Windows 95 / Windows NT International

Multilingual IME Specification for Applications

Version 1.19

1. Functions

1.1 List of IMM APIs to manipulate Input Context

These functions intend to manipulate Input Context. Applications use these functions if it wants to take full advantage from Chicago IME architecture. For example, if a window of an application creates Input Context, it can have its own context that will never be changed by other window. This means that even though the other windows are using IME, the window can take back previous status of the IME whenever the window gets activated.

Application that uses IME window can interact with IME through the window instance of IME window and it's an easy way to control IME. However, for those applications that want to be more user friendly don't use that user interface given by IME system. These IMM functions provide the way to interact IME through Input Context..

1.1.1 ImmCreateContext

Syntax HIMC WINAPI ImmCreateContext(*void*)

Feature This function allocate the memory for new Input Context and initialize it. An application calls this function to prepare its own Input Context.

Return value Returns the Input Context handle. If this function fail, this return value will be NULL.

1.1.2 ImmDestroyContext

Syntax BOOL WINAPI ImmDestroyContext(*hIMC*)

Parameter	Type	Description
hIMC	HIMC	The Input Context handle to be freed.

Feature This function release the Input Context and free memories.

Return value Returns TRUE if it is successful, else return FALSE.

Comments ***At the end of the application that create Input Context, this must be called for free resources for the context.***

1.1.3 ImmAssociateContext

Syntax HIMC WINAPI ImmAssociateContext(*hWnd, hIMC*)

Parameter	Type	Description

hWnd	HWND	The window handle to be associated with the context.
hIMC	HIMC	The Input Context handle given to hWnd.

Feature: This function associate the given window handle to Input Context. If hIMC is NULL, the window handle will not be associated with any Input Context. Then IME can not be used in the window.

Return value: Returns previous Input Context assigned to hWnd.

Note: An application calls this function to associate an Input Context to the window handle. Unless an application calls this function, system automatically gives default Input Context to the window handle. Before destroy Input Context associated to a window, this window have to re-associate to another Input context; typically, this different Input Context is original/default Input Context for a window.

1.1.4 ImmGetContext

Syntax: HIMC WINAPI ImmGetContext(*hWnd*)

Parameter	**Type**	**Description**
hWnd	HWND	Window handle to retrieve Input Context.

Feature: This function retrieves the Input Context that is assigned to given hWnd.

Return: Handle of an Input Context associated with hWnd.

Note: Each time an application accesses the contents of current Input Context, it should call this function first.

1.1.5 ImmGetCompositionString

Syntax: LONG WINAPI ImmGetCompositionString(*hIMC, dwIndex, lpBuf, dwBufLen*);

Parameter	**Type**	**Description**
hIMC	HIMC	Input Context handle
dwIndex	DWORD	One of followings

Value of *dwIndex*	Description
GCS_COMPSTR	To retrieve current composition string. This API fills lpBuf with composition string.
GCS_COMPATTR	To retrieve attribute of the composition string. This API fills lpBuf with attribute info.

GCS_COMPCLAUSE	To retrieve clause information of the composition string. This API fills lpBuf with clause information.
GCS_COMPREADSTR	To retrieve reading string of current composition. This API fills lpBuf with reading string of current composition.
GCS_COMPREADATTR	To retrieve attribute of reading string of current composition. This API fills lpBuf with attribute info.
GCS_COMPREADCLAUSE	To retrieve clause info of reading string of the composition string. This API fills lpBuf with the clause info.
GCS_CURSORPOS	To retrieve cursor position in the composition string. This API returns the cursor position.
GCS_DELTASTART	To retrieve start position of changing in the composition string. This API returns the position in the return of value.
GCS_RESULTSTR	To retrieve the string of composition result. This API fills lpBuf with composition result .
GCS_RESULTCLAUSE	To retrieve clause info of the result string. This API fills lpBuf with the clause info.
GCS_RESULTREADSTR	To retrieve the reading string. This API fills lpBuf with the string.
GCS_RESULTREADCLAUSE	To retrieve clause info of the reading string. This API fills lpBuf with the clause info.

lpBuf	LPVOID	Pointer to the buffer to be filled.
dwBufLen	DWORD	Length of the buffer.

Return value

Actual number of bytes count copied to lpBuf. If wBufLen is NULL, the function returns the length of the buffer required, if string case, this return value does not include the null terminating character. if the return value is <0 it will be one of followings.

WindowsNT-Unicode: dwBufLen specifies the size in bytes of the buffer, even if the buffer contains a unicode string. The return value of this function is always size in bytes, even if requested data is unicode string.

IMM_ERROR_NODATA The composition data retrieved isn't ready in the Input Context.

IMM_ERROR_GENERAL General error has been reported by IME.

Note: An application should call this API in response with WM_IME_COMPOSITION. IMM will remove the information when an application calls ImmReleaseContext.The format of the attribute information.
The attribute information is a single-byte array and specifies the attribute of string. The contents are as follows:

Value	Content
	Specifies the status of composition string.
ATTR_INPUT	Character currently being entered
ATTR_TARGET_CONVERTED	Character currently being converted (already converted)
ATTR_CONVERTED	Character given from conversion
ATTR_TERGET_NOTCONVERTED	Character currently being converted (yet to be converted)
ATTR_INPUT_ERROR	Charecter is error character and can not be converted by IME.
Other than above:	Reserved

Each content is as follows:

Character currently being entered: (ATTR_INPUT - 0x00)

The character the user is entering. In Japanese case, this character is a hiragana, katakana, or alphanumeric, which is yet to be converted by the IME. In Korean case, this character is Hangeul characters, which is not converted by IME yet. In Traditional and Simplified Chinese case, each IME may limit its character in some range.

Character currently being converted (already converted):
(ATTR_TARGET_CONVERTED - 0x01)

The character that has been selected for conversion by the user and converted by the IME.

Character given from conversion: (ATTR_CONVERTED - 0x02)

The character to which the IME has converted.

Character currently being converted (yet to be converted):
(ATTR_TARGET_NOTCONVERTED - 0x03)

The character that has been selected for conversion by the user and not yet converted by the IME. In Japanese case, this character is a hiragana, katakana, or alphanumeric, which the user has entered.

Charecter is error character and can not be converted by IME:
(ATTR_INPUT_ERROR - 0x04)

The character is an error character, the IME can not convert this character. For example, some consonants can not put together.

The length of the attribute information is same as the length of the string. Each byte corresponds to each byte of the string. Even if the string includes DBCS characters, the attribute information has the information bytes of both the lead byte and the second byte.

Windows NT-Unicode: The length of the attribute information is same as the length of the unicode string. Each byte corresponds to each unicode character of the string.

The format of the clause information.

The clause information is a double word array and specifies the numbers that are the positions of the clause. The position of the clause is one of a position of composition string and this clause starts from this position. At least, this length of information is two double words. This means the length of the clause information is four. The first double word has to be 0. This is the start position of the first clause. The last double word has to be the length of this string. For example, if the string has three clauses, the clause information has four double words. The first double word is 0. The second double word specifies the start position of the second clause. The third double word specifies the start position of the third clause. The last double word is the length of this string.

Windows NT-Unicode: "The position of a clause" is position counted in unicode characters. "the length of this string" means the size in unicode characters, not in bytes.

The rule of the cursor position.

This value indicates at what character in the composition string the cursor is, in terms of the count of that character. The counting starts at 0. If the cursor is to be positioned immediately after the composition string, this value shall be equal to the length of the composition string. In case there is no cursor (if such a condition exists), a value -1 is specified here. If an composition string does not exist, this member is invalid.

Windows NT-Unicode: "Position" and "length" is counted in unicode characters.

1.1.6 ImmSetCompositionString

Syntax BOOL WINAPI ImmSetCompositionString(*hIMC, dwIndex, lpComp, dwCompLen, lpRead, dwReadLen*);

Feature An application uses this function to fully control the appearance of the composition string.

Parameter	**Type**	**Description**
hIMC	HIMC	Input Context handle
dwIndex	DWORD	One of followings

Value of *dwIndex*	Description
SCS_SETSTR	
SCS_CHANGEATTR	
SCS_CHANGECLAUSE	

lpComp	LPCVOID	A pointer to the buffer that contains the updated string; the type of string determined by the value of dwIndex.
dwCompLen	DWORD	Length of the buffer.
lpRead	LPCVOID	A pointer to the buffer that contains the updated string; the type of string determined by the value of dwIndex.
dwReadLen	DWORD	Length of the buffer.

Windows NT-Unicode: dwCompLen and dwReadLen specifies the length of the buffer in bytes, even SCS_SETSTR is specified and the buffer contains unicode string.

Return value TRUE if successful otherwise FALSE.

Note The application can set lpComp, lpRead or both. If the application does not specify lpComp, lpComp have to be NULL and dwCompLen have to be 0.

1.1.7 ImmGetCompositionFont

Syntax BOOL ImmGetCompositionFont(*hIMC, lplf*)

Feature An application calls this function to get logfont that should be displayed at composition window.

Parameter	Type	Description
hIMC	HIMC	Specify the handle of Input Context.
lplf	LPLOGFONT	Specify the pointer to the logfont which is get from the context.

Return Value TRUE if successful, otherwise FALSE.

1.1.8 ImmSetCompositionFont

Syntax BOOL ImmSetCompositionFont(*hIMC, lplf*)

Feature An application calls this function to specify logfont that should be displayed at composition window.

Parameter	Type	Description
hIMC	HIMC	Specify the handle of Input Context.
lplf	LPLOGFONT	Specify the pointer to the logfont which is given to the context.

Return Value TRUE if successful, otherwise FALSE.

Note According to this information, the IME will change its behavior such as handling arrow keys. Even it is an application which never uses composition window

provided by IME, especially for vertical writing it is necessary to give an information to the IME about logical font that is displayed on screen.
This function generates WM_IME_NOTIFY with IMN_SETCOMPOSITIONFONT to application.

1.1.9 ImmGetCandidateListCount

Syntax DWORD ImeGetCandidateListCount(*hIMC, lpdwListCount*)

Parameter	Type	Description
hIMC	HIMC	Specify the handle of Input Context.
lpdwListCount	LPDWORD	The pointer to the buffer to receive number of candidate lists.

Return Value Number of bytes required to receive all candidate list if successful, otherwise zero.

Note An application should call this API in response with WM_IME_OPENCANDIDATE / WM_IME_CHANGECANDIDATE messages.

1.1.10 ImmGetCandidateList

Syntax DWORD WINAPI ImmGetCandidateList(*hIMC, dwIndex, lpCandidate, dwBufLen*)

Parameter	Type	Description
hIMC	HIMC	Specifies the handle of Input Context.
dwIndex	DWORD	Specifies an index of candidate list. This starts from 0 to n - 1.
lpCandList	LPCANDIDATELIST	The buffer filled with this function
dwBufLen	DWORD	Specifies the length of the buffer. If this parameter is NULL, the function returns the number of bytes required for the candidate list.

Return value Actual number of bytes that is copied into the buffer specified with *lpCandList*, if it returns successful, otherwise zero.

1.1.11 ImmReleaseContext

Syntax BOOL ImmReleaseContext(*hWnd, hIMC*);

Parameter	Type	Description
hWnd	HWND	Window handle that retrieved Input Context.
hIMC	HIMC	Handle of Input Context.

Return TRUE if successful. Otherwise FALSE.

Note: An application must call ImmReleaseContext for each call to ImmGetContext. This function also unlock the memory object prepared in Input Context.

1.1.12 ImmGetConversionStatus

Syntax BOOL WINAPI ImmGetConversionStatus(*hIMC, lpfdwConversion, lpfdwSentence*)

Feature Gets current conversion status.

Parameter	Type	Description
hIMC	HIMC	The context handle to retrieve the information.
lpfdwConversion	LPDWORD	a long pointer to the DWORD buffer to receive conversion mode.
lpfdwSentence	LPDWORD	a long pointer to the DWORD buffer to receive sentence mode.

Return value TRUE if successful, otherwise FALSE.

conversion mode includes following flag bits.

bits	
IME_CMODE_KATAKANA	means KATAKANA mode if this bit is 1. means HIRAGANA mode if this bit is 0.
IME_CMODE_NATIVE IME_CMODE_CHINESE IME_CMODE_HANGEUL IME_CMODE_JAPANESE	means NATIVE mode if this bit is 1. means ALPHANUMERIC mode if this bit is 0. Even the definitions for these constants are the same. It is better that an application just use IME_CMODE_NATIVE.
IME_CMODE_FULLSHAPE	means full shape mode if this bit is 1. means half shape mode if this bit is 0. .
IME_CMODE_ROMAN	ROMAN input mode if this bit is 1. Non ROMAN input mode if this bit is 0.
IME_CMODE_CHARCODE	Character code input mode if this bit is 1.

	Non character code input mode if this bit is 0.
IME_CMODE_HANJACONVERT	HANJA convert mode if this bit is 1. Non HANJA convert mode if this bit is 0.
IME_CMODE_SOFTKBD	Soft Keyboard mode if this bit is 1. Non Soft Keyboard mode if this bit is 0.
IME_CMODE_NOCONVERSION	Any conversion won't be processed by IME if this bit is ON. This is similar to CLOSE mode of IME.
IME_CMODE_EUDC	EUDC conversion mode if this bit is 1. Non EUDC conversion mode if this bit is 0.
IME_CMODE_SYMBOL	SYMBOL conversion mode if this bit is 1. Non SYMBOL conversion mode if this bit is 0.
Other bits	Reserved.

sentence mode includes following flag.

bits	Sentence mode
0 - 15	Can be one of the following value IME_SMODE_NONE No information for sentence. IME_SMODE_PLAURALCLAUSE The IME uses plaural clause information to carry our conversion processing. IME_SMODE_SINGLECONVERT The IME carries out conversion processing in single character mode. IME_SMODE_AUTOMATIC The IME carries out conversion processing in automatic mode. IME_SMODE_PHRASEPREDICT The IME uses phrase information to predict the next chracter.

16 - 31	Reserved for IME private use. IME can make use of these bit to communicate with its UI portion.

1.1.13 ImmGetConversionList

Syntax DWORD WINAPI ImmGetConversionList(*hKL, hIMC, lpSrc, lpDst, dwBufLen, uFlag*)

Feature Obtain the list of FE character or word from one character or word.

Parameter	Type	Description
hKL	DWORD	The *hKL* of one IME.
hIMC	HIMC	The handle of the Input Context that has private area for the IME.
lpSrc	LPCTSTR	Pointer to character string which is ended with null character.
lpDst	LPCANDIDATELIST	Pointer to the buffer where the result of conversion is stored.
dwBufLen	DWORD	Length of the buffer. If this parameter is NULL, the function returns the length of buffer required for *lpDst*.
uFlag	UINT	Currently it can be one of following three. GCL_CONVERSION specifies reading string to lpSrc and the IME returns the result string in the *lpDst*. GCL_REVERSECONVERSION specifies the result string in *lpSrc* the IME returns the reading string in the *lpDst*. GCL_REVERSE_LENGTH specifies the result string in *lpSrc* the IME returns the length that it can hanlde on GCL_REVERSECONVERSION call. For example, one IME can not reverse convert a result string with sentence period to a reading string. So it returns the string length in bytes without the sentence period.

Return value Number of bytes of result string list copied to the buffer.

1.1.14 ImmGetDefaultIMEWnd

Syntax HWND WINAPI ImmGetDefaultIMEWnd(*hWnd*)

Feature Gets the default window handle of the IME class.

Parameter	Type	Description
hWnd	HWND	The window handle of the application.

Return value Return the default window handle of IME class, NULL - failure.

Notes The system will create the default IME window for every thread. The IME window is created based on the IME class. The application can send WM_IME_CONTROL to this window.

1.1.15 ImmGetOpenStatus

Syntax BOOL WINAPI ImmGetOpenStatus(*hIMC*)

Feature Gets the open or close status of the IME.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.

Return value TRUE - open, FALSE - close.

1.1.16 ImmSetConversionStatus

Syntax BOOL WINAPI ImmSetConversionStatus(*hIMC, fdwConversion, fdwSentence*)

Feature Sets current conversion status.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.
fdwConversion	DWORD	Conversion status flags.
fdwSentence	DWORD	Sentence mode of IME.

Return value TRUE - successful, FALSE - failure.

Notes This API is called by IME and application. This function generates WM_IME_NOTIFY with IMN_SETCONVERSIONSTATUS.

1.1.17 ImmSetOpenStatus

Syntax BOOL WINAPI ImmSetOpenStatus(*hIMC, fOpen*)

Feature Opens or closes the IME.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.

fOpen	BOOL	Specifies whether to open or close the IME.

Return value TRUE - successful, FALSE - failure.

Notes This API is called by IME and application. This function generates WM_IME_NOTIFY with IMN_SETOPENSTATUS.

1.1.18 ImmGetStatusWindowPos

Syntax BOOL WINAPI ImmGetStatusWindowPos(*hIMC, lpptPos*)

Feature Gets the position of the status window. The dimensions are given in screen coordinates, relative to the upper-left corner of the display screen.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.
lpptPos	LPPOINT	Pointer to the buffer for receiving the position.

Return value TRUE - successful, FALSE - failure.

1.1.19 ImmSetStatusWindowPos

Syntax BOOL WINAPI ImmSetStatusWindowPos(*hIMC, lpptPos*)

Feature Sets the position of the status window. The coordinates are given in screen coordinates, relative to the upper-left corner of the display screen.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.
lpptPos	LPPOINT	Specifies the new position of the status window.

Return value TRUE - successful, FALSE - failure.

Notes This API is called by IME and application. This function generates WM_IME_NOTIFY with IMN_SETSTATUSWINDOWPOS.

1.1.20 ImmGetCompositionWindow

Syntax BOOL WINAPI ImmGetCompositionWindow(*hIMC, lpCompFrom*)

Feature Gets the information of the composition window.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.

lpCompForm	LPCOMPOSITIONFORM	Pointer to the buffer for receiving the information of the composition window.

Return value TRUE - successful, FALSE - failure.

1.1.21 ImmSetCompositionWindow

Syntax BOOL WINAPI ImmSetCompositionWindow(*hIMC, lpCompForm*)

Feature Sets the position of the composition window.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.
lpCompForm	LPCOMPOSITIONFORM	Specifies the new position and other related information to the composition window.

Return value TRUE - successful, FALSE - failure.

Notes This API is called by IME and application. This function generates WM_IME_NOTIFY with IMN_SETSCOMPOSITIONWINDOW.

1.1.22 ImmGetCandidateWindow

Syntax BOOL WINAPI ImmGetCandidateWindow(*hIMC, dwIndex, lpCandFrom*)

Feature Gets the information of the candidate window that is specified by dwIndex.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.
dwIndex	DWORD	The index of the candidate window.
lpCandForm	LPCANDIDATEFORM	Pointer to the buffer for receiving the information of the candidate window.

Return value TRUE - successful, FALSE - failure.

1.1.23 ImmSetCandidateWindow

Syntax BOOL WINAPI ImmSetCandidateWindow(*hIMC, lpCandForm*)

Feature Sets the position of the composition window.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.

lpCandForm	LPCANDIDATEFORM	Specifies the new position and other related information to the candidate window.

Return value TRUE - successful, FALSE - failure.

Notes This API is called by IME and application. This function generates WM_IME_NOTIFY with IMN_SETCANDIDATEPOS.

1.1.24 ImmNotifyIME

Syntax BOOL WINAPI ImmNotifyIME(*hIMC, dwAction, dwIndex, dwValue*)

Feature An application uses this function to notify IME about changing status of Input Context.

Parameter	**Type**	**Description**
hIMC	HIMC	The handle of the Input Context.
dwAction	DWORD	This parameter can be one of the follows.

dwAction	**Description**
NI_OPENCANDIDATE	An application makes the IME open the candidate list. Then if IME accept to open the candidate list, the application will receive WM_IME_NOTIFY (subfunction is IMN_OPENCANDIDATE) message.
dwIndex	an index of a candidate list to be opened.
dwValue	Not Used.
NI_CLOSECANDIDATE	An application makes the IME close the candidate list. Then if IME accept to close the candidate list, the application will receive WM_IME_NOTIFY (subfunction is IMN_CLOSECANDIDATE) message.
dwIndex	an index of a candidate list to be closed.
dwValue	Not Used.
NI_SELECTCANDIDATESTR	An application select one of candidates
dwIndex	an index of a candidate list to be selected.
dwValue	an index of a candidate string in the selected candidate list.

NI_CHANGECANDIDATELIST An application change the current selected candidate.

dwIndex an index of a candidate list to be selected.

dwValue Not Used.

NI_COMPOSITIONSTR An application makes the effect for IME about the composition string. This action is affected when there is the composition string in the input context.

dwIndex One of following values.

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 composition string.

CPS_CANCEL

To clear composition string and set the status as no composition string.

dwValue Not Used.

NI_SETCANDIDATE_PAGESTART	
	An application changes the page starting index of a candidate list.
dwIndex	an index of a candidate list to be changed. (0 - 31)
dwValue	New page start index.
NI_SETCANDIDATE_PAGESSIZE	
	An application change the pages size of a candidate list.
dwIndex	an index of a candidate list to be changed. (0 - 31)
dwValue	New page size.

Return value TRUE if succeeds otherwise FALSE.

1.1.25 ImmEscape

Syntax LRESULT WINAPI ImmEscape(*hKL, hIMC, uEscape, lpData*)

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

Parameter	**Type**	**Description**
hKL	HKL	The HKL of one IME.
hIMC	HIMC	The handle of Input Context.
uEscape	UINT	Specifies the escape function to be performed.
lpData	LPVOID	Points to the data required for the specified escape.

uEscape	**Meaning**
IME_ESC_QUERY _SUPPORT	Checks for implementation. If this escape is not implemented, the return value is zero.
IME_ESC_RESERVED_FIRST	The escape which is between IME_ESC_RESERVED_FIRST and IME_ESC_RESERVED_LAST is reserved by system.
IME_ESC_RESERVED_LAST	The escape which is between IME_ESC_RESERVED_FIRST and IME_ESC_RESERVED_LAST is reserved by system.
IME_ESC_PRIVATE_FIRST	The escape which is between IME_ESC_PRIVATE_FIRST and

	IME_ESC_PRIVATE_LAST is reserved for the IME, IME can freely use these escape functions for its own purposes.
IME_ESC_PRIVATE_LAST	The escape which is between IME_ESC_PRIVATE_FIRST and IME_ESC_PRIVATE_LAST is reserved for the IME, IME can freely use these escape functions for its own purposes.
IME_ESC_SEQUENCE_TO_INTERNAL	The escape is for Chinese specific. An application wants to run under all Far East platforms should not use it. It is for Chinese EUDC editor. The *(LPDOWRD)lpData is the sequence code and the return value is the character code for this sequence code. Normally, the Chinese IME will encode its reading character codes into sequence 1 to n.
IME_ESC_GET_EUDC_DICTIONARY	The escape is for Chinese specific. An application wants to run under all Far East platforms should not use it. It is for Chinese EUDC editor. On return from the function, the (LPTSTR)lpData is filled with the full path file name of EUDC dictionary. The size of this buffer pointed by lpData should be greater or equal to 80 * sizeof(TCHAR).
IME_ESC_SET_EUDC_DICTIONARY	The escape is for Chinese specific. An application wants to run under all Far East platforms should not use it. It is for Chinese EUDC editor. On return from the function, the (LPTSTR)lpData is the full path file name of EUDC dictionary. The path name should be less than 80 * sizeof(TCHAR).
IME_ESC_MAX_KEY	The escape is for Chinese specific. An application wants to run under all Far East platforms should not use it. It is for Chinese EUDC editor. The return value is the maximum key stokes for an EUDC character.
IME_ESC_IME_NAME	The escape is for Chinese specific. An application wants to run under all Far East platforms should not use it. It is for Chinese EUDC editor. On return from the function, the (LPTSTR)lpData is IME name to be displayed on EUDC editor. The size of this buffer pointed by lpData should be greater or equal to 16 * sizeof(TCHAR).

IME_ESC_HANJA_MODE	The escape is for Korean specific. An application wants to run under all Far East platforms should not use it. It is for conversion from Hangeul to Hanja. The input parameter (LPSTR)lpData is filled with Hangeul character which will be converted to Hanja, and it's null terminated string. When the application want to convert any Hangeul character to Hanja character in the same method of Hanja conversion with composition character is present, the application just request this function and IME set itself as Hanja conversion mode.

Return Value Returns zero if failure, otherwise the return value depend on each escape function.

1.1.26 ImmGetGuideLine

Syntax DWORD WINAPI ImmGetGuideLine(*hIMC,dwIndex, lpBuf,dwBufLen*)

Feature Gets the guide line information reported by the IME.

Parameter	Type	Description
hIMC	HIMC	The handle of the Input Context.
dwIndex	DWORD	The Indies to get the information of GuideLine. That defined as follows.
	GGL_LEVEL	The return value is the level of GuideLine.
	GGL_INDEX	The return value is the index of GuideLine.
	GGL_STRING	To get the string for GuideLine
lpBuf	LPTSTR	The pointer of the string for GuideLine. When dwIndex is GGL_STRING, this is acceptable. Otherwise this have to be NULL. It can be NULL when *dwBufLen* is 0.
dwBufLen	DWORD	Specifies the length of the buffer to get the string for GuideLine. If this parameter is NULL the function returns the length required (not including NULL terminator). When dwIndex is GGL_STRING, this is acceptable. Otherwise this have to be 0.

Windows NT-Unicode: dwBufLen specifies the size in bytes of the buffer

Return values

dwIndex is GGL_LEVEL.

dwIndex is GGL_LEVEL	Meaning

GL_LEVEL_NOGUIDELINE	There is no guideline. If old guideline is shown, UI should hide old guideline.
GL_LEVEL_FATAL	The fatal error occurs. Some data may be lost.
GL_LEVEL _ERROR	The error occurs. The handling may not be continued.
GL_LEVEL _WARNING	IME warns for user. The unexpected thing occurs, but IME can continue to handle.
GL_LEVEL _INFORMATION	The information for user.

dwIndex is GGL_INDEX.

dwIndex is GGL_INDEX	**Meaning**
GL_ID_UNKNOWN	Unknown Error. The application should refer Error String.
GL_ID_NOMODULE	IME can not find the module that IME needs.
GL_ID_NODICTIONARY	IME can not find the dictionary or the dictionary is strange.
GL_ID_CANNOTSAVE	Dictionary or the statistic data can not be saved.
GL_ID_NOCONVERT	IME can not convert any more.
GL_ID_TYPINGERROR	Typing error. IME can not handle this typing.
GL_ID_TOOMANYSTROKE	There are two many strokes for one character or one clause.
GL_ID_READINGCONFLICT	For example, some vowels can not put together for one character.
GL_ID_INPUTREADING	IME prompts the end user - now it is in inputting reading charcater state.
GL_ID_INPUTRADICAL	IME prompts the end user - now it is in inputting radical charcater state.
GL_ID_INPUTCODE	IME prompts the end user - now it is in inputting charcater code state.
GL_ID_CHOOSECANIDATE	IME prompts the end user - now it is in choosing candidate string state.
GL_ID_REVERSECONVERSION	IME prompts the user end - the information of reverse conversion. The information of reverse conversion can be got by ImmGetGuideLine(*hIMC*, GGL_PRIVATE. *lpBuf*, *dwBufLen*).The information filled in *lpBuf* is in CANDIDATELIST format.

dwIndex is GGL_STRING.

If dwBufLen is NULL, the return value is the length required (not including NULL terminator). Otherwise the return value specify the byte number to be copied into *lpB*

Windows NT-Unicode: If dwBufLen is 0, the reutrn value is the size in bytes required(not including unicode NULL terminator)

dwIndex is GGL_PRIVATE.

If dwBufLen is NULL, the return value is the length required. Otherwise the return value specify the byte number to be copied into *lpBuf*.

Notes The application can call this API on receiveing WM_IME_NOTIFY/IMN_GUIDELINE message.

1.2 List of IMM APIs to manipulate IME layout

1.2.1 ImmConfigureIME

Syntax BOOL WINAPI ImmConfigureIME(*hKL, hWnd, dwMode, lpData*)

Feature Brings up the dialog of the IME with the specified hKL.

Parameter	Type	Description
hKL	HKL	The HKL of one IME.
hWnd	HWND	The window handle of application. When IME show the its configuration dialog box, this window handle can be used as the parent window.
dwMode	DWORD	The mode of the dialog. That defined as follow.
	IME_CONFIG_GENERAL	The dialog for general purpose configuration.
	IME_CONFIG_REGWORD	The dialog for register word.
	IME_CONFIG_SELECTDICTIONARY	The dialog for selecting dictionary of IME.
lpData	LPVOID	The pointer to supplemental data when an application specifies IME_CONFIG_REGWORD to dwMode. Win95 only supports REGISTERWORD structure for this data. When an application doesn't specify IME_CONFIG_REGWORD, this parameter will be ignored.

Return value TRUE - successful, FALSE - failure.

Notes When an application specifies IME_CONFIG_REGWORD to dwMode, this API can take a pointer to data structure. Currently only REGISTERWORD structure is defined for the purpose. Using REGISTERWORD structure, an application can specify initial value for the configuration dialogbox of an IME..

See Also REGISTERWORD

1.2.2 ImmGetDescription

Syntax UINT WINAPI ImmGetDescription(*hKL, lpszDescription, uBufLen*)

Feature Gets the description of the IME with the specified HKL.

Parameter	Type	Description
hKL	HKL	The HKL of one IME.
lpszDescription	LPTSTR	The description of the IME to be filled.
uBufLen	UINT	Specifies the length of the buffer. If this parameter is NULL the function returns the length required (not including NULL terminator).
Windows NT-Unicode:		uBufLen specifies the size of the buffer in unicode characters. If this parameter is 0 , the function returns the size in unicode characters of the buffer required (not including unicode NULL terminator)

Return value Number of bytes (not including NULL terminator) copied into the buffer given with lpszDescription, otherwise zero.

Windows NT-Unicode: Return value of this function is number of unicode characters not including unicode NULL terminator copied into the buffer.

Notes General purpose API.

1.2.3 ImmGetIMEFileName

Syntax BOOL WINAPI ImmGetIMEFileName(*hKL, lpszFileName, uBufLen*)

Feature Gets the file name of the IME with the specified hKL.

Parameter	Type	Description
hKL	HKL	The HKL of one IME.
lpszFileName	LPTSTR	The buffer for the filename of the IME. It can be NULL when *uBufLen* is NULL.
uBufLen	UINT	Specifies the length of the buffer. If this parameter is NULL the function returns the length required (not including NULL terminator).
Windows NT-Unicode:		uBufLen specifies the length in unicode characters of the buffer.

Return value Number of bytes copied (not including NULL terminator) into the buffer given with *lpszFileName*, otherwise zero.

Windows NT-Unicode: Number of unicode characters copied (not including a unicode NULL terminator) into the buffer given with lpszFileName, otherwize zero.

Notes General Purpose API. The file comes from the registry - HKEY_LOCAL_MACHINE \ System \ CurrentControlSet \ control \ keyboard layouts \ <hKL> under the value name - IME file.

1.2.4 ImmGetProperty

Syntax DWORD WINAPI ImmGetProperty (*hKL, fdwIndex*)

Feature Gets the property and capability of the IME with the specified HKL.

Parameter	Type	Description
hKL	HKL	The HKL of one IME.
fdwIndex	DWORD	The IME property information ID.

fdwID	Meaning
IGP_PROPERTY	This API returns the property information.
IGP_CONVERSION	This API returns the capability of the conversion.
IGP_SENTENCE	This API returns the capability of the sentence mode.
IGP_UI	This API returns the UI capability. UI_CAP_xxx
IGP_SETCOMPSTR	This API returns the capability of ImmSetCompositionString API. SCS_CAP_xxxx
IGP_SLECT	This API returns the capability of the inheritance at ImeSelect time. SELECT_CAP_xxx
IGP_GETIMEVERSION	This API returns the version of Windows that IME is created for.

Return value The property or capability.

Notes General Purpose API. Refer to the IMEINFO Structure for the *lpdwProperty*, *lpdwConvCapability*, and *lpfdwSentenceCapability*.

The meaning of the bit of return value. dwIndex is IGP_PROPERTY

Properties	Description
IME_PROP_AT_CARET	This bit on indicates IME conversion window is at caret position. This bit off indicates a near caret position operation IME.
IME_PROP_SPECIAL_UI	This bit on indicates IME having a special UI. The application should not draw the IME User Interface by itself.

IME_PROP_CANDLIST_START_FROM_1	This bit on indicates the UI of candidate list string start from 0 or 1. Application can draw the candidate list string by adding the “1”, “2”, “3”, or etc in front of the candidate string.
IME_PROP_UNICODE	This bit on indicates the string contents of the input context is in UNICODE or not.

The meaning of the bit of return value. dwIndex is IGP_UI

Properties	**Description**
UI_CAP_2700	The UI support when escape of LogFont is 0 or 2700.
UI_CAP_ROT90	The UI support when escape of LogFont is 0, 900, 1800 or 2700.
UI_CAP_ROTANY	The UI support any escape.

The meaning of the bit of return value. dwIndex is IGP_SETCOMPSTR

Properties	**Description**
SCS_CAP_COMPSTR	This IME can generate the composition string by SCS_SETSTR.
SCS_CAP_MAKEREAD	When calling ImmSetCompositionString with SCS_SETSTR, the IME can create the reading of composition string without lpRead. Under IME that has this capability, the application does not need to set lpRead for SCS_SETSTR.

The meaning of the bit of return value. dwIndex is IGP_SELECT

Properties	**Description**
SELECT_CAP_CONVMODE	The IME has the capability of inheritance of conversion mode at ImeSelect()
SELECT_CAP_SENTENCE	The IME has the capability of inheritance of sentence mode at ImeSelect()

This capability is for the application. When the end user change the IME, the application can know the conversion mode will be inherited or not by seeing this capability. If the new selected IME does not have this caps, the application can not expect the new mode and it have to get the mode again

The meaning of the return value. dwIndex is IGP_GETIMEVERSION

Value	**Description**
IMEVER_0310	The IME was created for Window 3.1.
IMEVER_0400	The IME was created for Window 95.

1.2.5 ImmInstallIME

Syntax HKL WINAPI ImmInstallIME(*lpszIMEFile, lpszLayoutText*)

Feature Install an IME into the system.

Parameter	**Type**	**Description**
lpszIMEFile	LPCTSTR	The full path name of the IME.

lpszLayoutText	LPCTSTR	The layout text of the IME, it is the name of the IME.

Return value The HKL of this IME.

Notes The API is used by setup program of an IME only.

1.2.6 ImmIsIME

Syntax BOOL WINAPI ImmIsIME(*hKL*)

Feature Checks whether this HKL is a HKL of IME or not. The multilingual application can know whether it should mind IME or not by calling this function.

Parameter	Type	Description
hKL	HKL	A HKL to be checked.

Return value TRUE - it is a HKL of one IME, else - FALSE.

1.3 List of IMM APIs to manipulate IME hot keys

The IME hot key is for changing IME input mode or switching the IME. The IME hot key for switching directly to an IME is called direct switch hot key. The direct switch hot key is range from IME_HOTKEY_DSWITCH_FIRST to IME_HOTKEY_DSWITCH_LAST. It is registered by an IME if the IME wants to has such a hot key. The IME hot key is effective in all IME and no matter which IME is active. In Chicago-FE, several predefined hot key functionality are defined by IMM. The IMM itself provides the functionality (different handling routines) of those hot key functions. Every hot key functionality in Chicago-FE has a different hot key ID in IMM, each ID has it own functionality according to requirements of each country. Application has no way to add another predefined hot key ID into the system.

The predefined hot key IDs are -

Hot Key ID	Description
IME_CHOTKEY_IME_NONIME_TOGGLE	The hot key of Windows for Simplified Chinese Edition, this hot key toggle between IME and non IME.
IME_CHOTKEY_SHAPE_TOGGLE	The hot key of Windows for Simplified Chinese Edition, this hot key toggle the shape conversion mode of IME.
IME_CHOTKEY_SYMBOL_TOGGLE	The hot key of Windows for Simplified Chinese Edition, this hot key toggle the symbol coversion mode of IME. The symbol mode indicates that user can input Chinese punctuation and symbols (full shape chars) by mapping it to the punctuation and symbol keystrokes of keyboard.
IME_JHOTKEY_CLOSE_OPEN	The hot key of Windows for Japanese Edition, this hot key toggle between close and open.

IME_THOTKEY_IME_NONIME_TOGGLE	The hot key of Windows for (Traditional) Chinese Edition this hot key toggle between IME and non IME.
IME_THOTKEY_SHAPE_TOGGLE	The hot key of Windows for (Traditional) Chinese Edition, this hot key toggle the shape conversion mode of IME.
IME_THOTKEY_SYMBOL_TOGGLE	The hot key of Windows for (Traditional) Chinese Edition, this hot key toggle the symbol coversion mode of IME.

1.3.1 ImmSimulateHotKey

Syntax BOOL WINAPI ImmSimulateHotKey(*hWnd, dwHotKeyID*)

Feature This API simulates the same function as this IME global hot key being pressed.

Parameter	Type	Description
hWnd	HWND	The window handle of the application.
dwHotKeyID	DWORD	The IME global hot key ID.

Return value TRUE - successful, FALSE - failure.

Notes This API only can called from applications.

1.4 List of misc IMM APIs

1.4.1 ImmGetVirtualKey

Syntax UINT WINAPI ImmGetVirtualKey(*hWnd*)

Feature This API get the real virtual key which is preprocessed by an IME.

Parameter	Type	Description
hWnd	HWND	The window handle that receive the key message.

Return value The real virtual key value.

Notes After a need key input message preprocessed by an IME, the virtual key value is changed to VK_PROCESSKEY. This API only can be called by an application which get a virtual key value - VK_PROCESSKEY and want to know the read virtual key.

1.4.2 ImmIsUIMessage

Syntax BOOL WINAPI ImmIsUIMessage(*hWndIME, msg, wParam, lParam*)

Feature This API filters the messages needed for IME window and send it to hWndIME if necessary.

Parameter	Type	Description
hWndIME	HWND	The handle of IME system class widow.
msg	UINT	message to be filtered.
wParam	WPARAM	wParam of the message
lParam	LPARAM	lParam of the message

Return value TRUE - the message is processed by IME User Interface. FALSE - the message is not processed by IME User Interface.

Notes Typically, an application uses this function to display any composition string or candidate list given by IME. If the hWndIME is NULL, the API checks whether this is a IME User Interface message.

1.4.3 ImmRegisterWord

Syntax BOOL WINAPI ImmRegisterWord(*hKL, lpszReading, dwStyle, lpszString*)

Feature This API register a string into the dictionary of the IME of this hKL.

Parameter	Type	Description
hKL	HKL	Specifies the IME.
lpszReading	LPCTSTR	The reading string of the register string.
dwStyle	DWORD	The style of the register string. It includes - IME_REGWORD_STYLE_EUDC: The string is in EUDC range. IME_REGWORD_STYLE_USER_FIRST, IME_REGWORD_STYLE_USER_LAST: The constants range from IME_REGWORD_STYLE_USER_FIRST to IME_REGWORD_STYLE_USER_LAST are for private styles of the IME ISV. IME ISV can define it own style freely. For ex. #define MSIME_NOUN (IME_REGWORD_STYLE_USER_FIRST) #define MSIME_VERB (IME_REGWORD_STYLE_USER_FIRST +1)
lpszString	LPCTSTR	The string to be registered.

Return value TRUE if the function is successful, FALSE if not.

1.4.4 ImmUnregisterWord

Syntax BOOL WINAPI ImmUnregisterWord(*hKL, lpszReading, dwStyle, lpszString*)

Feature This API remove a register string from the dictionary of the IME of this hKL.

Parameter	Type	Description
hKL	HKL	Specifies the IME.

lpszReading	LPCTSTR	The reading string of the register string.
dwStyle	DWORD	The style of the register string. It includes - IME_REGWORD_STYLE_EUDC : The string is in EUDC range. IME_REGWORD_STYLE_USER_FIRST, IME_REGWORD_STYLE_USER_LAST : The constants above this number is for private styles of the IME ISV. IME ISV can define it own style freely. For ex. #define MSIME_NOUN (IME_REGWORD_STYLE_USER_FIRST) #define MSIME_VERB (IME_REGWORD_STYLE_USER_FIRST +1)
lpszString	LPCTSTR	The string to be registered.

Return value TRUE if the function is successful, FALSE if not.

1.4.5 ImmGetRegisterWordStyle

Syntax UINT WINAPI ImmGetRegisterWordStyle(*hKL, nItem, lpStyleBuf*)

Feature This API gets the available styles in the specified IME.

Parameter	Type	Description
hKL	HKL	Specifies the IME.
nItem	UINT	The maximum number of styles that the buffer can hold.
lpStyleBuf	LPSTYLEBUF	The buffer to be filled.

Return value Returns the number of the styles copied to the buffer or, if *nItems* is zero, returns the number of the styles that are available in the IME.

1.4.6 ImmEnumRegisterWord

Syntax UINT WINAPI ImmEnumRegisterWord(*hKL, lpfnEnumProc, lpReading, dwStyle, lpszString, lpData*)

Feature This API enumerates the information of register strings with specified reading string, style, and register string.

Parameter	Type	Description
hKL	HKL	Specifies the IME.
lpfnEnumProc	REGISTERWORDENUMPROC	Address of call back function.
lpszReading	LPCTSTR	Specifies the reading string to be enumerated. If *lpszReading* is NULL, this API enumerates all available reading strings that match with the specified *dwStyle* and *lpszString*.
dwStyle	DWORD	Specifies the style to be enumerate. If *dwStyle* is NULL, this API enumerates all available

		styles that match with the specified *lpszReading* and *lpszString*.
lpszString	LPCTSTR	Specifies the register string to be enumerate. If *lpszString* is NULL, this API enumerates all register strings that match with the specified *lpszReading* and *dwStyle*.
lpData	LPVOID	Address of application supplied data.

Return value: If this function is succeeds, the return value is the last value return by the callback function. Its meaning is defined by the application.

Notes: If all *lpszReading*, *dwStyle*, and *lpszString* are NULL, it will enumerates all register strings in the dictionary of IME.

1.4.7 EnumRegisterWordProc

Syntax: UINT CALLBAK EnumRegisterWordProc(*lpReading*, *dwStyle*, *lpszString*, *lpData*)

Feature: This function is an application defined callback function that process data of register string from **ImmEnumRegisterWord**.

Parameter	Type	Description
lpszReading	LPCTSTR	The matched reading string.
dwStyle	DWORD	The matched style.
lpszString	LPCTSTR	The matched register string.
lpData	LPVOID	Address of application supplied data.

Return value: The return value must be a nonzero value to continue enumeration; to stop enumeration, it must be zero.

2. Messages

2.1 WM_IME_SETCONTEXT

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

wParam: fSet = (BOOL)wParam;
fSet is TRUE when the Input Context gets active for the application. When it is FALSE, the Input Context gets inactive for the application..

lParam: lParam is the combination of following bits.
ISC_SHOWUICOMPOSITIONWINDOW

To show the composition window by UI window.

ISC_SHOWUIGUIDWINDOW

To show the guide window by UI window.

ISC_SHOWUISOFTKBD

To show the soft keyboard by UI window.

ISC_SHOWUICANDIDATEWINDOW

To show the candidate window of Index 0 by UI window.

(ISC_SHOWUICANDIDATEWINDOW << 1)

To show the candidate window of Index 1 by UI window.

(ISC_SHOWUICANDIDATEWINDOW << 2)

To show the candidate window of Index 2 by UI window.

(ISC_SHOWUICANDIDATEWINDOW << 3)

To show the candidate window of Index 3 by UI window.

Note — If the application draws the composition window by itself, UI window does not need to show its composition window. Then the application has to clear ISC_SHOWUICOMPOSITIONWINDOW bit of lParam and calls DefWindowProc() or ImmIsUIMessage() with it.

Return value — The return value of DefWindowProc() or ImmIsUIMessage.

2.2 WM_IME_CONTROL

This message is a group of sub messages to control IME User Interface. An application uses this message to interact with IME User Interface created by the application.

Followings are list of sub message classified by the value of wParam.

2.2.1 IMC_CLOSESTATUSWINDOW

Feature — An application sends this message to *IME window* to hide the status window of IME.

lParam — Not Used.

Return Value — Non-zero indicates failure otherwise 0.

Note — When the status window of IME is hidden, this message does nothing.
When the application send this message to IME window, the application won't receive the notification message (IMN_CLOSESTATUSWINDOW).

2.2.2 IMC_OPENSTATUSWINDOW

Feature — An application sends this message to *IME window* to show the status window of IME. When the system is not in a mode of 'Show IME Status', the status window will not be shown.

lParam — Not Used.

Return Value	Non-zero indicates failure otherwise 0.
Note	When the status window of IME is shown, this message does nothing. 'Show IME Status' can be changed from taskBar by the end user. The application can know the status window will be shown or hidden by receiving WM_IME_NOTIFY IMN_OPEN/CLOSESTATUSWINDOW. When the application send this message to IME window, the application won't receive the notification message (IMN_OPENSTATUSWINDOW).

2.2.3 IMC_GETCANDIDATEPOS

Feature	An application sends this message to *IME window* to get the position of the candidate window. IME can adjust the position of a candidate window regarding to screen boundary or other concerns. An application can get the real position of candidate window to decide whether it want to reposition it to another position.
lParam	lParam = (LPARAM) lpCANDIDATENFORM, buffer to retrieve the position of the candidate window.
Return Value	Non-zero indicates failure otherwise 0.
Note	The return value in the buffer is in the window coordinates of the focus window of an application.

2.2.4 IMC_GETCOMPOSITONWINDOW

Feature	An application sends this message to *IME window* to get the position of the composition window. An IME may adjust the position of a composition window regarding to some concerns. An application can get the real position of composition window to decide whether it want to reposition it to another position.
lParam	lParam = (LPARAM) lpCOMPOSITIONFORM, buffer to retrieve the position of the composition window.
Return Value	Non-zero indicates failure otherwise 0.
Note	This return value in the buffer is in the window coordinates of the focus window of an application.

2.2.5 IMC_GETSTATUSWINDOWPOS

Feature	An application sends this message to *IME window* to get the position of the status window.
lParam	Not Used.
Return Value	The value specifies a POINTS structure that contains the x- and y-coordinates of the position of the status window. The dimensions are given in screen coordinates, relative to the upper-left corner of the display screen.
Note	The POINTS structure has the following form:

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

2.2.6 IMC_SETCANDIDATEPOS

Feature	An application sends this message to *IME window* to specify recommended position to display a candidate window. This is particularly for such an application which displays composition characters without IME User Interface but uses IME User Interface to display candidates.
lParam	lParam = (LPARAM) lpCANDIDATEFORM, the application should specify the dwIndex field in lpCANDIDATEFORM.
Return Value	Non-zero indicates failure otherwise 0.

2.2.7 IMC_SETCOMPOSITONFONT

Feature	An application sends this message to *IME window* to specify a font to be used for displaying intermediate characters in the composition window.
lParam	lParam = (LPLOGFONT)lpLogFont
Return Value	Non-zero indicates failure otherwise 0.
Note	The *IME User Interface* changes current selected font in the Input Context when it processes this message.

2.2.8 IMC_SETCOMPOSITONWINDOW

Feature	An application sends this message to the *IME window* to specify style of composition window. In contrast to 3.1, this message specifies composition style to current active Input Context so once an application specify the style, *IME User Interface* will follow the style whenever it is given the Input Context.

lParam	lParam = (LPARAM) lpCOMPOSITIONFORM
	
	tagCOMPOSITIONFORM {
	DWORD dwStyle
	POINT ptCurrentPos;
	RECT rcArea;
	};
	See instructions of COMPOSITIONFORM structure for detail.

Return Value	Non-zero indicates failure otherwise 0.
Note	IME User Interface has a default style of composition window that is equal to CFS_POINT style. If an application hasn't specified any composition style into its Input Context, *IME User Interface* retrieves the current caret position and window client area when it opens composition window.

2.2.9 IMC_SETSTATUSWINDOWPOS

Feature	An application sends this message to *IME window* to set the position of the status window.

lParam — This specifies a POINTS structure that contains the x- and y-coordinates of the position of the status window. The dimensions are given in screen coordinates, relative to the upper-left corner of the display screen.

Return Value — Non-zero indicates failure otherwise 0.

Note — The POINTS structure has the following form:

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

2.3 WM_IME_COMPOSITION

Feature — This is sent 2 bytes of composition character to the application. When the application receive this message, IME updated the composition status as a result of user's key stroke. *IME User Interface* will change its appearance when it processes this message. An application can call **ImmGetCompositionString** to obtain new detail composition status**.**

wParam — Includes 2 bytes of DBCS character that is the latest change of composition character.
Windows NT-Unicode: Includes a unicode character.

lParam — *lParam* includes the combination of following flags. Basically the flag indicates which information was changed. An application check this to retrieve necessary information.

GCS_COMPSTR
GCS_COMPATTR
GCS_COMPCLAUSE
GCS_COMPREADSTR
GCS_COMPREADATTR
GCS_COMPREADCLAUSE
GCS_CURSORPOS
GCS_DELTASTART
GCS_RESULTSTR
GCS_RESULTCLAUSE
GCS_RESULTREADSTR
GCS_RESULTREADCLAUSE

The following value indicate special meaning as style bits for WM_IME_COMPOSITION.

CS_INSERTCHAR	An IME specifies this value when wParam shows a composition character which should be inserted into 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 doesn't want an application to move 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 calling ImmGetCompositionString() API to the current caret position but shouldn't move caret. Subsequent WM_IME_COMPOSITION with GCS_RESULTSTR will replace this character.

Return value Not used.

NOTE An application that wants to display composition characters by itself should not pass this message to either *Application IME window* or DefWindowProc(). DefWindowProc() processes this message to pass it to *Default IME window.*
IME will give this message to an application with NO GCS_ bit set when it just cancels the current composition. An application which draws its own composition string should delete the string when it receives the message.

See Also **ImeGetCompositionString**

2.4 WM_IME_COMPOSITIONFULL

Feature This message is sent to an application when IME User Interface finds no space to extend area for the composition window anymore. Application should specify the way to display window for IME User Interface in process of this messages.

wParam Not Used.
lParam Not Used.

Return Value Not Used.

Note This messages is sent to an application via SendMessage() by IME User Interface not by IME itself . This is a notification.

See Also IMC_SETCOMPOSITONWINDOW

2.5 WM_IME_ENDCOMPOSITION

Feature This message is sent to an application when IME close composition window.

wParam / lParam Not Used.

Return Value Not Used.

Note An application that wants to display composition characters by itself should not pass this message to either *Application IME window* window or DefWindowProc(). DefWindowProc() processes this message to pass it to *Default IME window.*

2.6 WM_IME_STARTCOMPOSITION

Feature This message is sent immediately before an IME generates composition string as a result of a user's key stroke. The *IME User Interface* will open its composition window when it receives this message.

wParam / lParam Not Used.

Return Value Not Used.

Note An application that wants to display composition characters by itself should not pass this message to either *Application IME window* window or DefWindowProc(). DefWindowProc() processes this message to pass it to *Default IME window.*

2.7 WM_IME_NOTIFY

This message is a group of sub messages to notify application or IME window of IME status.

Followings are list of sub message classified by the value of wParam. It is generated by the IME.

2.7.1 IMN_CLOSESTATUSWINDOW

Feature This message is sent when an IME is about to close a status window.

lParam Not Used.

Return Value Not Used.

Note The *IME User Interface* will close status window when it processes this message.

2.7.2 IMN_OPENSTATUSWINDOW

Feature This message is sent when an IME is about to create a status window. An application processes this message to display status window for the IME by itself.
Application can get information about status window with **ImmGetConversionStatus** API.

lParam NotUsed

Return Value Not Used

Note The *IME User Interface* will create a system window when it processes this message.

This message is sent by IMM when ImmSetActiveIMEContext() is called. IME sets strings to be displayed in system window into Input Context.

See Also **ImmGetConversionStatus**

2.7.3 IMN_OPENCANDIDATE

Feature This message is sent when an IME is about to open a candidates window. An application processes this message to call **ImmGetCandidateListCount** () / **ImmGetCandidateList()** to display candidates by its own way.

lParam Shows which candidate list should be updated.
Ex. if bit0 is 1, the first candidate list should be updated. If bit31 is 1, the 32nd candidate list should be updated.

Return Value Not Used.

Note The *IME User Interface* creates a candidate window when it processes this message.

See Also **ImeGetCandidateListCount** / **ImeGetCandidateList**, WM_IME_CHANGECANDIDATE

2.7.4 IMN_CHANGECANDIDATE

Feature This message is sent when an IME is about to change the content of candidates window. An application processes this message to display candidates by itself.

lParam Shows which candidate list should be updated.
Ex. if bit0 is 1, the first candidate list should be updated. If bit31 is 1, the 32nd candidate list should be updated.

Return Value Not Used.

Note The *IME User Interface* redraw a candidate window when it processes this message.

See Also **ImmGetCandidate** / **ImmGetCandidateListCount**

2.7.5 IMN_CLOSECANDIDATE

Feature This message is sent when an IME is about to close candidates window. An application processes this message to be informed about the end of candidate processing.

lParam Shows which candidate list should be closed.
Ex. if bit0 is 1, the first candidate list should be updated. If bit31 is 1, the 32nd candidate list should be updated.

Return Value Not Used

Note The UI window destroy a candidate window when it processes this message.

2.7.6 IMN_SETCONVERSIONMODE

Feature This message is sent when the conversion mode of the input context was updated. When the application or the IME window (it will pass to the UI window) receive this message, The application or the UI window can get information about system window with **ImmGetConversionStatus** API.

lParam Not Used.

Return Value Not Used

Note The UI window redraws its status window, if the status window shows the conversion mode.

2.7.7 IMN_SETSENTENCEMODE

Feature This message is sent when the sentence mode of the input context was updated. When the application or the IME window (it will pass to the UI window) receive this message, the application or the UI window can get information about system window with **ImmGetConversionStatus** API.

lParam Not Used.

Return Value Not Used

Note The UI window redraws its status window, if the status window shows the sentence mode.

2.7.8 IMN_SETOPENSTATUS

Feature This message is sent when the open status of the input context was updated. When the application or the IME window (it will pass to the UI window) receive this message, the application or IME's UI can get the information with **ImmGetOpenStatus** API.

lParam Not Used.

Return Value Not Used

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

2.7.9 IMN_SETCANDIDATEPOS

Feature This message is sent when an IME is about to move candidates windows. An application processes this message to be informed about the end of candidate processing.

lParam Shows which candidate list should be moved.

Ex. if bit0 is 1, the first candidate list should be updated. If bit31 is 1, the 32nd candidate list should be updated.

Return Value Not Used

Note The UI window moves a candidate window when it processes this message.

2.7.10 IMN_SETCOMPOSITIONFONT

Feature This message is sent when the logic Font of the Input Context was updated. When the application or the IME window (it will pass to the UI window) receive this message, the application or the UI window can get the information about composition font with **ImmGetCompositionFont** API.

lParam Not Used.

Return Value Not Used

Note The UI windows use lfFont to draw the text of the composition string.

2.7.11 IMN_SETCOMPOSITIONWINDOW

Feature This message is sent when the composition form of the input context was updated. When the IME window receive this message, the cfCompForm of the Input Context can be referred to get the new conversion mode.

lParam Not Used.

Return Value Not Used

Note The *IME User Interface* uses cfCompForm to draw the composition window.

2.7.12 IMN_SETSTATUSWINDOWPOS

Feature This message is sent when the ptStatusWndPos of the input context was updated. When the application or IME's UI receive this message, the ptStatusWndPos of the input context can be referred to get the new conversion mode.

lParam Not Used.

Return Value Not Used

Note *IME User Interface* windows use lfFont to draw the text of the composition string.

2.7.13 IMN_GUIDELINE

Feature	This message is sent when an IME is about to show the error or information. Application processes this message to call **ImmGetGuideLine**() to display the error or the information from IME.
lParam	Not Used. Have to be 0.
Return Value	Not Used.
Note	*IME User Interface* window may create a information window when it processes this message and show the information string.
See Also	**ImmGetGuideLine, GUIDELINE Structure**

2.8 WM_IME_KEYDOWN / WM_IME_KEYUP

Feature	This is sent to an application when IME need to generate WM_KEYDOWN / WM_KEYUP message. The form of value to be sent is same as original English Windows WM_KEYDOWN / WM_KEYUP.
wParam	This variable gets same as original English Windows WM_KEYDOWN / WM_KEYUP.
lParam	This variable gets same as original English Windows WM_KEYDOWN / WM_KEYUP.
Return value	Not used.
Note	Application can handle this message same as WM_KEYDOWN / WM_KEYUP message, or DefWindowProc() processes this message to generate WM_KEYDOWN / WM_KEYUP message with same wParam and lParam. This message is usually generated by IME to keep message order.

2.9 WM_IME_CHAR

Feature	This is sent to an application when IME get a character of the conversion result. The form of value to be sent is similar to original English Windows WM_CHAR. The difference is that wParam can include 2 byte of character.
wParam	Includes 2 bytes for a FE character . Windows NT-Unicode: Includes a unicode character
lParam	This variable gets similar as original English Windows WM_CHAR.

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

Return value	Not used.
Note	DefWindowProc() processes this message to generate 2 of WM_CHAR messages each of that includes 1 byte of DBCS character in the case if it includes 2byte of FE character. If the message just includes a SBCS character, DefWindowProc() generates 1 WM_CHAR.

2.10 VK_PROCESSKEY

Feature	This is sent to an application as a wParam of WM_KEYDOWN or WM_KEYUP. When this virtual key is generated, the real virtual key is saved in the System or the messages that was generated by IME are stored in the Input Context. When the applications find this VKey by calling GetMessage() or PeekMessage(), the applications should call TranslateMessage(). If the application want to know the real VKey, it can get by calling ImmGetVirtualKey().
lParam	must be 1.