Chinese specific. An application that wants to run under all Far East platforms should not use this. It is for the Chinese EUDC editor. The *(LPWORD)*lpData* is the sequence code, and the return value is the character code for this sequence code. Typically, the Chinese IME will encode its reading character codes into sequence 1 to *n*.
IME_ESC_GET_EUDC_DICTIONARY	Escape that is Chinese specific. An application that wants to run under all Far East platforms should not use this. It is for the Chinese EUDC editor. On return from the function, the (LPTSTR)*lpData* is filled with the full path file name of the EUDC dictionary. The size of this buffer pointed by *lpData* should be greater or egual to MAX_PATH * sizeof(TCHAR). Note: Windows 95/98 and Windows NT 4.0 EUDC editor expect IMEs just use the buffer up to 80*sizeof(TCHAR).
IME_ESC_SET_EUDC_DICTIONARY	Sets the EUDC dictionary file. On input, the lpData parameter is the pointer to a null-terminated string specifying the full path. For use by the Chinese EUDC editor; do not use in other applications.
IME_ESC_MAX_KEY	Escape that is Chinese specific. An application that wants to run under all Far East platforms should not use this. It is for the Chinese EUDC editor. The return value is the maximum keystrokes for a EUDC character.
IME_ESC_IME_NAME	Escape that is Chinese specific. An application that wants to run under all Far East platforms should not use this. It is for the Chinese EUDC editor. On return from the function, the (LPTSTR) is the IME name to be displayed on the EUDC editor. The size of this buffer pointed to by *lpData* should be greater or equal to 16 * sizeof(TCHAR).
IME_ESC_SYNC_HOTKEY	Escape that is (Traditional) Chinese specific. An application that wants to run under all Far East platforms should not use this. It is for synchronizing between different IMEs. The input parameter *(LPDWORD)*lpData* is the IME private hot key ID. If this ID is zero, this IME should check every private hot key ID it concerns.

IME_ESC_HANJA_MODE	Escape that is Korean specific. An application that wants to run under all Far East platforms should not use this. It is for conversion from Hangeul to Hanja. The input parameter (LPSTR)*lpData* is filled with Hangeul characters that will be converted to Hanja and its null-terminated string. When an application wants to convert any Hangeul character to a Hanja character by using the same method as the Hanja conversion when the composition character is present, the application only needs to request this function. The IME will then set itself as the Hanja conversion mode.
IME_ESC_GETHELPFILENAME	Escape that is the name of the IME's help file. On return from the function, the (LPTSTR)*lpData* is the full path file name of the IME's help file. The path name should be less than MAX_PATH * sizeof(TCHAR). This is added to Windows 98 and Windows 2000. Note: Windows 98 expects the path length is less than 80 TCHARs.
IME_ESC_PRIVATE_HOTKEY	*lpdata* points to a DWORD that contains the hot key ID (in the range of IME_HOTKEY_PRIVATE_FIRST and IME_HOTKEY_PRIVATE_LAST). After the system receives the hot key request within this range, the IMM will dispatch it to the IME using the **ImeEscape** function. Note: Windows® 95 does not support this escape.

Return Values

If the function fails, the return value is zero. Otherwise, the return value depends on each escape function.

Comments

Parameter validation should be inside each escape function for robustness.

When *uEscape* is IME_ESC_QUERY _SUPPORT, *lpData* is the pointer to the variable that contains the IME escape value. Following is an example that can be used to determine if the current IME supports IME_ESC_GETHELPFILENAME.

DWORD dwEsc = IME_ESC_GETHELPFILENAME;

LRESULT lRet = ImmEscape(hKL, hIMC, IME_ESC_QUERYSUPPORT, (LPVOID)&dwEsc);

See Also

ImmEscape

ImeSetActiveContext

The **ImeSetActiveContext** function notifies the current IME active Input Context.

BOOL
ImeSetActiveContext(
HIMC *hIMC*,
BOOL *fFlag*
)

Parameters

hIMC
Input context handle.

fFlag
Two flags are provided. TRUE indicates activated and FALSE indicates deactivated.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

Comments

The IME is informed by this function about a newly selected Input Context. The IME can carry out initialization, but it is not required.

See Also

ImeSetActiveContext

ImeProcessKey

The **ImeProcessKey** function preprocesses all the keystrokes given through the IMM and returns TRUE if that key is necessary for the IME with a given Input Context.

BOOL
ImeProcessKey(
HIMC *hIMC*,
UINT *uVirKey*,
DWORD *lParam*,
CONST LPBYTE *lpbKeyState*
)

Parameters

hIMC
Input context handle

uVirKey
Virtual key to be processed.

lParam
lParam of key messages.

lpbKeyState
Points to a 256-byte array that contains the current keyboard state. The IME should not modify the content of the key state.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

Comments

The system decides whether the key is handled by IME or not by calling this function. If the function returns TRUE before the application gets the key message, the IME will handle the key. The system will then call the **ImeToAsciiEx** function. If this function returns FALSE, the system recognizes that the key will not be handled by the IME and the key message will be sent to the application.

For IMEs that support IME_PROP_ACCEPT_WIDE_VKEY on Windows 2000, ImeProcessKey will receive full 32 bit value for uVirKey, which is injected by using SendInput API through VK_PACKET. uVirKey will include 16-bit Unicode in hiword even the IME may be ANSI version.

For IMEs that do not support IME_PROP_ACCEPT_WIDE_VKEY, Unicode IME's ImeProcessKey will receive VK_PACKET with zero'ed hiword. Unicode IME still can return TRUE so ImeToAsciiEx will be called with the injected Unicode. ANSI IME's ImeProcessKey will not receive anything. The injected Unicode will

be discarded if the ANSI IME is open. If the ANSI IME is closed, the injected Unicode message will be posted to application's queue immediately.

NotifyIME

The **NotifyIME** function changes the status of the IME according to the given parameters.

BOOL
NotifyIME(
HIMC *hIMC*,
DWORD *dwAction*,
DWORD *dwIndex*,
DWORD *dwValue*
)

Parameters

hIMC
Input context handle.

dwAction
Following are the context items that an application can specify in the *dwAction* parameter.

Context Item		Description
NI_OPENCANDIDATE		Application has the IME open the candidate list. If the IME opens the candidate list, the IME sends a **WM_IME_NOTIFY** (subfunction is IMN_OPENCANDIDATE) message.
	dwIndex	Index of the candidate list to be opened.
	dwValue	Not used.
NI_CLOSECANDIDATE		Application has the IME close the candidate list. If the IME closes the candidate list, the IME sends a **WM_IME_NOTIFY** (subfunction is IMN_CLOSECANDIDATE) message.
	dwIndex	Index of the candidate list to be closed.
	dwValue	Not used.
NI_SELECTCANDIDATESTR		Application selects one of the candidates.
	dwIndex	Index of the candidate list to be selected.
	dwValue	Index of the candidate string in the selected candidate list.
NI_CHANGECANDIDATELIST		Application changes the currently selected candidate.
	dwIndex	Index of the candidate list to be selected.
	dwValue	Not used.
NI_SETCANDIDATE_PAGESTART		Application changes the page starting index of the candidate list.
	dwIndex	Index of the candidate list to be changed.
	dwValue	New page start index.
NI_SETCANDIDATE_PAGESIZE		Application changes the page size of the candidate list.
	dwIndex	Index of the candidate list to be changed.
	dwValue	New page size.
NI_CONTEXTUPDATED		Application or system updates the Input Context.

	dwIndex	When the value of *dwValue* is IMC_SETCONVERSIONMODE, *dwIndex* is the previous conversion mode. When the value of *dwValue* is IMC_SETSENTENCEMODE, *dwIndex* is the previous sentence mode. For any other *dwValue, dwIndex* is not used.
	dwValue	*O*ne of following values used by the **WM_IME_CONTROL** message: IMC_SETCANDIDATEPOS IMC_SETCOMPOSITIONFONT IMC_SETCOMPOSITIONWINDOW IMC_SETCONVERSIONMODE IMC_SETSENTENCEMODE IMC_SETOPENSTATUS
NI_COMPOSITIONSTR		Application changes the composition string. This action takes effect only when there is a composition string in the Input Context.
	dwIndex	The following values are provided for *dwIndex:* CPS_COMPLETE To determine the composition string as the result string. CPS_CONVERT To convert the composition string. CPS_REVERT To revert the composition string. The current composition string will be canceled and the unconverted string will be set as the composition string. CPS_CANCEL To clear the composition string and set the status as no composition string.
	dwValue	Not used.

dwIndex
Dependent on *uAction*.

dwValue
Dependent on *uAction*.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

See Also

ImmNotifyIME

ImeSelect

The **ImeSelect** function is used to initialize and uninitialize the IME private context.

BOOL
ImeSelect(
HIMC *hIMC***,**

BOOL *fSelect*
)

Parameters

hIMC
Input context handle

fSelect
Two flags are provided. TRUE indicates initialize and FALSE indicates uninitialize (free resource).

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

ImeSetCompositionString

The **ImeSetCompositionString** function is used by an application to set the IME composition string structure with the data contained in the *lpComp* or *lpRead* parameters. The IME then generates a **WM_IME_COMPOSITION** message.

BOOL WINAPI
ImeSetCompositionString(
HIMC *hIMC*,
DWORD *dwIndex*,
LPCVOID *lpComp*,
DWORD *dwCompLen*,
LPCVOID *lpRead*,
DWORD *dwReadLen*
);

Parameters

hIMC
Input context handle.

dwIndex
The following values are provided for *dwIndex.*

Value	Description
SCS_SETSTR	Application sets the composition string, the reading string, or both. At least one of the *lpComp* and *lpRead* parameters must point to a valid string. If either string is too long, the IME truncates it.
SCS_CHANGEATTR	Application sets attributes for the composition string, the reading string, or both. At least one of the *lpComp* and *lpRead* parameters must point to a valid attribute array.
SCS_CHANGECLAUSE	Application sets the clause information for the composition string, the reading string, or both. At least one of the *lpComp* and *lpRead* parameters must point to a valid clause information array.

SCS_QUERYRECONVERTSTRING	Application asks the IME to adjust its **RECONERTSTRINGSTRUCTRE**. If the application calls the **ImeSetCompositionString** function with this value, the IME adjusts the **RECONVERTSTRING** structure. The application can then pass the adjusted **RECONVERTSTRING** structure to this function with SCS_RECONVERTSTRING. The IME will not generate any **WM_IMECOMPOSITION** messages.
SCS_SETRECONVERTSTRING	Application asks the IME to reconvert the string contained in the **RECONVERTSTRING** structure.

lpComp
Pointer to the buffer that contains the updated string. The type of string is determined by the value of *dwIndex*.

dwCompLen
Length of the buffer in bytes.

lpRead
Pointer to the buffer that contains the updated string. The type of string is determined by the value of *dwIndex*. If the value of *dwIndex* is SCS_SETRRECONVERTSTRING or SCS_QUERYRECONVERTSTRING, *lpRead* will be a pointer to the **RECONVERTSTRING** structure that contains the updated reading string. If the selected IME has the value SCS_CAP_MAKEREAD, this can be NULL.

dwReadLen
Length of the buffer in bytes.

Comments

For Unicode, *dwCompLen* and *dwReadLen* specifies the length of the buffer in bytes, even if SCS_SETSTR is specified and the buffer contains a Unicode string.

SCS_SETRECONVERTSTRING or SCS_QUERYRECONVERTSTRING can be used only for IMEs that have an SCS_CAP_SETRECONVERTSTRING property. This property can be retrieved by using the **ImmGetProperty** function.

ImeToAsciiEx

The **ImeToAsciiEx** function generates a conversion result through the IME conversion engine according to the *hIMC* parameter.

UINT
ImeToAsciiEx(
UINT *uVirKey*,
UINT *uScanCode*,
CONST LPBYTE *lpbKeyState*,
LPTRANSMSGLIST *lpTransMsgList*,
UINT *fuState*,
HIMC *hIMC*
)

Parameters

uVirKey
Specifies the virtual key code to be translated. When the property bit IME_PROP_KBD_CHAR_FIRST is on, the upper byte of the virtual key is the aid character code.

For Unicode, the upper word of *uVirKey* contains the Unicode character code if the IME_PROP_KBD_CHAR_FIRST bit is on.

uScanCode
Specifies the hardware scan code of the key to be translated.

LpbKeyState
Points to a 256-byte array that contains the current keyboard state. The IME should not modify the content of the key state.

lpTransMsgList
Point to a TRANSMSGLIST buffer to receive the translated message result. This was defined as a double word buffer in Windows 95/98 and Windows NT 4.0 IME document, and the double word buffer format is [Length of the pass in translated message buffer] [Message1] [wParam1] [lParam1] {[Message2] [wParam2] [lParam2]{...{...{...}}}}.

fuState
Active menu flag.

hIMC
Input context handle.

Return Values

The return value indicates the number of messages. If the number is greater than the length of the translated message buffer, the translated message buffer is not enough. The system then checks *hMsgBuf* to get the translation messages.

Comments

On Windows 2000, a new 32bit-width virtual key code, using VK_PACKET in LOBYTE of wParam and the high word is Unicode, can be injected by using SendInput.

For ANSI IMEs that support IME_PROP_ACCEPT_WIDE_VKEY, ImeToAsciiEx may receive up to 16bit ANSI code for a character. It will be packed as below. The character is injected from SendInput API through VK_PACKET.

24-31 bit	16-23 bit	8-15 bit	0-7 biy
Reserved	Trailing DBCS byte(if any)	Leading byte	VK_PACKET

See Also

ImmToAsciiEx

ImeRegisterWord

The **ImeRegisterWord** function registers a string into the dictionary of this IME.

BOOL WINAPI
ImeRegisterWord(
LPCTSTR *lpszReading***,**
DWORD *dwStyle***,**
LPCTSTR *lpszString*
)

Parameters

lpszReading
Reading string of the registered string.

dwStyle
Style of the registered string. The following values are provided.

Value	Description
IME_REGWORD_STYLE_EUDC	String is within the EUDC range

IME_REGWORD_STYLE_USER_FIRST to IME_REGWORD_STYLE_USER_LAST	Constants range from IME_REGWORD_STYLE_USER_FIRST to IME_REGWORD_STYLE_USER_LAST and are used for private styles of the IME ISV. The IME ISV can freely define its own style. For example: #define MSIME_NOUN (IME_REGWORD_STYLE_USER_FIRST) #define MSIME_VERB (IME_REGWORD_STYLE_USER_FISRT +1)

lpszString
String to be registered.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

ImeUnregisterWord

The **ImeUnregisterWord** function removes a registered string from the dictionary of this IME.

```
BOOL WINAPI
    ImeUnregisterWord(
    LPCTSTR lpszReading,
    DWORD dwStyle,
    LPCTSTR lpszString
   )
```

Parameters

lpszReading
Reading string of the registered string.

dwStyle
Style of the registered string. Please refer to the **ImeRegisterWord** function for a description of *dwStyle*.

lpszString
String to be unregistered.

Return Values

If the function is successful, the return value is TRUE. Otherwise, the return value is FALSE.

ImeGetRegisterWordStyle

The **ImeGetRegisterWordStyle** function gets the available styles in this IME.

```
UINT WINAPI
    ImeGetRegisterWordStyle(
    UINT nItem,
    LPSTYLEBUF lpStyleBuf
   )
```

Parameters

nItem
Maximum number of styles that the buffer can hold.

lpStyleBuf
Buffer to be filled.

Return Values

The return value is the number of the styles copied to the buffer. If *nItems* is zero, the return value is the buffer size in array elements needed to receive all available styles in this IME.

ImeEnumRegisterWord

The **ImeEnumRegisterWord** function enumerates the information of the registered strings with specified reading string, style, and registered string data.

```
UINT WINAPI
    ImeEnumRegisterWord(
    hKL,
    REGISTERWORDENUMPROC lpfnEnumProc,
    LPCTSTR lpszReading,
    DWORD dwStyle,
    LPCTSTR lpszString,
    LPVOID lpData
   )
```

Parameters

hKL
Input language handle.

lpfnEnumProc
Address of callback function.

lpszReading
Specifies the reading string to be enumerated. If *lpszReading* is NULL, **ImeEnumRegisterWord** enumerates all available reading strings that match the specified *dwStyle* and *lpszString* parameters.

dwStyle
Specifies the style to be enumerated. If *dwStyle* is NULL, **ImeEnumRegisterWord** enumerates all available styles that match the specified *lpszReading* and *lpszString* parameters.

lpszString
Specifies the registered string to be enumerated. If *lpszString* is NULL, **ImeEnumRegisterWord** enumerates all registered strings that match the specified *lpszReading* and *dwStyle* parameters.

lpData
Address of application-supplied data.

Return Values

If the function is successful, the return value is the last value returned by the callback function. Its meaning is defined by the application.

Comments

If all *lpszReading dwStyle*, and *lpszString* parameters are NULL, **ImeEnumRegisterWord** enumerates all registered strings in the IME dictionary. If any two of the input parameters are NULL, **ImeEnumRegisterWord** enumerates all registered strings matching the third parameter.

ImeGetImeMenuItems

The **ImeGetImeMenuItems** function gets the menu items that are registered in the IME menu.

```
DWORD WINAPI
    ImeGetImeMenuItems(
    HIMC hIMC,
    DWORD dwFlags,
    DWORD dwType,
    LPIMEMENUITEMINFO lpImeParentMenu,
    LPIMEMENUITEMINFO lpImeMenu,
    DWORD dwSize
   )
```

Parameters

hIMC
The *lpMenuItem* contains menu items that are related to this input context.

dwFlags
Consists of the following bit combinations.

Bit	Description
IGIMIF_RIGHTMENU	If this bit is 1, this function returns the menu items for the right click Context menu.

dwType
Consists of the following bit combinations.

Bit	Description
IGIMII_CMODE	Returns the menu items related to the conversion mode.
IGIMII_SMODE	Returns the menu items related to the sentence mode.
IGIMII_CONFIGURE	Returns the menu items related to the configuration of IME.
IGIMII_TOOLS	Returns the menu items related to the IME tools.
IGIMII_HELP	Returns the menu items related to IME help.
IGIMII_OTHER	Returns the menu items related to others.
IGIMII_INPUTTOOLS	Returns the menu items related to the IME input tools that provide the extended way to input the characters.

lpImeParentMenu
Pointer to the **IMEMENUINFO** structure that has MFT_SUBMENU in *fType*. **ImeGetImeMenuItems** returns the submenu items of this menu item. If this is NULL, *lpImeMenu* contains the top-level IME menu items.

lpImeMenu
Pointer to the buffer to receive the contents of the menu items. This buffer is the array of **IMEMENUITEMINFO** structure. If this is NULL, **ImeGetImeMenuItems** returns the number of the registered menu items.

dwSize
Size of the buffer to receive the **IMEMENUITEMINFO** structure.

Return Values

The return value is the number of the menu items that were set into *lpIM*. If *lpImeMenu* is NULL, **ImeGetImeMenuItems** returns the number of menu items that are registered in the specified *hKL*.

Comments

ImeGetImeMenuItems is a new function for Windows 98 and Windows 2000.