OLE Automation

SimpleEditorMode 3
IsEnabled 3
ReadFile 3
SaveAs 3
SaveAsFile 3
Print 3
PrintSetup 3
Undo 4
UndoHistory 4
Copy 4
Cut 4
Paste 4
PasteSpecial 4
InsertPicture 4
Character 5
Paragraph 5
StyleUse 5
StyleDefine 5
Brush 5
GoToBookmark 5
GoToBlock 6
GoToLine 6
InsertText 6
AppMaximize 6
AppMinimize 6
FileClose 6
FileExit 7
EditReplace 7
FilePageSetup 7
FilePageMargins 8
DocumentProtection 8
ToolsProtectDocument 8
ToolsUnProtectDocument 8
ToolsProtectSection 8
InsertNamedBlock 8
ReplaceNamedBlockText 9
OpenDocument 9
SetCaretChangeMsg 9
GetBold 9
SetBold 9
GetItalic 9

SetItalic 10
GetUnderlineType 10
SetUnderlineType 10
GetTextColor 10
SetTextColor 11
GetParagraphColor 11
SetParagraphColor 11
GetAlign 11
SetAlign 11
InsertPictureFromFile 12
RemoveObject 12
IsObjectMode 13
GoToTextMode 13
SetScaleType 13
SetScale 13
GetScaleType 13
GetScale 14
SaveAsFileEx 14
GrowFont 14
ShrinkFont 14
GetFontFace 14
GetFontCharSet 15
SetFontFace 15
GetFontSize 15
SetFontSize 15
IsDocumentDirty 15
SetDocumentDirty 15
Find 15
RemoveText 16
GetText 16
SavePosition 16
RestorePosition 16
MovePosition 16
Příklady aktivace funkcí OLE Automation v 602Text 17
Vytvořeni instance objektu 602Text 17
Volání funkce OLE automation přes IDispatch Interface 17
Volání funkce Print 17
Volání funkce ReadFile 17
Uvolnění objektu 18

Funkce OLE Automation lze volat přes *IDispatch Interface*.

SimpleEditorMode

```
void SimpleEditorMode(void);
```

Popis: **602Text** v in-place aktivaci bude bez nástrojových lišt a nabídky menu.
Od verze 2000 (3.3.2000 version "0E10") funkce navíc vypne i pravítka.

IsEnabled

```
int IsEnabled(int dispid);
```

Popis: Test zjištění, zda je funkci (zadané číslem: **DISPIDs**) v daném okamžiku možno volat.

ReadFile

```
int ReadFile(LPOLESTR pFileName, int iReadOnly);
```

Popis: Funkce načte soubor. Při úspěšném provedení vrací NOERROR.

SaveAs

```
(DLG) void SaveAs(void)
```

Popis: Funkce pomocí dialogu **Uložit jako** uloží aktuální dokument.

SaveAsFile

```
int SaveAsFile(LPOLESTR pFileName)
```

Popis: Funkce uloží aktuální dokument pod jménem **pFileName** (ukládaný formát se odvozuje z přípony v **pFileName**). Při úspěšném provedení vrací NOERROR.

Viz: `SaveAsFileEx`

Print

```
void Print(void);
```

Popis: Funkce vytiskne aktuální dokument.

PrintSetup

```
(DLG) void PrintSetup(LPOLESTR pPrintName);
```

Popis: Funkce zobrazí dialog **Tisk** k nastavení tiskárny.

Undo

```
void Undo(void);
```

Popis: Funkce zruší poslední provedené změny. Je adekvátní příkazu **Odvolat** z menu **Úpravy**.

UndoHistory

```
(DLG) void UndoHistory(void);
```

Popis: Funkce pomocí dialogu **Odvolat** odvolá vybrané změny. Je adekvátní příkazu **Odvolat ...** z menu **Úpravy**.

Copy

```
void Copy(void);
```

Popis: Funkce zkopíruje označený blok textu do schránky. Je adekvátní příkazu **Kopírovat** z menu **Úpravy**.

Cut

```
void Cut(void);
```

Popis: Funkce z dokumentu vyjme označený blok textu a uloží jej do schránky. Je adekvátní příkazu **Vyjmout** z menu **Úpravy**.

Paste

```
void Paste(void);
```

Popis: Funkce vloží blok textu ze schránky na místo určené kurzorem. Je adekvátní příkazu **Vložit** z menu **Úpravy**.

PasteSpecial

```
(DLG) void PasteSpecial(void);
```

Popis: Funkce pomocí dialogu **Vložit jinak** vloží blok textu ze schránky na místo určené kurzorem. Je adekvátní příkazu **Vložit jinak** z menu **Úpravy**.

InsertPicture

```
(DLG) void InsertPicture(void);
```

Popis: Funkce pomocí dialogu **Otevřít obrázek** vloží vybraný obrázek do aktuálního dokumentu. Je adekvátní příkazu **Obrázek** z menu **Vložit**.

Character

```
(DLG) void Character(void);
```

Popis: Funkce pomocí dialogu **Pismo** upraví písmo v dokumentu dle zadaných parametrů. Je adekvátní příkazu **Písmo** z menu **Formát**.

Paragraph

```
(DLG) void Paragraph(void);
```

Popis: Funkce pomocí dialogu **Odstavec** upraví zarovnání, řádkování a odsazení dle zadaných parametrů. Je adekvátní příkazu **Odstavec** z menu **Formát**.

StyleUse

```
void StyleUse(void);
```

Popis: Funkce pomocí dialogu **Styl odstavce** umožní přiřadit odstavci nebo skupině odstavců zvolený styl.

StyleDefine

```
(DLG) void StyleDefine(void);
```

Popis: Funkce pomocí dialogu **Definice stylu odstavce** nastaví zvolené parametry. Je adekvátní příkazu **Definice stylu odstavce** z menu **Formát**.

Brush

```
void Brush(void);
```

Popis: Funkce aktivuje štěteček.

GoToBookmark

```
int GoToBookmark(LPOLESTR pBookmarkName);
```

Popis: Funkce umožní odskok na záložku. Parametr **`pBookmarkName`** je jméno záložky. Při úspěšném provedení vrací NOERROR.

GoToBlock

```
int GoToBlock(LPOLESTR pBlockName);
```

Popis: Funkce umožňuje odskok na pojmenovaný blok, který zároveň označí. Parametr **`pBlockName`** je jméno bloku. Při úspěšném provedení vrací NOERROR.

GoToLine

```
int GoToLine(int iLine, int iColumn);
```

Popis: Funkce umožní odskok na řádek daný parametrem **`iLine`** a sloupec daný parametrem **`iColumn`**. Při úspěšném provedení vrací 1.

InsertText

```
int InsertText(LPOLESTR);
```

Popis: Funkce vloží textový řetězec na aktuální pozici textového kurzoru. Při úspěšném provedení vrací NOERROR.

AppMaximize

```
void AppMaximize(void);
```

Popis: Funkce maximalizuje aplikaci 602Text.

AppMinimize

```
void AppMinimize(void);
```

Popis: Funkce minimalizuje aplikaci 602Text.

FileClose

```
void FileClose(AT_SAVEPARAM iSave);
```

Popis: Funkce uzavře dokument otevřený funkcí **ReadFile()**.

Hodnoty parametru **`iSave`**:

```
typedef enum {
AT_NORMAL       = 0    zobrazí se dotaz, zda uložit změněný dokument
AT_SAVE         = 1    změněný dokument se automaticky uloží
AT_CLOSE        = 2    dokument se uzavře bez uložení
} AT_SAVEPARAM
```

FileExit

```
void FileExit(AT_SAVEPARAM iSave);
```

Popis: Funkce zavře aplikaci.

Hodnoty parametru **iSave**:

```
typedef enum {
AT_NORMAL        = 0    zobrazí se dotaz, zda uložit změněné dokumenty
AT_SAVE          = 1    změněné dokumentz se automaticky uloží
AT_CLOSE         = 2    dokumenty se uzavře bez uložení
} AT_SAVEPARAM
```

EditReplace

```
void EditReplace(LPOLESTR pFind, LPOLESTR pReplace, AT_FINDREPLACEPARAM
  iParam);
```

Popis: Funkce nahradí text daný parametrem **pFind** textem daným parametrem **pReplace**.

Hodnoty parametru **iParam**:

```
typedef enum {
AT_FIND_UP     = 0x00001      směr nahoru
AT_MATCH_CASE  = 0x00002      rozlišovat malá a velká písmena
AT_WHOLE_WORD  = 0x00004      pouze celá slova
AT_REPLACE_ALL = 0x00008      nahradit vše
} AT_FINDREPLACEPARAM;
```

FilePageSetup

```
void FilePageSetup(AT_PAGESIZETYPE iSizetype, int iOrientation, double
  iWidth, double iHeight);
```

Popis: Funkce nastaví vlastnosti stránky:

iSizetype - velikost stránky (viz. **AT_PAGESIZETYPE**)

```
typedef enum {
AT_A4              = 0
AT_LETER           = 1
AT_LEGAL           = 2
AT_USERDEFINED     = 3
} AT_PAGESIZETYPE;
```

iOrientation - 0 = portrait; 1 = landscape
iWidth, iHeight mají význam pro iSizetype´= - AT_USERDEFINED a určují šířku a výšku stránky

FilePageMargins

```
void FilePageMargins(double iLeft, double iTop, double iRight, double
   iBottom;
```

Popis: Funkce nastaví okraje stránky.

DocumentProtection

```
int DocumentProtection(void);
```

Popis: Funkce vrací 1 je-li dokument uzamčen; v opačném případě (není-li dokument uzamčen) vrací 0.

ToolsProtectDocument

```
int ToolsProtectDocument(LPOLESTR pPassword);
```

Popis: Funkce uzamkne dokument. Při úspěšném provedení vrací 1.

ToolsUnProtectDocument

```
int ToolsUnProtectDocument(LPOLESTR pPassword);
```

Popis: Funkce odemkne dokument. Při úspěšném provedení vrací 1.

ToolsProtectSection

```
int ToolsProtectSection(int iSection, int iProtect)
```

Popis: Funkce určí, zda bude sekce po zavolání funkce **ToolsProtectDocument()** zamčena nebo odemčena pro editaci.

`iSection`	číslo sekce (první má číslo 1)
`iProtect = 0`	sekce bude odemčena
`iProtect = 1`	sekce bude zamčena

Při úspěšném provedení vrací 1.

InsertNamedBlock

```
int InsertNamedBlock(LPOLESTR pBlockName, LPOLESTR pText);
```

Popis: Funkce vloží pojmenovaný blok na aktuální pozici textového kurzoru. Při úspěšném provedení vrátí nenulovou hodnotu.

`pBlockName`	jméno pojmenovaného bloku
`pText`	text, který bude vložen do pojmenovaného bloku

ReplaceNamedBlockText

```
int ReplaceNamedBlockText(LPOLESTR pBlockName, LPOLESTR pText);
```

Popis: Funkce nahradí obsah pojmenovaného bloku textem. Při úspěšném provedení vrátí nenulovou hodnotu.

`pBlockName`	jméno pojmenovaného bloku
`pText`	text, který bude vložen do pojmenovaného bloku

OpenDocument

```
void OpenDocument();
```

Popis: Funkce otevře dokument v 602Text.

SetCaretChangeMsg

```
void SetCaretChangeMsg(UINT hWindow, UINT uMessage, UINT Immediately);
```

Popis: Funkce se používá v případě, kdy chce aplikace sledovat změny pozice textového kurzoru.

`uMessage`		číslo zprávy, kterou editor pošle při změně pozice textového kurzoru oknu **`hWindow`**
`fImmediately`	= TRUE	zprávu pošle hned po přesunu textového kurzoru
	= FALSE	zprávu pošle s prodlevou (až je-li čas - obdobně, jako se provádí aktualizace nástrojové lišty v 602Text)

GetBold

```
int GetBold(void);
```

Popis: Funkce slouží ke zjištění vlastností písma na aktuální pozici textového kurzoru nebo v označeném bloku.

Vrací 1, je-li nastaveno tučné písmo, 0 není-li nastaveno tučné písmo.

SetBold

```
void SetBold(int iParam);
```

Popis: Funkce slouží k nastavení tučného písma na aktuální pozici textového kurzoru nebo v označeném bloku.

Hodnoty parametru **`Param`**:

`1`	nastaví tučné písmo
0	vypne tučné písmo

GetItalic

```
int GetItalic(void);
```

Popis: Funkce slouží ke zjištění vlastností písma na aktuální pozici textového kurzoru nebo v označeném bloku.

Vrací: 1 je-li nastavena kurzíva
0 není-li nastavena kurzíva

SetItalic

```
void SetItalic(int iParam);
```

Popis: Funkce slouží k nastavení kurzívy na aktuální pozici textového kurzoru nebo v označeném bloku.

Hodnoty parametru **`Param`**: 1 nastaví kurzívu
0 vypne kurzívu

GetUnderlineType

```
int GetUnderlineType(void);
```

Popis: Funkce slouží ke zjištění vlastností písma na aktuální pozici textového kurzoru nebo v označeném bloku.

Vrací **AT_UDERLINETYPE** nebo **-1**, je-li potržení neurčité.

```
typedef enum {
```

AT_UDERLINE_NONE	text není podtržen
AT_UDERLINE_ALL	podtržení včetně mezer
AT_UDERLINE_WORD	podtržení slov
AT_UDERLINE_DOUBLE	dvojité podtržení

```
} AT_UDERLINETYPE;
```

SetUnderlineType

```
void SetUnderlineType(int iType);
```

Popis: Funkce nastavuje způsob pdtržení textu.

```
typedef enum {
```

AT_UDERLINE_NONE	text není podtržen
AT_UDERLINE_ALL	podtržení včetně mezer
AT_UDERLINE_WORD	podtržení slov
AT_UDERLINE_DOUBLE	dvojité podtržení

```
} AT_UDERLINETYPE;
```

GetTextColor

```
int GetTextColor(void);
```

Popis: Funkce slouží ke zjištění barvy textu na aktuální pozici řádkového kurzoru, event. označeného bloku textu - pak vrací **COLORREF**. Pokud je barva neurčitá (je označen vícebarevný blok textu) funkce vrací **-1**.

SetTextColor

```
void SetTextColor(COLORREF iTextColor);
```

Popis: Funkce zajišťuje nastavení barvy písma.

GetParagraphColor

```
int GetParagraphColor(void);
```

Popis: Funkce slouží ke zjištění barvy pozadí odstavce ve kterém je umístěn řádkový kurzor - pak vrací **COLORREF**. Pokud je barva neurčitá (je označen vícebarevný blok textu) funkce vrací **-1**.

SetParagraphColor

```
void SetParagraphColor(COLORREF iParaColor);
```

Popis: Funkce slouží ke nastavení barvy pozadí odstavce ve kterém je umístěn řádkový kurzor.

GetAlign

```
int GetAlign(void);
```

Popis: Funkce vrací v textovém módu zarovnání odstavce nebo pozici objektu je-li označen objekt. Návratová hodnota je **AT_XALIGNTYPE** nebo je-li zarovnání neurčité **-1**.

```
typedef enum
      {
         AT_XALIGN_LEFT        zarovnání vlevo
         AT_XALIGN_CENTER      zarovnání na střed
         AT_XALIGN_RIGHT       zarovnání na vpravo
         AT_XALIGN_FLUSH       zarovnání do bloku
      } AT_XALIGNTYPE;
```

SetAlign

```
void SetAlign(int iAlignType);
```

Popis: Funkce nastaví zarovnání (vpravo, vlevo, na střed, odoustranné).

Hodnoty parametru **iAlignType** viz funkce **GetAlign**.

InsertPictureFromFile

```
int InsertPictureFromFile(LPOLESTR pPictureName, int iInitAttribute);
```

Popis: Funkce vrací číslo vytvořeného objektu:

`iInitAttribute` = -1		vlastnosti dle aktuálního nastavení pro vlasnosti nových objektů
`iInitAttribute` ! = -1	`LOWORD`	umístění viz **AT_OBJECT_ATTRIBUTES_L**
	`HIWORD`	obtékání viz **AT_OBJECT_ATTRIBUTES_H**

Například:

`iInitAttribute` = 0x00010000	pevný a obtékaný okolo objektu

```
typedef enum {
      AT_FIXED                    = 0x00000  pevný
      AT_FIXEDONPAGE              = 0x00001  pevný na stránce
      AT_FLOATINGWITHPARA         = 0x00002  plovoucí se odstavcem
      AT_FLOATINGWITHCOLUMN       = 0x00004  plovoucí se sloupcem
      AT_FLOATINGWITHCHARACTER    = 0x00008  plovoucí s písmenem
      AT_THROUGHCHAPTER           = 0x00010  opakovací v celé kapitole
      AT_THROUGHDOCUMENT          = 0x00020  opakovací v dokumentu
pro opakovací:
      AT_ON_EVERY_PAGE            = 0x00100  opakovací na každé stránce
      AT_ODD_PAGE_ONLY            = 0x00200  opakovací na lichých stránkách
      AT_EVEN_PAGE_ONLY           = 0x00400  opakovací na sudých stránkách
      AT_EXCUDE_FIRST_PAGE        = 0x00800  není na první stránce
pro všechny:
      AT_LOCKED                   = 0x10000  uzamknou velikost a polohu
      } AT_OBJECT_ATTRIBUTES_L;

      typedef enum {
        AT_WRAPNONE                 = 0x00000
        AT_WRAPALL                  = 0x00001
        AT_WRAPTOPBOTTOM            = 0x00002
      } AT_OBJECT_ATTRIBUTES_H;
```

RemoveObject

```
int RemoveObject(int iObjectID);
```

Parametry: `iObjectID` číslo objektu

Popis: Funkce slouží k odstranění objektu. Při úspěšném provedení vrací 1.

IsObjectMode

```
int IsObjectMode(void);
```

Popis: Funkce testuje je-li vybrán objekt resp. více objektů.

Vrací: 1 objekt je vybrán

0 objekt není vybrán

GoToTextMode

```
void GoToTextMode(void);
```

Popis: Editor se přepne do režimu vkládání textu.

SetScaleType

```
int SetScaleType(int iScaleType)
```

Popis: Funkce nastaví zvětšení dokumentu dle **`iScaleType`** (typu **AT_SCALETYPE**).

Při úspěšném provedení vrací 1.

```
typedef enum
      {
         AT_MARGINS_WIDTH      zvětšení dokumentu mezi okraji
         AT_PAGE_WIDTH         šířka stránky
         AT_PAGE               celá stránka
         AT_PERCENTCOUNT       dáno procentem - nastavení pomocí funkce
```

SetScale()

```
      } AT_SCALETYPE;
```

SetScale

```
int SetScale(int iPercentCount);
```

Parametry: `iPercentCount` počet procent

Popis: Funkce provede zvětšení dokumentu dle zadaného počtu procent. Při úspěšném provedení vrací 1.

GetScaleType

```
int GetScaleType();
```

Popis: Funkce vrací **AT_SCALETYPE**.

Viz: **SetScaleType**

GetScale

```
int GetScale();
```

Popis: Funkce vrací aktuální zvětšení dokumentu v procentech.

SaveAsFileEx

```
int SaveAsFileEx(LPOLESTR pFileName, int iFormat);
```

Popis: Funkce uloží aktivní dokument pod jmenem **`pFileName`**; **`iFormat`** určuje ukládání (typu **AT_FILE_FORMAT**). Při úspěšném provedení vrací NOERROR.

```
typedef enum
      {
        AT_602TextDokument
        AT_602TextTemplate
        AT_WinText3_95
        AT_Text602
        AT_MSWord95
        AT_MSWord2000
        AT_HTML
        AT_XHTML
        AT_Ascii
      } AT_FILE_FORMAT
```

GrowFont

```
void GrowFont();
```

Popis: Funkce provede zvětšení písma. Je adekvátní tlačítku **Zvětšit** z nástrové lišty **Formát**.

ShrinkFont

```
void ShrinkFont();
```

Popis: Funkce provede zmenšení písma. Je adekvátní tlačítku **Zmešit** z nástrové lišty **Formát**.

GetFontFace

```
LPOLESTR GetFontFace(void);
```

Popis: Funkce vrátí název fontu (v místě textového kurzoru).

GetFontCharSet

```
int GetFontCharSet(void);
```

Popis: Funkce vrátí znakovou sadu fontu (CharSet) v místě textového kurzoru. Při neúspěchu vrátí **-1**.

SetFontFace

```
int SetFontFace(LPOLESTR pFontName, int iCharSet);
```

Popis: Funkce nastaví font dle **`pFontName`** + **`iCharSet`**. Při úspěšném provedení vrací nenulovou hodnotu.

GetFontSize

```
int GetFontSize(void);
```

Popis: Funkce vrátí velikost fontu nebo **-1**.

SetFontSize

```
int SetFontSize(int iFontSize);
```

Popis: Funkce nastaví velikost fontu. Při úspěšném provedení vrací nenulovou hodnotu.

IsDocumentDirty

```
int IsDocumentDirty(void);
```

Popis: Funkce vrátí: 1 dokument byl změněn
0 dokument nezměněn

SetDocumentDirty

```
void SetDocumentDirty(int iDirty);
```

Popis: Funkce nastaví **`iDirty`**: 0 dokument nezměněn
1 dokument změněn

Find

```
int Find(LPOLESTR pFind, AT_FINDREPLACEPARAM  iParam);
```

Popis: `pFind` hledaný string (označí se)
`iParam` viz funkce **EditReplace**

Při úspěšném provedení vrací **1**.

RemoveText

```
int RemoveText()
```

Popis: Funkce odstraní označený text nebo znak na pozici kurzoru. Při úspěšném provedení vrací **1**.

GetText

```
LPOLESTR GetText();
```

Popis: Funkce vrátí text obsažený v označeném bloku.

SavePosition

```
int SavePosition();
```

Popis: Zapamatuje si aktuální pozici kurzoru. Při úspěšném provedení vrací **1**.

RestorePosition

```
int RestorePosition();
```

Popis: Návrat na pozici, jaká byla ve chvíli volání **`SavePosition()`**. Při úspěšném provedení vrací **1**.

MovePosition

```
int MovePosition(int iBack, int iShiftCtrl);
```

Popis: Funkce změní pozici kursoru:

`iBack` = 0 dopředu

`iBack` = 1 dozadu

`LOWORD(iShiftCtrl)`	= 0	pouhý přesun pozice kurzoru (bez označení textu)
	= 1	zároveň se provádí označení textu (obdobně jako při přesunu pozice kurzoru se současným stisknutím klávesy **Shift**)
`HIWORD(iShiftCtrl)`	= 0	přesun pozice kurzoru po písmenech
	= 1	přesun pozice kurzoru po slovech (obdobně jako při přesunu pozice kurzoru se současným stisknutím klávesy **Ctrl**)

Při úspěšném provedení vrací **1**.

Příklady aktivace funkcí OLE Automation v 602Text

Vytvořeni instance objektu 602Text

```
DEFINE_GUID(602TEXT_CLSID,0x45068E61L,0x1257,0x101B,0x89,0x7A,0x04,0x02,
0x1C, 0x00, 0x70, 0x02);

LPDISPATCH   pDispatch;

CoCreateInstance(WINTEXT_CLSID, NULL, CLSCTX_LOCAL_SERVER |
CLSCTX_INPROC_SERVER, IID_IDispatch, (LPVOID *)&pDispatch);
```

Volání funkce OLE automation přes IDispatch Interface

```
if  (pDispatch) {
       DISPID      rgDispId;
       HRESULT      hrErr;
       LPOLESTR    pOleNameů
       VARIANT     varResult;
```

Volání funkce Print

```
OLECHAR     szPrint[] = OLESTR("Print");
DISPPARAMS   dispparamsNoArgs = {NULL, NULL, 0, 0};

pOleName = szPrint;
hrErr = pDispatch->GetIDsOfNames(IID_NULL, &pOleName, 1,
LOCALE_USER_DEFAULT, &rgDispId);
pDispatch->Invoke(rgDispId, GUID_NULL, LOCALE_SYSTEM_DEFAULT,
DISPATCH_METHOD,
&dispparamsNoArgs, &varResult, NULL, NULL);
```

Volání funkce ReadFile

```
VARIANT    varg2[2];
DISPPARAMS dispparams2 = {varg2,0,2,0};
OLECHAR    szFileName[] = OLESTR("c:\\navod.txt");

VariantInit(&varResult);
VariantInit(&varg2[0]);
VariantInit(&varg2[1]);

varResult.vt = VT_I4;

varg2[1].vt     = VT_BSTR;
varg2[1].bstrVal = SysAllocString(szFileName);

varg2[0].vt     = VT_I4;
varg2[0].intVal = 0;

pDispatch->Invoke(rgDispId, GUID_NULL, LOCALE_USER_DEFAULT, DISPATCH_METHOD,
       &dispparams2, &varResult, NULL, NULL);

VariantClear(&varg2[0]);
VariantClear(&varg2[1]);
```

Uvolnění objektu

```
    pDispatch->Release();
        }
```