About Polar SpellChecker

Let us introduce Polar SpellChecker Component v2.0!

Polar SpellChecker Component includes ActiveX and Dynamic-link library component for adding **spell checking** functionality, and ActiveX and Dynamic-link library component for adding **auto correct** functionality to a higher-level application. SpellChecker and AutoCorrect components contain all functions necessary for usual spell checking and auto correct:

- get suggestions for words that are not in dictionary
- change, ignore or add words in custom dictionary
- correct two initial caps
- capitalization of the first letter in the sentence
- user defined exceptions for capitalizing of the first letter in the sentence
- correction of accidental use of CAPS LOCK key
- user definable words for replacing while typing, and others.

It also includes **SpellMaker - Wizard** for the creation of your own dictionaries. From the simple txt file written by the rule "one word" - "one line", SpellMaker will generate a .lex file which can be used by Polar SpellChecker.

Or you can simply download one of dictionaries offered on our site:
www.polarsoftware.com/download.html

- Czech NEW!
- Danish
- Dutch
- French
- German
- Italian
- Latin
- Polish NEW!
- Slovenian NEW!
- Spanish
- Swedish
- UK English NEW!
- US English

(You can switch between dictionaries at runtime.), or let your end-users to download them themselves, from the inside of your application, with just one line of code!

This help will introduce all functions that **Polar SpellChecker** DLL can perform and all properties and methods contained in **Polar SpellChecker** OCX.

Technical Support

If you need help using **Polar SpellChecker**, contact Polar in any of the following ways:

- visit our home page at www.polarsoftware.com.
- E-mail: support@polarsoftware.com or info@polarsoftware.com.

Using Polar SpellChecker with Microsoft Visual Basic 5.0

Adding POLAR SpellChecker control to Toolbox:

From "Project" menu select "Components…" or press Ctrl + T

In "Components" dialog find "POLAR SpellChecker ActiveX Control module" in the list

Select it and click checkbox next to it

Click "Apply" and then "OK" to close dialog

Now there will be new icon for POLAR SpellChecker in the toolbox

Using POLAR SpellChecker control

1. Create new standard project
2. Insert POLAR SpellChecker component into the form
3. Create five buttons and change their caption to "Open Dictionary", "About", "IsWordExist", "GetSuggestion", "AddWord"
4. Create one text box
5. Create one list box
6. Create one label
7. For button "Open Dictionary" put this code:

```
SpellChecker1.OpenDictionary "c:\MyApp\english.lex", "c:
\MyApp\custom.dic"
```

(for lex file put any dictionary that You have)

8. For button "About" put this code:

```
SpellChecker1.AboutBox
```

9. For button "IsWordExist" put this code:

```
If (SpellChecker1.IsWordExist(Text1.Text)) Then
    Label1.Caption = "Exists"
Else
    Label1.Caption = "Doesn't exist"
End If
```

10. For button "GetSuggestion" put this code:

```
Dim Str As String
Dim count As Integer
List1.Clear
count = 0
Do
```

```
        Str = SpellChecker1.GetSuggestion(Text1.Text, count)
        count = count + 1
        If Str <> "" Then
            List1.AddItem Str
        End If
    Loop Until Str = ""
```

11. For button "AddWord" put this code:

```
SpellChecker1.AddWord Text1.Text
```

12. Run application
13. Click "Open Dictionary"
14. Click "About"
15. Write a word in text box
16. Click "IsWordExist"

    If word exist, label will be "Exists"

    If word doesn't exist, label will be "Doesn’t Exist"

17. Click "GetSuggestion"

    There will be list of suggestions for that word in list-box

18. Write non existing word in textbox
19. Click "IsWordExist"

    label will be "Doesn’t Exist"

20. Click "AddWord"
21. Click "IsWordExist"

    label will be "Exists"

Using Polar SpellChecker with Microsoft Visual C 5.0

Adding POLAR SpellChecker control:

From "Project" menu select "Add To Project" / "Components and Controls"

In "Registered ActiveX Controls" directory select "POLAR SpellChecker Control"

Click button "Insert"

Click "OK" on next dialogs

Now there will be new icon for POLAR SpellChecker in the "Controls" palette

Using POLAR SpellChecker control

1. Create new MFC AppWizard project (Dialog Based)
2. Insert POLAR SpellChecker component into the form
3. Create five buttons and change their caption to "Open Dictionary", "About", "IsWordExist", "GetSuggestion", "AddWord"
4. Create one edit box
5. Create one list box
6. Create one static and change its ID to IDC_STATIC_TEXT
7. Add member variable (Category: control) for SpellChecker control and name it m_SpellChecker
8. Add member variable (Category: Value, Variable Type: CString) for edit box and name it m_edit
9. Add member variable (Category: Value, Variable Type: CString) for static and name it m_static
10. Add member variable (Category: control) for list-box control and name it m_listbox
11. For button "Open Dictionary" put this code:

```
m_SpellChecker.OpenDictionary("c:\MyApp\english.lex", "c:
\MyApp\custom.dic");
```

(for lex file put any dictionary that You have)

12. For button "About" put this code:

```
m_SpellChecker.AboutBox();
```

13. For button "IsWordExist" put this code:

```
UpdateData(TRUE);
if( m_SpellChecker.IsWordExist(m_edit) )
m_static = "Exists";
else
m_static = "Doesn't Exist";
UpdateData(FALSE);
```

14. For button "GetSuggestion" put this code:

```
UpdateData(TRUE);
CString strSuggestion;
INT nCount = 0;
m_listbox.ResetContent();
do{
  strSuggestion = m_SpellChecker.GetSuggestion(m_edit, nCount);
  nCount++;
  if (strSuggestion != "")
      m_listbox.AddString(strSuggestion);
}while( !(strSuggestion == "") );
```

15. For button "AddWord" put this code:

```
UpdateData(TRUE);
m_SpellChecker.AddWord(m_edit);
```

16. Run application
17. Click "Open Dictionary"
18. Click "About"
19. Write a word in edit box
20. Click "IsWordExist"

    If word exist, static will be "Exists"

    If word doesn't exist, static will be "Doesn't Exist"

21. Click "GetSuggestion"

    There will be list of suggestions for that word in list-box

22. Write non existing word in edit box
23. Click "IsWordExist"

    static will be "Doesn't Exist"

24. Click "AddWord"
25. Click "IsWordExist"

    static will be "Exists"

Using Polar SpellChecker with Borland C++ Builder 1.0

Adding POLAR SpellChecker control to tool palette:

From "Component" menu select "Install…"

In "Install Components" dialog click "ActiveX" button

Find "POLAR SpellChecker ActiveX Control module" in the list and select it

Click "OK" on both dialogs

Now there will be new icon for POLAR SpellChecker in the palette (Active X)

Using POLAR SpellChecker control

1. Create new project
2. Insert POLAR SpellChecker component into the form
3. Create five buttons and change their caption to "Open Dictionary", "About", "IsWordExist", "GetSuggestion", "AddWord"
4. Create one edit box
5. Create one list box
6. Create one label
7. For button "Open Dictionary" put this code:

```
SpellChecker1->Delphi := TRUE;
SpellChecker1->OpenDictionary("c:\\s\\US.lex", "c:\\s\\custom.dic");
```

(for lex file put any dictionary that You have)

8. For button "About" put this code:

```
SpellChecker1->AboutBox();
```

9. For button "IsWordExist" put this code:

```
if( SpellChecker1->IsWordExist(Edit1->Text) )
Label1->Caption = "Exists";
else
Label1->Caption = "Doesn't Exist";
```

10. For button "GetSuggestion" put this code:

```
String strSuggestion;
short nCount = 0;
ListBox1->Clear();
do{
  strSuggestion = SpellChecker1->GetSuggestion(Edit1->Text, nCount);
```

```
    nCount++;
    if (strSuggestion != "")
        ListBox1->Items->Add(strSuggestion);
}while( !(strSuggestion == "") );
```

11. For button "AddWord" put this code:

```
SpellChecker1->AddWord(Edit1->Text);
```

12. Run application
13. Click "Open Dictionary"
14. Click "About"
15. Write a word in edit box
16. Click "IsWordExist"

    If word exist, label will be "Exists"

    If word doesn't exist, label will be "Doesn’t Exist"

17. Click "GetSuggestion"

    There will be list of suggestions for that word in list-box

18. Write non existing word in edit box
19. Click "IsWordExist"

    label will be "Doesn’t Exist"

20. Click "AddWord"
21. Click "IsWordExist"

    label will be "Exists"

Using Polar SpellChecker with Borland Delphi 3.0

Adding POLAR SpellChecker control to tool palette:

From "Component" menu select "Import ActiveX Control…"

Find "POLAR SpellChecker ActiveX Control module" in the list and select it

Click "Install"

Click "OK" in "Install" dialog

Click "OK" for rebuilding

Close "Package" dialog and save changes

Now there will be new icon for POLAR SpellChecker in the palette (Active X)

Using POLAR SpellChecker control

1. Create new project
2. Insert POLAR SpellChecker component into the form
3. Create five buttons and change their caption to "Open Dictionary", "About", "IsWordExist", "GetSuggestion", "AddWord"
4. Create one edit box
5. Create one list box
6. Create one label
7. For button "Open Dictionary" put this code:

```
SpellChecker1.Delphi := TRUE;
SpellChecker1.OpenDictionary('c:\s\US.lex','c:\s\custom.dic');
```

(for lex file put any dictionary that You have)

8. For button "About" put this code:

```
SpellChecker1.AboutBox;
```

9. For button "IsWordExist" put this code:

```
if( SpellChecker1.IsWordExist(Edit1.Text) ) then
Label1.Caption := 'Exists'
else
Label1.Caption := 'Doesn't Exist';
```

10. For button "GetSuggestion" put this code:

```
var strSuggestion:widestring;
    nCount: smallint;
```

```
begin
nCount := 0;
ListBox1.Clear();
repeat
  strSuggestion := SpellChecker1.GetSuggestion(Edit1.Text, nCount);
  nCount := nCount + 1;
  if (strSuggestion <> '') then
      ListBox1.Items.Add(strSuggestion);
until( strSuggestion = '' );
end;
```

11. For button "AddWord" put this code:

```
SpellChecker1.AddWord(Edit1.Text);
```

12. Run application
13. Click "Open Dictionary"
14. Click "About"
15. Write a word in edit box
16. Click "IsWordExist"

    If word exist, label will be "Exists"

    If word doesn't exist, label will be "Doesn’t Exist"

17. Click "GetSuggestion"

    There will be list of suggestions for that word in list-box

18. Write non existing word in edit box
19. Click "IsWordExist"

    label will be "Doesn’t Exist"

20. Click "AddWord"
21. Click "IsWordExist"

    label will be "Exists"

BOOL OpenDictionary(LPCTSTR pszMainDictFileName, LPCTSTR pszCustomDictFileName)

Visual Basic: OpenDictionary String, String

Borland Delphi : OpenDictionary(WideString, WideString)

Borland C++ Builder: VARIANT_BOOL OpenDictionary(BSTR pszMainDictFileName, BSTR pszCustomDictFileName

pszMainDictFileName pointer to a string that holds name of main dictionary

pszCustomDictFileName pointer to a string that holds name of custom dictionary

Return Value

TRUE Opening was successful

FALSE Opening was not successful

Remarks

Opens main and custom dictionary. If custom dictionary does not exist, it will be created.

If pszMainDictFileName or pszCustomDictFileName is not full path, path of directory where OCX is located will be added. This method must be called before using any other method.

See also

CloseDictionary

void CloseDictionary()

Visual Basic: CloseDictionary

Borland Delphi : CloseDictionary

Borland C++ Builder: void CloseDictionary(void)

Remarks

Closes active dictionary and frees memory. It is not necessary to call this method, because dictionary will be automatically closed when program ends or if another dictionary is opened with OpenDictionary.

See also

OpenDictionary

BOOL IsWordExist (LPCTSTR pszWord)

Visual Basic: IsWordExist String

Borland Delphi : IsWordExist(WideString)

Borland C++ Builder: VARIANT_BOOL:IsWordExist(BSTR pszWord)

pszWord pointer to a string that holds the word

Return Value

TRUE word exists

FALSE word does not exist

Remarks

Use this method to see if word exists in currently opened dictionary (including custom dictionary)

See also

GetSuggestion

BSTR GetSuggestion (LPCTSTR pszWord, short nSuggestionIndex)

Visual Basic: GetSuggestion String, Integer

Borland Delphi : GetSuggestion(WideString, Smallint)

Borland C++ Builder: BSTR GetSuggestion(BSTR pszWord, short nSuggestionIndex)

pszWord	pointer to a string that holds the word
nSuggestionIndex	index number of suggestion

Return Value

suggestion word

Remarks

Use this method to get suggestions for word. To get all suggestions loop nSuggestionIndex from 0, and increase it until empty string is returned.

See also

IsWordExist

GetReplacement

void AddWord (LPCTSTR pszNewWord)

Visual Basic: AddWord String

Borland Delphi : AddWord(WideString)

Borland C++ Builder: void AddWord(BSTR pszNewWord)

pszNewWord pointer to a string that holds the word

Remarks

Use this method to add new word to custom dictionary

See also

AddToChangeAll

GetReplacement

void AddToChangeAll (LPCTSTR pszFind, LPCTSTR pszReplace)

Visual Basic: AddToChangeAll String, String

Borland Delphi : AddToChangeAll(WideString, WideString)

Borland C++ Builder: void AddToChangeAll(BSTR pszFind, BSTR pszReplace)

pszFind pointer to a string that holds the word that will be replaced

pszReplace pointer to a string that holds the word that will be used as replacement

Remarks

Use this method to add word for automatic replacement (Change All). Replacement word will be returned with method GetReplacement.

See also

AddWord

GetReplacement

AddToIgnoreAll

BSTR GetReplacement (LPCTSTR pszWord)

Visual Basic: GetReplacement String

Borland Delphi : GetReplacement(WideString)

Borland C++ Builder: BSTR GetReplacement(BSTR pszWord)

pszWord pointer to a string that holds the word

Return Value

replacement word or empty string if there is no replacement

Remarks

Use this method to get replacement for word that was added using AddToChangeAll method. This method should be used before GetSuggestion to check if word was added for Change All.

See also

AddToChangeAll

GetSuggestion

BSTR CheckText (LPCTSTR pszText, short *nExitStatus)

Visual Basic: CheckText(pszText As String, nExitStatus As Integer)

Borland Delphi : function CheckText(const Wide String pszText; var nExitStatus : SmallInt) : WideString

Borland C++ Builder: BSTR CheckText(BSTR pszText, short* nExitStatus)

pszText Specifies the text to be checked.

nExitStatus Returns error status:

-1	Error, dictionary has not been opened.
0	String has been checked.
1	User has clicked Cancel button in the Spelling Checker dialog-box.

Return Value

Returns checked text with wanted exchanges.

Remarks

Use this method to check written text with options for suggesting words (AlwaysSuggest) and ignoring words with numbers (IgnoreWordsWithNumbers) and words in uppercase (IgnoreWordsInUppercase). This method has three exit statuses. One of them is status of proper spell checking and other two appear when spell checking is skipped. Spell checking can be skipped if the dictionary is not opened or if user cancels spell checking. In those cases nExitStatus will contain nonzero value.

Example

The following example checks the content of a text box.

```
Dim Result As Integer
If SpellChecker1.OpenDictionary("c:\xxx\spellus.lex", "custom.dic") Then
   CheckedText = SpellChecker1.CheckText(Text1.Text, Result)
   Text1.Text = CheckedText
Else
   MsgBox "Cannot open dictionary"
End If
```

BSTR CheckTextVBS (LPCTSTR pszText)

Visual Basic: Function CheckTextVBS(pszText As String) As String

Borland Delphi : function CheckTextVBS(String pszText) : WideString

Borland C++ Builder: BSTR CheckTextVBS(BSTR pszText)

pszText Specifies the text to be checked.

Return Value

Returns checked text with wanted exchanges.

Remarks

Use this method to check written text with options for suggesting words (AlwaysSuggest) and ignoring words with numbers (IgnoreWordsWithNumbers) and words in uppercase (IgnoreWordsInUppercase). Use this method in VBScript instead CheckText, because VBScript does not support passing parameters by reference.

Example

The following example checks the content of a text box.

```
Dim Result As Integer
If SpellChecker1.OpenDictionary("c:\xxx\spellus.lex", "custom.dic") Then
   CheckedText = SpellChecker1.CheckTextVBS(Text1.Text)
   Text1.Text = CheckedText
Else
   MsgBox "Cannot open dictionary"
End If
```

void AboutBox()

Visual Basic: AboutBox

Borland Delphi : AboutBox

Borland C++ Builder: void AboutBox(void)

Remarks

Shows Polar SpellChecker about box.

BOOL Reset()

Visual Basic: Reset

Borland Delphi : Reset

Borland C++ Builder: VARIANT_BOOL Reset(void)

Return Value

Returns TRUE on success.

Remarks

Use this method to reset opened dictionary. This method will delete contents of buffers storing words added by methods Ignore All, Change All,...This method is substitute for closing and opening dictionary again.

BOOL IsCharAlpha (short ch)

Visual Basic: IsCharAlpha Integer

Borland Delphi : IsCharAlpha(Smallint)

Borland C++ Builder: BOOL IsCharAlpha(short)

ch char ordinal number (e.g. A = 65)

Return Value

TRUE char is alphabetical

FALSE char is not alphabetical

Remarks

Use this method to check is character alphabetical

See also

IsCharLower

IsCharUpper

BOOL IsCharLower(short ch)

Visual Basic: IsCharLower Integer

Borland Delphi : IsCharLower(Smallint)

Borland C++ Builder: BOOL IsCharLower(short)

ch char ordinal number (e.g. A = 65)

Return Value

TRUE char is lower case

FALSE char is not lower case

Remarks

Use this method to check is character lower case

See Also

IsCharAlpha

IsCharUpper

BOOL IsCharUpper(short ch)

Visual Basic: IsCharUpper Integer

Borland Delphi : IsCharUpper(Smallint)

Borland C++ Builder: BOOL IsCharUpper(short)

ch char ordinal number (e.g. A = 65)

Return Value

TRUE char is upper case

FALSE char is not upper case

Remarks

Use this method to check is character upper case

See also

IsCharAlpha

IsCharLower

void AddToIgnoreAll (LPCTSTR pszWord)

Visual Basic: AddToIgnoreAll String

Borland Delphi : AddToIgnoreAll(WideString)

Borland C++ Builder: void AddToIgnoreAll(BSTR pszWord)

pszWord pointer to a string that holds the word

Remarks

Use this method to add word to ignore list

See also

AddToChangeAll

BSTR GetLanguageName(LPCTSTR pszFileName)

Visual Basic: GetLanguageName String

Borland Delphi : GetLanguageName(WideString)

Borland C++ Builder: BSTR GetLanguageName(BSTR pszDictFileName)

pszFileName pointer to a string that holds the name of dictionary. If full path is not specified than directory of OCX will be used.

Return Value

name of the language in dictionary or empty string if file was not found.

Remarks

Use this method to get full name of language in dictionary specified by pszFileName

void SetDelphi(BOOL bNewValue)

BOOL GetDelphi()

Borland Delphi: Delphi

bNewValue new state for "Delphi" property

Remarks

Use this property for checking or setting if working environment is Borland Delphi.

In Delphi add line to your code to set this property to TRUE:

```
SpellChecker1.Delphi := TRUE;
```

before using any other method. In other developing environments this property does not need to be changed.

void SetAlwaysSuggest(BOOL bNewValue)

BOOL GetAlwaysSuggest()

Visual Basic: AlwaysSuggest

Borland Delphi : AlwaysSuggest

Borland C++ Builder: AlwaysSuggest()

bNewValue new state for AlwaysSuggest property

Return Value

TRUE	SpellChecker will generate a list of suggested corrections for each unmatched word
FALSE	SpellChecker will not make suggestions

Remarks

This property defines if a list of suggested corrections for each unmatched word will be displayed when user checks spelling. It influences only while using CheckText method. Default value of this property is TRUE and can be changed in a dialog-box.

void SetIgnoreWordsInUppercase(BOOL bNewValue)

BOOL GetIgnoreWordsInUppercase()

Visual Basic: IgnoreWordsInUppercase

Borland Delphi : IgnoreWordsInUppercase

Borland C++ Builder: IgnoreWordsInUppercase()

bNewValue new state for IgnoreWordsInUppercase property

Return Value

TRUE words written in uppercase will be ignored

FALSE words written in uppercase will not be ignored

Remarks

This property defines if words in which each character is an uppercase letter will be ignored. It influences only while using CheckText method. Default value of property IgnoreWordsInUppercase is TRUE and can be changed in a dialog-box..

See also

IgnoreWordsWithNumbers

void SetIgnoreWordsWithNumbers(BOOL bNewValue)

BOOL GetIgnoreWordsWithNumbers()

Visual Basic: IgnoreWordsWithNumbers

Borland Delphi : IgnoreWordsWithNumbers

Borland C++ Builder: IgnoreWordsWithNumbers()

bNewValue new state for IgnoreWordsWithNumbers property

Return Value

TRUE words with numbers will be ignored

FALSE words with numbers will not be ignored

Remarks

This property defines if words containing numbers will be ignored. It influences only while using CheckText method. Default value of property IgnoreWordsWithNumbers is TRUE and can be changed in a dialog-box.

See also

IgnoreWordsInUppercase

Polar SpellChecker DLL

Polar SpellChecker DLL package consists of following files:

Polspell.Dll - dynamic-link library (DLL)

Polspell.Lib - Visual C++ library file

Polspell.Def - Polspell.Dll definition file

Polspell.H - Header for Polspell.Dll

Mfcspell.H - MFC wrapper for Polspell.Dll

This help includes syntax and definition only for functions declared in Polspell.H header file. If you choose to use **Polar SpellChecker** using MFC wrapper, use functions declared in Mfcspell.H header file. Those functions have different declaration but definitions and syntax are same as described here.

Example

This example shows how to use *PS_CheckText* function using DLL APIs and MFC wrapper.

```
void CTestDLLDlg::OnBtnSpell()
{
  char szText[10240];
// using DLL APIs
  if(GetDlgItemText(IDC_EDIT_TEXT, szText, 10240) > 0)
  {
    HSPELL hSpell = PS_Init();
    if(PS_OpenDictionary(hSpell, "spellus.lex", "custom.dic"))
    {
     short nError;
     HGLOBAL hglb = PS_CheckText(hSpell, szText, &nError);
     if(hglb)
     {
       LPTSTR pszChecked = (LPTSTR)GlobalLock(hglb);
       if(pszChecked)
       {
         SetDlgItemText(IDC_EDIT_CHECKED, pszChecked);
       }
       GlobalUnlock(hglb);
       GlobalFree(hglb);
     }

     PS_CloseDictionary(hSpell);
  }
  PS_Destroy(hSpell);
}

// using MFC wrapper

if(GetDlgItemText(IDC_EDIT_TEXT, szText, 10240) > 0)
```

```
{
 TSpellingChecker SpellCheck;

 if(SpellCheck.OpenDictionary("spellus.lex", "custom.dic"))
 {
   short nError;

   CString strCheckedText = SpellCheck.CheckText(szText, &nError);
   SetDlgItemText(IDC_EDIT_CHECKED_MFC, strCheckedText);

  SpellCheck.CloseDictionary();
 }
}
```

BOOL DownloadDictionaries(BSTR pszDownloadURL, BSTR pszInstallDir)

Visual Basic: DownloadDictionaries (*pszDownloadURL* As String, *pszInstallDir* As String) As Boolean

Borland Delphi : DownloadDictionaries(const *pszDownloadURL* : WideString, const *pszInstallDir* : WideString)

Borland C++ Builder: BOOL DownloadDictionaries(BSTR pszDownloadURL, BSTR pszInstallDir)

Downloads dictionaries from the Internet, Intranet or LAN.

pszDownloadURL string that holds the URL of the download redirection file (if you leave this as an empty string, your end users will be pointed to our site)

pszInstallDir string that holds the path of the directory that dictionaries will be downloaded to (if you leave this as an empty string, dictionaries will be downloaded to the same location where the "polspell.ocx" file is located)

Return Value

True	dictionaries are successfully downloaded
False	An error occurred during downloading, or user canceled the operation

Remarks

This function can be very useful if you are making app for international use. You can place dictionaries on your site, and let your users to download one they need. This function connects to the Internet and offers dialog for downloading dictionaries. On user's response, downloads selected dictionaries. How to get dictionaries? We offer you to download 12 dictionaries from our site, or you can make your own dictionaries using Polar SpellMaker.

Using this function, you must check the following:

1. Additional DLLs are required
2. Format of the download redirection file
3. Internet connection on your end-users machine

Additional DLLs are required

This function requires additional DLLs for execution:

gteinet.dll	Module for download dictionary options
GwdUnz32.dll	Module for download dictionary options

These files must be redistributed with polspell.ocx and they must be copied to the win/sys directory, or to the same location where the polspell.ocx is placed.

Format of the download redirection file

It is very important that *pszDownloadURL* is not location of dictionaries. It is location of text file, saved in **".ins"** format. That file must contain informations about dictionaries that are offered for download.

See Format of the download redirection file

Note

If you leave *pszUpdateURL* as an empty string, your users will be pointed to our site, what we do not suggest. We will often change locations of dictionaries, and the dialog that offers dictionaries will contain our commercial messages.

Internet connection on your end-users machine

To detect the type of access to the Internet when downloading dictionaries, Polar SpellChecker looks through *Control Panel*. So be sure that the *Internet* option in *Control Panel* is set to the proper type of connection. If the machine has access to the Internet via proxy server, be sure everything is set in the *Control Panel*.

BOOL OptionsDlg(BSTR pszDictionaryDir, BSTR * pbstrCustomDicFileName, BSTR * pbstrCurDicFileName, BSTR pszDownloadURL, boolean bInternetSupport)

Visual Basic:	OptionsDlg(*pszDictionaryDir* As String, *pbstrCustomDicFileName* As String, *pbstrCurDicFileName* As String, *pszDownloadURL* As String, *bInternetSupport* As Boolean) As Boolean
Borland Delphi :	OptionsDlg(const *pszDictionaryDir* : WideString, var *pbstrCustomDicFileName* : WideString, var *pbstrCurDicFileName* : WideString, const *pszDownloadURL* : WideString, *bInternetSupport : WordBool*) : WordBool
Borland C++ Builder:	BOOL OptionsDlg(BSTR pszDictionaryDir, BSTR * pbstrCustomDicFileName, BSTR * pbstrCurDicFileName, BSTR pszDownloadURL, boolean bInternetSupport);

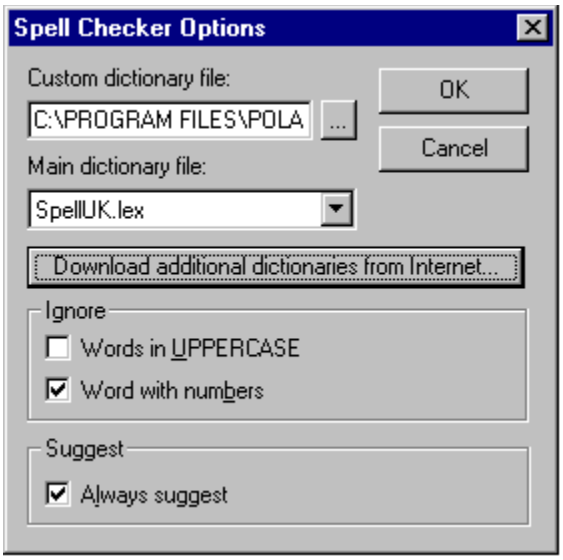
pszDictionaryDir	string that holds the path of the directory that contains dictionaries and where they will be downloaded to if the *bInternetSupport* is set to True (if you leave this as an empty string, dictionaries will be downloaded to the same location where the "polspell.ocx" file is located)
pbstrCustomDicFileName	string that holds the full path and name of the current custom dictionary
pbstrCurDicFileName	string that holds current main dictionary file name on local disk (relative to program directory)
pszDownloadURL	string that holds the URL of the download redirection file (if you leave this as an empty string, your end users will be pointed to our site)
bInternetSupport	A Boolean expression that determins whether this Options dialog will contain ability of downloading dictionaries

Return Value

Returns nonzero value on success.

Remarks

Using this function, with a single line of code, dialog box for setting all necessary spell checking options will appear:



Choose path and name of the current custom dictionary
Select current main dictionary
Download dictionaries from the Internet, Intranet or LAN (optional)
Set options: IgnoreWordsInUppercase, IgnoreWordsWithNumbers, AlwaysSuggest;

Important

If you set *bInternetSupport* parameter to True, you will enable your users ability of downloading dictionaries. Therefor, you must check the following:

1. Additional DLLs are required
2. Format of the download redirection file
3. Internet connection on your end-users machine

Additional DLLs are required

This function requires additional DLLs for execution:

gteinet.dll	Module for download dictionary options
GwdUnz32.dll	Module for download dictionary options

These files must be redistributed with polspell.ocx and they must be copied to the win/sys directory, or to the same location where the polspell.ocx is placed.

Format of the download redirection file

It is very important that *pszDownloadURL* is not location of dictionaries. It is location of text file, saved in **".ins"** format. That file must contain informations about dictionaries that are offered for download.

See Format of the download redirection file

Internet connection on your end-users machine

To detect the type of access to the Internet when downloading dictionaries, Polar SpellChecker looks through *Control Panel*. So be sure that the *Internet* option in *Control Panel* is set to the proper type of

connection. If the machine has access to the Internet via proxy server, be sure everything is set in the *Control Panel*.

Format of the download redirection file

URL parameter in DownloadDictionaries and OptionsDlg methods is not location of dictionaries. It is location of text file, saved in ".ins" format. That file must contain information about dictionaries that are offered for download.

It must be written in format:

```
dictionary_file_name, language_name, dictionary_date, URL, file_size
```

SpellUK.lex, English (British), 15.03.99, http://www.polarsoftware.com/download/activex/spellchecker/SpellUK.zip, 336771

SpellFRA.lex, French, 15.03.99, http://www.polarsoftware.com/download/activex/spellchecker/SpellFRA.zip, 255152

SpellGER.lex, German, 15.03.99, http://www.polarsoftware.com/download/activex/spellchecker/SpellGER.zip, 697479

Part	Description
`dictionary_file_name`	Script file name on local disk (relative to program directory)
`language_name`	Name of the language (Do not use commas in the language name)
`dictionary_date`	File date in the format dd.mm.yyyy (day.month.year) year can be represent with 2 or 4 digits (Y2K)
`URL`	Location of the dictionary
`file_size`	Aproximate file size, expressed in bytes (important for the progress)
`;comment`	Comment
`*URL Message`	Commecial messages (appears in the left-bottom corner of the dialog box, as the hyperlink to URL location e.g. `*http://www.polarsoftware.com Download Polar Calculator`)

You can see sample file "dictionaries.ins" which came with the installation of Polar SpellChecker.

Dictionaries can be located on Internet, Intranet or local machine. They also must be zipped.

Note

If you leave URL as an empty string, your users will be pointed to our site, what we do not suggest. We will often change locations of dictionaries, and the dialog that offers dictionaries will contain our commercial messages.

PS_AddToChangeAll

void PS_AddToChangeAll(HSPELL *hSpell*, **LPCTSTR** *pszFind*, **LPCTSTR** *pszReplace*)

Use this function to add word for automatic replacement (Change All). Replacement word will be returned with PS_GetReplacement function.

Parameters

hSpell
handle to spelling checker

pszFind
pointer to a string that holds the word that will be replaced

pszReplace
pointer to a string that holds the word that will be used as replacement

PS_AddToIgnoreAll

void PS_AddToIgnoreAll(HSPELL *hSpell*, **LPCTSTR** *pszWord*)

Use this function to add word to ignore list.

Parameters

hSpell
handle to spelling checker

pszWord
pointer to a string that holds the word

PS_AddWord

void PS_AddWord(HSPELL *hSpell*, **LPCTSTR** *pszNewWord*)

Use this function to add new word to custom dictionary.

Parameters

hSpell
handle to spelling checker

pszNewWord
pointer to a string that holds the word

See Also

PS_AddToChangeAll

PS_GetReplacement

PS_CheckText

HGLOBAL PS_CheckText(HSPELL *hSpell*, **LPCTSTR** *pszText*, **short *** *nExitStatus*)

Use this function to check written text with options for suggesting words (PS_GetAlwaysSuggest, PS_SetAlwaysSuggest) and ignoring words with numbers (PS_GetIgnoreWordsWithNumbers, PS_SetIgnoreWordsWithNumbers) and words in uppercase (PS_GetIgnoreWordsInUppercase, PS_SetIgnoreWordsInUppercase). This function has three exit statuses. One of them is status of proper spell checking and other two appear when spell checking is skipped. Spell checking can be skipped if the dictionary is not opened or if user cancels spell checking. In those cases *nExitStatus* parameter will contain nonzero value.

Return Value

Returns handle to global memory which contains checked text, or NULL on error/empty string.

Parameters

hSpell
handle to spelling checker

pszText
text to be checked

nExitStatus
Returns error status:
-1 Error, dictionary has not been opened.
0 String has been checked.
1 User has clicked Cancel button in the Spelling Checker dialog-box.

PS_CloseDictionary

void PS_CloseDictionary(HSPELL *hSpell***)**

Closes the active dictionary and frees memory. It is not necessary to call this function, because dictionary will be automatically closed when you call PS_Destroy function or if another dictionary is opened with PS_OpenDictionary.

Parameters

hSpell
handle to spelling checker

PS_Destroy

void PS_Destroy(HSPELL *hSpell***)**

The PS_Destroy function deletes spelling checker instance created with PS_Init function.

Parameters

hSpell
handle to spelling checker

PS_DownloadDictionaries

BOOL PS_DownloadDictionaries(HSPELL *hSpell*, **LPCTSTR** *pszUpdateURL*, **LPCTSTR** *pszInstallDir*)

Connects to the Internet and offers dialog for downloading dictionaries. On user's response, downloads selected dictionaries.

Parameters

hSpell
handle to spelling checker

pszUpdateURL
string that holds the URL of the download redirection file (if you leave this as an empty string, your end users will be directed to our site)

pszInstallDir
string that holds the path of the directory that dictionaries will be downloaded to (if you leave this as an empty string, dictionaries will be downloaded to the same location where the "polspell.dll" file is located)

Return Value

Returns nonzero value on success.

Remarks

This function can be very useful if you are making app for international use. You can place dictionaries on your site, and let your users to download one they need. How to get dictionaries? We offer you to download 12 dictionaries from our site, or you can make your own dictionaries using Polar SpellMaker.

This function will connect user's machine to the Internet and offer dialog for downloading dictionaries. On user's response it will start downloading selected dictionaries.

Using this function, you must check the following:

1. Additional DLLs are required
2. Format of the download redirection file
3. Internet connection on your end-users machine

Additional DLLs are required

This function requires additional DLLs for execution:

gteinet.dll	Module for download dictionary options
GwdUnz32.dll	Module for download dictionary options

These files must be redistributed with polspell.dll and they must be copied to the win/sys directory, or to the same location where the polspell.dll is placed.

Format of the download redirection file

It is very important that *pszUpdateURL* is not location of dictionaries. It is location of text file, saved in

".ins" format. That file must contain informations about dictionaries that are offered for download.

See Format of the download redirection file

Note

If you leave *pszUpdateURL* as an empty string, your users will be pointed to our site, what we do not suggest. We will often change locations of dictionaries, and the dialog that offers dictionaries will contain our commercial messages.

Internet connection on your end-users machine

To detect the type of access to the Internet when downloading dictionaries, Polar SpellChecker looks through *Control Panel*. So be sure that the *Internet* option in *Control Panel* is set to the proper type of connection. If the machine has access to the Internet via proxy server, be sure everything is set in the *Control Panel*.

PS_GetAlwaysSuggest

BOOL PS_GetAlwaysSuggest(HSPELL *hSpell***)**

Returns nonzero value if a list of suggested corrections for each unmatched word will be displayed when user checks spelling. It influences only while using PS_CheckText function.

Parameters

hSpell
handle to spelling checker

See Also

PS_SetAlwaysSuggest

PS_GetIgnoreWordsInUppercase

BOOL PS_GetIgnoreWordsInUppercase(HSPELL *hSpell***)**

Returns nonzero value if words in which each character is an uppercase letter will be ignored. It influences only while using function.

Default value of this option is TRUE and can be changed in a dialog-box.

Parameters

hSpell
handle to spelling checker

See Also

PS_SetIgnoreWordsInUppercase

PS_GetIgnoreWordsWithNumbers

BOOL PS_GetIgnoreWordsWithNumbers(HSPELL *hSpell***)**

Returns nonzero value if words containing numbers will be ignored. It influences only while using function.

Default value of this option is TRUE and can be changed in a dialog-box.

Parameters

hSpell
handle to spelling checker

See Also

PS_SetIgnoreWordsWithNumbers

PS_GetLanguageName

BOOL PS_GetLanguageName(HSPELL *hSpell*, **LPCTSTR** *pszDictFileName*, **LPTSTR** *pszLangName*, **unsigned** *nSize*)

Use this function to get full name of language in dictionary specified by *pszDictFileName*.

Return Value

Returns nonzero value on success.

Parameters

hSpell
handle to spelling checker

pszDictFileName
pointer to a string that holds the name of dictionary

pszLangName
Points to the buffer that is to receive the null-terminated language name.

nSize
Specifies the maximum number of bytes to copy, including the terminating null character.

PS_GetReplacement

BOOL PS_GetReplacement(HSPELL *hSpell*, **LPCTSTR** *pszWord*, **LPTSTR** *pszReplacement*, **unsigned** *nSize*)

Use this function to get replacement for word that was added using PS_AddToChangeAll function. This function should be used before PS_GetSuggestion to check if word was added for Change All.

Return Value

Returns zero if there is no replacement for that word.

Parameters

hSpell
handle to spelling checker

pszWord
pointer to a string that holds the word

pszReplacement
Points to the buffer that is to receive the null-terminated replacement string.

nSize
Specifies the maximum number of bytes to copy, including the terminating null character.

See Also

PS_AddToChangeAll

PS_GetSuggestion

PS_GetSuggestion

BOOL PS_GetSuggestion(HSPELL *hSpell*, **LPCTSTR** *pszWord*, **short** *nSuggestionIndex*, **LPTSTR** *pszSuggestion*, **unsigned** *nSize*)

Use this function to get suggestions for word. To get all suggestions, loop *nSuggestionIndex* from 0, and increase it until FALSE is returned.

Parameters

hSpell
handle to spelling checker

pszWord
pointer to a string that holds the word

nSuggestionIndex
zero based index number of suggestion

pszSuggestion
Points to the buffer that is to receive the null-terminated suggestion string.

nSize
Specifies the maximum number of bytes to copy, including the terminating null character.

PS_Init

HSPELL PS_Init(void)

The PS_Init function creates new spelling checker instance. After this function you must call PS_OpenDictionary function.

Return Value

Handle of the spelling checker

PS_IsCharAlpha

BOOL PS_IsCharAlpha(HSPELL *hSpell***, short** *ch***)**

Use this function to check if character is alphabetical.

Return Value

Returns nonzero if a character is alphabetical.

Parameters

hSpell
handle to spelling checker

ch
character to test

PS_IsCharLower

BOOL PS_IsCharLower(HSPELL *hSpell*, **short** *ch*)

Returns nonzero if a character is lowercase.

Parameters

hSpell
handle to spelling checker

ch
character to test

PS_IsCharUpper

BOOL PS_IsCharUpper(HSPELL *hSpell*, **short** *ch*)

Returns nonzero if a character is uppercase.

Parameters

hSpell
handle to spelling checker

ch
character to test

PS_IsWordExist

BOOL PS_IsWordExist(HSPELL *hSpell*, **LPCTSTR** *pszWord***)**

Use this function to see if word exists in currently opened dictionary (including custom dictionary).

Return Value

Returns nonzero value if word exists.

Parameters

hSpell
handle to spelling checker

pszWord
pointer to a string that holds the word

PS_OpenDictionary

BOOL PS_OpenDictionary(HSPELL *hSpell*, **LPCTSTR** *pszMainDictFileName*, **LPCTSTR** *pszCustomDictFileName*)

Opens main and custom dictionary. If custom dictionary does not exist, it will be created. This function must be called before using any other function.

Return Value

Returns nonzero value on success.

Parameters

hSpell
handle to spelling checker (see PS_Init)

pszMainDictFileName
pointer to a string that holds name of main dictionary

pszCustomDictFileName
pointer to a string that holds name of custom dictionary

See Also

PS_CloseDictionary

PS_Init

PS_OptionsDialog

BOOL PS_OptionsDialog(HSPELL *hSpell***, LPCTSTR** *pszDictionaryDir***, LPTSTR** *pszCustomDicFileName***, unsigned** *nCustomDicSize***, LPTSTR** *pszCurDicFileName***, unsigned** *nCurDicSize***, LPCTSTR** *pszURL***, BOOL** *bInternetSupport***)**

Invokes dialog box for setting all necessary spell checking options

Parameters

hSpell
handle to spelling checker

pszDictionaryDir
string that holds the path of the directory that contains dictionaries and where they will be downloaded to if the *bInternetSupport* is set to True (if you leave this as an empty string, dictionaries will be downloaded to the same location where the "polspell.dll" file is located)

pszCustomDicFileName
Points to a buffer that contains a full path and name of the current custom dictionary file used to initialize the **Custom Dictionary File** edit control. When this function returns, this buffer contains the drive designator, path, file name, and extension of the selected file.

nCustomDicSize
Maximum number of characters to copy to the pszCustomDicFileName buffer. If the text exceeds this limit, it is truncated.

pszCurDicFileName
Points to a buffer that contains a current main dictionary file name on local disk (relative to program directory) used to initialize the **Main Dictionary file** edit control. When this function returns, this buffer contains the drive designator, path, file name, and extension of the selected file.

nCurDicSize
Maximum number of characters to copy to the *pszCurDicFileName* buffer. If the text exceeds this limit, it is truncated.

pszURL
string that holds the URL of the download redirection file (if you leave this as an empty string, your end users will be pointed to our site)

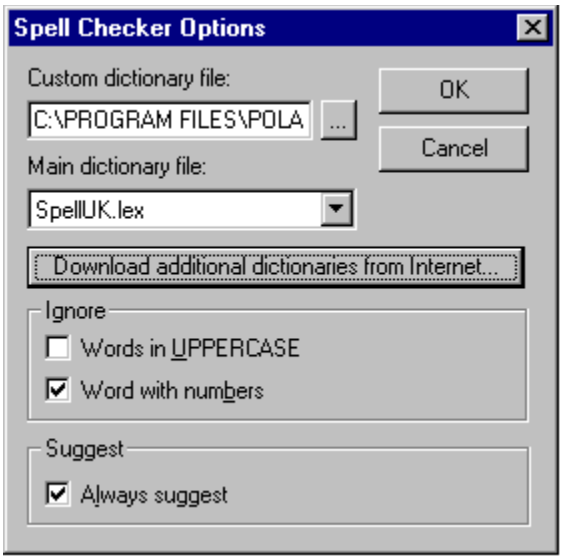
bInternetSupport
A Boolean expression that determins whether this Options dialog will contain ability of downloading dictionaries

Return Value

Returns nonzero value on success.

Remarks

Using this function, with a single line of code, dialog box for setting all necessary spell checking options will appear:



Choose path and name of the current custom dictionary
Select current main dictionary
Download dictionaries from the Internet, Intranet or LAN (optional)
Set options: IgnoreWordsInUppercase, IgnoreWordsWithNumbers, AlwaysSuggest;

Important

If you set *bInternetSupport* parameter to True, you will enable your users ability of downloading dictionaries. Therefor, you must check the following:

1. Additional DLLs are required
2. Format of the download redirection file
3. Internet connection on your end-users machine

Additional DLLs are required

This function requires additional DLLs for execution:

gteinet.dll	Module for download dictionary options
GwdUnz32.dll	Module for download dictionary options

These files must be redistributed with polspell.dll and they must be copied to the win/sys directory, or to the same location where the polspell.dll is placed.

Format of the download redirection file

It is very important that *pszDownloadURL* is not location of dictionaries. It is location of text file, saved in **".ins"** format. That file must contain informations about dictionaries that are offered for download.

See Format of the download redirection file

Internet connection on your end-users machine

To detect the type of access to the Internet when downloading dictionaries, Polar SpellChecker looks through *Control Panel*. So be sure that the *Internet* option in *Control Panel* is set to the proper type of

connection. If the machine has access to the Internet via proxy server, be sure everything is set in the *Control Panel*.

PS_Reset

BOOL PS_Reset(HSPELL *hSpell***)**

Use this method to reset opened dictionary. This method will delete contents of buffers storing words added by IgnoreAll, ChangeAll,... This method is substitute for closing and opening dictionary again.

Parameters

hSpell
handle to spelling checker

Return Value

Returns nonzero value on success.

See Also

PS_OpenDictionary

PS_CloseDictionary

PS_SetAlwaysSuggest

void PS_SetAlwaysSuggest(HSPELL *hSpell*, **BOOL** *bNewValue*)

Sets Always Suggest option.

Parameters

hSpell
handle to spelling checker

bNewValue
Specifies Always Suggest option. This parameter must be TRUE or FALSE.

See Also

PS_GetAlwaysSuggest

PS_SetIgnoreWordsInUppercase

void PS_SetIgnoreWordsInUppercase(HSPELL *hSpell*, **BOOL** *bNewValue*)

Sets Ignore Words in Uppercase option.

Parameters

hSpell
handle to spelling checker

bNewValue
Specifies Ignore Words in Uppercase option. This parameter must be TRUE or FALSE.

See Also

PS_GetIgnoreWordsInUppercase

PS_SetIgnoreWordsWithNumbers

void PS_SetIgnoreWordsWithNumbers(HSPELL *hSpell*, **BOOL** *bNewValue*)

The PS_SetIgnoreWordsWithNumbers function sets Ignore Words With Numbers option.

Parameters

hSpell
handle to spelling checker

bNewValue
Specifies Ignore Words With Numbers option. This parameter must be TRUE or FALSE.

See Also

PS_GetIgnoreWordsWithNumbers