About Visual Speller

VisualSpeller is an OCX custom control that allows you to add spell checking to your application with a minimum of programming.

VisualSpeller is primarily a spell checking enginea tool for adding spell checking functionality to a higher-level application. Using it is a matter of passing in text for spell checking and reading the corrected text when spell checking is complete. Standard features include:

- **Multiple standard dictionaries** You can access multiple standard dictionaries during a spell check.
- **Multiple custom dictionaries** You can create dictionaries customized for specific documents, activities, departments, organizations, and so on. You can access multiple custom dictionaries and add words to custom dictionaries during a spell check.
- **Ignore/replace lists** For each dictionary, VisualSpeller maintains a list of words to be ignored and a list of words to be replaced by other words. When one of these words appears in a spell check, VisualSpeller either acts automatically or prompts the user, according to your instructions.
- **Word or block checking** Text to be spell checked can be passed a word at a time or in multiword blocks.
- **Automatic suggestion generation** You can generate suggested spellings either automatically or on demand. The number of suggestions can be limited in order to improve performance.
- **Shared memory for dictionaries** This reduces the total amount of memory needed when multiple users access the same dictionary.
- **Extensive customization capability** You can use default settings or customize almost every aspect of operation.
- **Standard dictionary maker** This application allows you to create your own standard dictionaries and manage their contents.

Technical Support

The Visual Components technical support staff is ready to help you with any problem you have using VisualSpeller. If you need help, contact Visual Components in any of the following ways:

- **By phone.** You can contact our technical support staff at 913-599-6500 on weekdays between 8:30 and 5:30 Central Time.
- **By FAX.** You can contact us by FAX at 913-599-6597.
- **Via BBS.** You can contact us through our 24-hour bulletin board service at 913-599-6713.
- **Via CompuServe.** You can contact us through CompuServe74774,443.

  Visual Components also maintains a message and library section in the MS Windows Components A+ Forum on CompuServe. These sections are used for peer to peer suppor and the distribution of example projects, maintenance releases, etc. To reach the Visual Components section, type:

  GO VISTOOLS

  When communicating with Visual Components via the CompuServe forums, include our account number with all messages. This assures that your message receives prompt attention.
- **On the World Wide Web.** Contact us at www.visualcomp.com
- **By mail.** Please send correspondence to the following address:
  Customer Service Department
  Visual Components, Inc.
  15721 College Blvd.
  Lenexa, KS 66219

In Europe, contact:

Visual Components Europe
Lenexa House
11 Eldon Way
Paddock Wood, Kent
England TN12 6BE

Phone +44 1892 834343
Fax +44 1892 835843
BBS +44 1892 835579

Distributing a VisualSpeller application

When you develop an application using VisualSpeller, you can distribute run time versions of it subject to the conditions in your Visual Components license agreement.

You can use all of the files included with VisualSpeller as you develop your application. The 32-bit run time files are as follows:

File	Description
VSPELL32.OCX	VisualSpeller OCX custom control
VSPELL32.DLL	VisualSpeller Dynamic Link Library
AMERICAN.VTD	U.S. English dictionary supplied with VisualSpeller.
VSPELL.HLP	Context sensitive help for the Word Not Found and Spell Options dialog boxes.

In addition, running 32-bit VisualSpeller applications require the presence of the latest versions of MFCANS32.DLL, OC30.DLL, and MSVCRT20.DLLon your user's systems. These files are provided on your VisualSpeller installation disks.

The 16-bit run time files are as follows:

File	Description
VSPELL16.OCX	VisualSpeller OCX custom control
VSPELL16.DLL	VisualSpeller Dynamic Link Library
VSPELL.HLP	Context sensitive help for the Word Not Found and Spell Options dialog boxes.
AMERICAN.VTD	U.S. English dictionary supplied with VisualSpeller.

In addition, running 16-bit VisualSpeller applications require the presence of the latest versions of OC25.DLL, OLE2DISP.DLL, and TYPELIB.DLL on your user's systems. The file OC25.DLL is provided with VisualSpeller, the other two DLLs are not.

Spellchecking Context

A context is one of the most fundamental VisualSpeller concepts. It is the specific information necessary to manage the spellchecking process for an application. In most development environments, a context is assigned automatically when you add the VisualSpeller OCX control to your project. A context contains the following information:

- Text to be spellchecked.
- Dictionaries to be used.
- Words to ignore (treat as correct) or to replace (substitute another word for).
- Information about the current misspelled word.
- Suggested spellings for the current misspelled word.
- User options for controlling spellchecker behavior.

Information not included in a context includes information about dictionaries themselves. This information is maintained by VisualSpeller on behalf of all contexts, allowing all users of a dictionary to share that dictionary.

When VisualSpeller assigns a context, much of the information it contains is default information. The application can adjust the context by assigning values to properties. One of the most important adjustments would be to open at least one standard dictionary. Opening a dictionary adds it to a list of dictionaries available in the current context. When an application opens a dictionary already in use by another context, the dictionary is shared. Both contexts can search the dictionary without interfering with each other.

Text Buffer

In a multi-word spellcheck, one performed in a **CheckText** operation rather than a **CheckWord** operation, the text to be spellchecked resides in a buffer. Placing text there and later retrieving it are the first and last steps in any multi-word spellcheck.

In some environments, such as Visual Basic, the buffer is internal and filled when you assign a string to the **Text** property. In other development environments, the buffer can be either internal or external.VisualSpeller automatically controls the size of internal buffers. If you use external buffers, you must control their sizing through your code.

Parsing

BeforeVisualSpeller can perform a **CheckText** operation, it must parse the text into discrete words. In this process,VisualSpeller considers a word to be any unbroken string comprised of alphanumeric characters, apostrophes, and hyphens, excluding leading and trailing apostrophes and hyphens. All other characters are ignored. In a **CheckWord** operation,VisualSpeller does not parse the text but instead considers it to be a single word.

Word Lookup

The most basic function of the spell checking control is to look up words in dictionaries. When it finds a word, the word is assumed to be spelled correctly. When it fails to find a word, the search continues in the ignore/replace list. This list contains exceptions specified by the user for the current spellcheck. If the word is not found in the ignore/replace list, it is assumed to be misspelled.

At that point VisualSpeller normally generates a list of suggested spellings for the word and presents them to the user for a decision. Normally,VisualSpeller searches for words first in standard dictionaries, then in custom dictionaries, then in the ignore/replace list.

Ignore/Replace Lists

Often a word is spelled correctly even though it doesnt appear in one of the open dictionaries. If the word could appear in other documents, the user may decide to add it to a custom dictionary. If, however, it is not likely to appear in other documents or if the user just doesnt want to take the time to add it to a dictionary, the user can elect to add it to an ignore/replace list.

An ignore/replace list is essentially a temporary custom dictionary. It allows dynamic adjustment of the spellchecking process. Each word in an ignore/replace list is flagged in one of the following ways:

VSIR_SPELLED_OK	Treat the word as though it is spelled correctly without asking the user for confirmation.
VSIR_REPLACE_ALL	Replace all future occurrences of the word with **ReplacementWord** without asking the user for confirmation.
VSIR_MISSPELLED	Ask user to confirm whether word is misspelled.
VSIR_PROMPT_REPLACE	Ask the user before replacing the word with **ReplacementWord**.
VSIR_HYPHENATION	Hyphenation information for the word can be added to the list. This code, if used, must be added to one of the codes listed above and the **Hyphenation** property must be set.

There are two types of ignore/replace lists:common and dictionary-specific. Both are memory resident and associated with a specific context. The common ignore/ replace list is created and enabled automatically by . A dictionary ignore/replace list is associated with a specific standard dictionary. (Custom dictionaries dont have ignore/replace lists because they are themselves reusable ignore/replace lists.) Enabling the ignore/replace list for a dictionary causes it to become an extension of the common ignore/replace list. Words added to the common ignore/replace list are also added to any dictionary ignore/replace lists that is enabled and for which the associated dictionary is enabled.

The purpose of a dictionary-specific ignore/replace list is to allow language-specific control over the replacement process. For instance, suppose you are spellchecking a document that contains both English and French paragraphs. You want to ignore instances of spizbot in English but not in French. You could accomplish this by enabling dictionary-specific ignore/replace lists, then performing separate spellchecks on the English and French sections of the document, enabling only the appropriate dictionary in each case. In a default spellcheck, dictionary ignore/replace lists are disabled.

Case Sensitivity

VisualSpeller is inherently case sensitive. That is, it always looks for an exact match for the current word. In normal usage, however, words are often capitalized, written in all caps, abbreviated, and so on. To cover most such cases,VisualSpeller looks for a misspelled word as follows:

1. Search for the word as it was originally presented.
2. If the word is in initial caps or all caps, search for the word in all lower case.
3. If the word contains hyphens and the **AllowJoinedWords** property is True, check each individual word (**CheckText** only, not **CheckWord**).
4. If the word was followed in text by a period, add a period and try steps 1 and 2 again to check for possible abbreviation (**CheckText** only, not **CheckWord**).

If any variation is found, the original word is considered correct. You can alter the way VisualSpeller handles case sensitivity by changing the **EnableSpellOptions** property. This property allows you to ignore words written in all caps, to ignore words that contain numerals, or to require an exact match.

Suggestions

When VisualSpeller identifies a misspelled word, it also generates a list of suggested replacement words. To do this, it modifies the misspelled word in a series of ways, checks each new word, and places valid ones in the list. A default search uses all of the following modification techniques, in order.

- **Capitalization**. The word is checked in all lower case, initial caps, and all caps.

  Example:Windows

  Suggestions:windows, Windows, WINDOWS
- **Splits**. A space is inserted between each successive pair of letters and the resulting two words are checked individually. If both are valid words, the pair become a suggestion.

  Example:ofthe

  Suggestions:o fthe, of the, oft he, ofth e
- **CharSwap**. Successive pairs of letters are swapped and resulting word(s) spellchecked.

  Example:adujst

  Suggestions:daujst, audjst, adjust, adusjt, adujts
- **Doubles**. Occurrences of doubled letters are moved to adjacent letters.

  Example:seetings

  Suggestions:ssetings, settings
- **Exchange**. Each letter is replaced with all other alphabetic characters.

  Example:qill

  Suggestions:aill, bill, cill, dill . . . qall, qbll, qcll, qdll . . .
- **Deletes**. Each letter is removed in sequence.

  Example:coldt

  Suggestions:oldt, cldt, codt, colt, cold
- **Hyphens**. A hyphen is inserted between successive pairs of letters.

  Example:byline

  Suggestions:b-yline, by-line, byl-ine, byli-ne, bylin-e
- **Insertions**. An extra letter is inserted between successive pairs of letters, running through the entire alphabet.

  Example:bll

  Suggestions:ball, bbll, bcll, bdll, bell . . . blal, blbl, blcl, bldl . . .

VisualSpeller stops generating suggestions when it reaches SuggestionsLimit or MaxSuggestions, whichever is greater. No matter how many suggestions are generated internally, only the number specified by **MaxSuggestions** appear in the list. Suggestions are ordered from most likely to least likely as determined by VisualSpeller.

Normally,VisualSpeller offers suggestions that match the case of the misspelled word. If the misspelled word appears in lower case, for instance,VisualSpeller presents lower case suggestions. Likewise, if the misspelled word appears in all caps or initial caps, only similarly capitalized suggestions appear in the list.

Note You can generate suggestions for any word, not just a misspelled word. For instance, you could create a function that allows the user to highlight the word trails and generate a list of alternatives, such as trials.

Standard Dictionaries

A standard dictionary is an indexed and compressed dictionary created with the Visual Components Dictionary Maker (VTDMAKER.EXE). It is not updatable and is always opened as shareable. When a standard dictionary is loaded, its index and certain other data become memory resident, but all other information is read into memory only as it is needed and managed by a caching system. The data in a standard dictionary is divided into four sections:

- Header
- Index (information on where to find parts of the word lists)
- Section of the word list that is required to be in memory
- Generally large section of the word list that resides on disk

The disk-resident section is organized in blocks. These blocks are designed so VisualSpeller must read only one of them to check the dictionary for a specific word.

In addition to the four main sections, a standard dictionary can have an optional sub-dictionary of commonly used words. This section, when present, is loaded totally in memory. The words in the common list are duplicated in the main word list of the dictionary. The common word list provides fast access to the most often used words in the dictionary.

Within properties, standard dictionaries are referenced by a dictionary index number, which is a positive number from 1 to 255. The first standard dictionary opened for a context that is still open is number 1; the second standard dictionary opened that is still open is number 2, and so on. When a dictionary is closed, other dictionaries with are renumbered to fill the gap.

Custom Dictionaries

Dictionary Caching

Maximum Number of Open Dictionaries

Enabling and Loading Dictionaries

Dictionary Categories

Custom Dictionaries

A custom dictionary is an ANSI text file in which each line represents a word. When loaded, it is copied entirely into memory and stored in a hash table. Because of this, a custom dictionary may take longer to load than a standard dictionary.

There are two types of custom dictionary: compatible and extended. A compatible dictionary is one that uses the Microsoft Word custom dictionary format. It is a simple list of words arranged one per line. The default extension for a compatible dictionary is DIC.

The format of an extended custom dictionary is unique to VisualSpeller. In addition to a word list, an extended dictionary can also contain replacement and hyphenation information. This allows it to act as a disk-based ignore/replace list. The default extension for an extended custom dictionary is VTC.

Within properties, custom dictionaries are referenced by a dictionary index number, which is a negative number from -1 to -255. The first custom dictionary opened for a context that is still open is number -1; the second custom dictionary opened that is still open is number -2, and so on. As you may note, when a dictionary is closed, the remaining open dictionaries are renumbered to fill the gap.

Standard Dictionaries

Sharing Custom Dictionaries

Alphabetization in Custom Dictionaries

Extended Custom Dictionary Format

Exclusion and Automatic Replacement Lists

Dictionary Caching

Maximum Number of Open Dictionaries

Enabling and Loading Dictionaries

Dictionary Categories

Sharing Custom Dictionaries

Normally, VisualSpeller opens a custom dictionary as shareable and writable, and leaves it open. When a context requests an update, the disk file and the in-memory list are updated simultaneously. Because of this, multiple contexts can write to the same custom dictionary. Updates are immediately visible to all contexts.

Note It is possible to open a custom dictionary that is read-only. In this case, the dictionary can be used for spellchecking, but no updates are possible. Use the **CustomIsUpdateable** property to test for this situation.

Each context maintains a separate dictionary list based on the order in which the dictionaries were opened by the context. This order determines how VisualSpeller searches dictionaries. The first dictionary in the list is the one VisualSpeller searches first. Because of this, it is important that the dictionary that is likely to contain the most words be loaded first.

Dictionaries are referenced through the dictionary array. When a dictionary is opened, it is added to the array. When it is closed, it is removed from the array. The index for a specific dictionary changes therefore as dictionaries come and go. This is easy to handle because the user normally chooses a dictionary from a list box. As long as the application reloads the list box each time it is displayed, the index returned by the list box will be correct.

Custom Dictionaries

Alphabetization in Custom Dictionaries

Extended Custom Dictionary Format

Exclusion and Automatic Replacement Lists

Alphabetization in Custom Dictionaries

To make searching a custom dictionary more efficient, VisualSpeller sorts it into a special hashing order when it places the dictionary in memory. Because of this, there is no benefit to alphabetizing the words in a custom dictionary. So words in a custom dictionary appear in the order in which they were added to the dictionary. This makes the **AddToCustom** action very fast.

Custom Dictionaries

Sharing Custom Dictionaries

Extended Custom Dictionary Format

Exclusion and Automatic Replacement Lists

Extended Custom Dictionary Format

Like a compatible custom dictionary, an extended custom dictionary is a simple text file. Its format, however, provides for replacement and hyphenation information along with correctly spelled words. An extended custom dictionary always begins with the following line:

```
    VTSpell*Extended,language
```

Language is the language code for the dictionary. It is one of the following values:

Code	Language
0	American
1	English
2	French
3	German
4	Spanish
5	Portuguese
6	Italian
7	Dutch
8	Swedish
9	Finnish
10	Norwegian
11	Latin
12	Welsh
13	Polish
14	Hungarian
15	Flemish
16	Czech
17	Icelandic
18	Esperanto
19	Catalan
20	Romanian
21	Bulgarian
22	Russian
23	Quechua
24	Turkish
25	Indonesian
26	Hebrew
27	Danish
28	Canadian

The general format for each line in an extended custom dictionary is as follows:

```
    Word [?][/|\|"Replacement/|\|"] [Hyphenation]
```

Word is the string that VisualSpeller compares to each word in the text buffer. The question mark, if present, specifies that user intervention is required. Omitting the question mark specifies that no user intervention is required. Replacement, if present, is the default replacement word for Word. The question mark and the Replacement string combine to have the following meanings:

?	***Replacement***	**Meaning**
Not Present	Not Present	Treat any occurrence of Word as a correct spelling.
Present	Not Present	Treat any occurrence of Word as a normal misspelling.
Not Present	Present	Automatically replace any occurrence of Word with Replacement without intervention from the user.
Present	Present	Treat any occurrence of Word as a misspelling with Replacement as the only suggestion.

Replacement must be bracketed by slashes (/), backslashes (\), or double quotes ("). The replacement word itself must not contain the delimiter. In the following example, the name MacDonald has the suggested replacement McDonald:

```
MacDonald ?/McDonald/
```

Hyphenation information can be added to standard dictionaries and to extended custom dictionaries. This information is not used by VisualSpeller in any way. It is provided solely for use by applications. When present, hyphenation information has the form #^#^ . . .   where # is the syllable length and the caret (^) is a preferred hyphenation point. A hyphen (-) can be used to mark less desirable hyphenation points. The sum of the syllable lengths must be less than the word length. In the following example, the word present is hyphenated present:

```
present 3^
```

Here, the word magnificent is hyphenated magnificent. Notice that two of the three hyphenation points are marked with a caret. This indicates that magnificent and magnificent are the preferred hyphenations for the word:

```
magnificent 3^3-1^
```

Hyphenation information applies to the original word, not to the replacement word. Because of this, hyphenation information is meaningless in an entry that also contains replacement information.

Custom Dictionaries

Sharing Custom Dictionaries

Alphabetization in Custom Dictionaries

Exclusion and Automatic Replacement Lists

Exclusion and Automatic Replacement Lists

You can use an extended custom dictionary to create exclusion lists and automatic replacement lists. An exclusion list is a list of words that you always want to be considered invalid even if they are listed as valid in another dictionary. If, for example, you tend to confuse the words effect and affect, you could create an extended custom dictionary in which these words are tagged VSIR_MISSPELLED. Then, you could change the **SearchOrder** so custom dictionaries are first, and make sure you open the exclusion dictionary first. When you perform a spellcheck, VisualSpeller looks for each word first in the exclusion list and asks for confirmation on any use of effect or affect.

An automatic replacement list is an alternate find and replace mechanism. If, for example, you prefer never to use contractions in a document, you could create an extended dictionary that contains only contractions and replacements tagged with VSIR_REPLACE_ALL. When you spellcheck with **AutoReplace** on, all contractions listed in the dictionary are automatically and transparently replaced. If you prefer to confirm each replacement, tag each word in the automatic replacement list with VSIR_MISSPELLED.

Custom Dictionaries

Sharing Custom Dictionaries

Alphabetization in Custom Dictionaries

Extended Custom Dictionary Format

Dictionary Caching

VisualSpeller can access a dictionary stored in memory much faster than one stored in a disk file. Because of this, custom dictionaries and ignore/replace lists are always stored in memory. Standard dictionaries, however, are often too large to be stored entirely in memory and must therefore be cached.

Caching sets aside a small group of memory blocks that are shared among standard dictionaries. When VisualSpeller needs the information in a section of the dictionary, it looks first in the cache. If the information is not there, it copies the information from the dictionary file to an unused memory block in the cache. Eventually, the cache is full. From this point forward, information from the disk replaces the oldest information in the cache.

If VisualSpeller fails to find a word in the cache, however, it searches the custom dictionaries and ignore/replace lists before seaching the rest of the standard dictionary on disk. This maximizes the opportunity for finding the word in memory.

VisualSpeller allows you to adjust the size of the cache at run time. The goal of caching is to strike a balance between performance and memory usage. Searching is fastest when a dictionary is totally in memory, but memory availability often makes this impossible. A larger cache improves performance at the expense of memory. A smaller cache conserves memory at the expense of performance.

A higher cache setting for a specific standard dictionary forces more of that dictionary into memory and diminishes its need for caching. A lower setting may force less of the dictionary into memory and increases its need for caching.

The effectiveness of the cache depends on factors such as its size, the size of the dictionary, and the words being checked. For instance, if a one block cache were used with a dictionary that contains only two blocks of data, VisualSpeller would have a 50% chance of finding a specific word in the cache. In this case, the cache would be 50% effective. With a large dictionary, such as AMERICAN.VTD, which has over 200 blocks of data, a one block cache would require a read almost every time.

Users can change the performance level by adjusting the Performance control in the Spell Options dialog box.

Standard Dictionaries

Custom Dictionaries

Dictionary Caching

Enabling and Loading Dictionaries

Dictionary Categories

Maximum Number of Open Dictionaries

The number of dictionaries that VisualSpeller can open and use are limited only by memory and available file handles. Standard dictionaries, when used by multiple contexts, share memory but not file handles. Every context must have a separate file handle for the dictionary.

Standard Dictionaries

Custom Dictionaries

Enabling and Loading Dictionaries

Making a dictionary available for spellchecking involves three processes. The dictionary must be opened, the information in the dictionary must be loaded into memory, and the dictionary must be enabled. VisualSpeller opens a dictionary in response to an **OpenStandard** or **OpenCustom** action. It closes a dictionary in response to a **CloseDictionary** action.

Loading a custom dictionary copies it entirely into memory. Loading a standard dictionary copies all or part of it into memory, depending on the setting of DictionaryPerformance. Unloading a dictionary leaves it open but removes it from memory and flushes all of its data from the cache if it is not in use by another context. Pre-loading a dictionary speeds initial access at the expense of memory. Unloading it saves memory but causes a delay when it is needed again. Use **LoadDictionary** to load an open dictionary and **UnLoadDictionary** to unload it.

Enabling a dictionary flags it as available for spellchecking. Disabling a dictionary flags it as unavailable for spellchecking. Enabling and disabling do not affect whether a dictionary resides in memory. When a spellcheck begins, however, all enabled dictionaries are immediately loaded into memory. After a spellcheck, dictionaries must be explicitly unloaded. Use **EnableDictionary** to enable or disable an open and loaded dictionary.

The idea behind enabling and disabling is to make multi-language spellchecking practical. Rather than continually opening and closing dictionaries, you can open all of the dictionaries you will need, then enable or disable them as necessary during the spellcheck. This avoids opening and closing during a spellcheck and makes the spellcheck much faster.

Standard Dictionaries

Custom Dictionaries

Dictionary Categories

Dictionary categories make it easy to perform operations on multiple dictionaries. Most properties and functions that require an index also accept dictionary categories. You can combine categories with the Or operator or by adding them together. For example, VSCAT_STANDARD by itself selects all standard dictionaries. VSCAT_STANDARD + VSCAT_ENABLED, however, selects only standard dictionaries that are enabled. Dictionary categories are as follows:

Category	Description
VS_ALL	All dictionaries (enabled or not)
VSCAT_STANDARD	All standard dictionaries (enabled or not)
VSCAT_IRLIST	All ignore/replace lists (enabled or not)
VSCAT_CUSTOM	All custom dictionaries (enabled or not)
VSCAT_ENABLED	All enabled dictionaries
VSCAT_DISABLED	All disabled dictionaries

SpellChecking Overview

Spellchecking can be as simple or as elaborate a process as you need for your application. In a straightforward situation, you can integrate a spellchecking control into your application with just a few lines of code. When you need greater control, VisualSpeller provides almost free rein as you tailor its functionality to your application.VisualSpeller checks text in blocks up to 65,535 characters long. By dividing longer blocks into 64K sections, you can check documents of almost any length.

Basic Spellcheck Procedure

VisualSpeller has properties that display a standard dialog box when a misspelling occurs. If you decide to use the standard dialog box, your main task is to provide the text to be spellchecked and later to update the original text if replacements are made. The simplest VisualSpeller implementation is as follows:

```
VSpell1.OpenStandard = "american.vtd"
VSpell1.ClearCounts = True
While (text is left to spellcheck)
    VSpell1.CheckText = (next block of text)
    While VSpell1.ResultCode < 0
        '
        '  prompt for misspelling if AutoPopUp is disabled
        '
    VSpell1.ResumeCheck = VSR_NOTHING_TO_CHECK
    Wend
    If VSpell1.ResultCode <> 0 Then
        .
        .  handle error condition
        .
    Else
        .
        .  zero result code means spellcheck is complete
        .  replace changed text, if necessary
        .
    End If
Wend
```

When a misspelling occurs, the **ResultCode** property provides important information you can use to determine what action to take. A negative **ResultCode** specifies a normal condition such as a misspelling or an ignore/replace list hit that requires attention. A positive value specifies an error condition.

Error and Status Conditions

Properties That Control Spellchecking

Properties Changed by the Spellchecking Process

Spellchecking a Single Word

Spellchecking a Text Block

Properties That Control Spellchecking

The following are properties that provide basic control over multi-word spellchecking. All of them are available at both design time and run time. None affect single word spellchecking:

Property	Affect on Spellchecking
AutoPopUp	Causes to display the dialog box automatically when a word is misspelled.
AutoReplace	Enables global replacements.
AutoSuggest	Causes to call FindSuggestions when a word is misspelled.
BreakWordCount	Causes to return control to the application after processing a specific number of words.
IgnoreFullCaps	Determines whether checks words in all caps.
IgnorePartialNumbers	Determines whether checks words that contain a mixture of letters and numbers.
MultiLine	Enables use of LineBreak property for determining line endings. Causes to increment the CurrentLine and LineOffset properties automatically.
SpellOptions =VSOPT_EXACT_MATCH	Determines whether exact case matches are required.
SpellOptions =VSOPT_IGNORE_PURE_NUMBERS	Causes to check words comprised entirely of numerals.
SpellOptions = VSOPT_RETURN_EACH_WORD	Causes to return control to the application after each word.

Properties Changed by the Spellchecking Process

Properties Changed by the Spellchecking Process

The following table shows how single-word and multi-word spellchecking affect certain properties.

Property	CheckWord Effect	CheckText Effect
MisspelledWord	None.	Clears property after each word. When a misspelling occurs, sets property to the misspelled word or the word that needs replacement.
ReplacementWord	Clears property when spellchecking begins. Sets property to replacement word if a word tagged for replacement is found in an ignore/replace list.	Clears property when spellchecking begins. Sets property to replacement word if a word tagged for replacement is found in an ignore/replace list.
CheckWord	If word is found in dictionary, sets property to actual word found. (Case of found word may differ from original.)	If word is found in dictionary, sets property to actual word found. (Case of found word may differ from original.)
Hyphenation	Sets property to hyphenation data found in dictionary if word is found and hyphenation data is available.	Sets property to hyphenation data found in dictionary if word is found and hyphenation data is available.
ResumeOffset,WordOffset	None.	Sets ResumeOffset to location of next word to spellcheck. Sets WordOffset to location of word currently being spellchecked.
IRAction	Sets property to prescribed action if word is found in an ignore/replace list.	Sets property to prescribed action if word is found in an ignore/replace list.
WhereFound,IRWhereFound	Sets property to a code that identifies the dictionary or list that contains a found word.	Sets property to a code that identifies the dictionary or list that contains a found word.
WordCount,ReplaceCount	None.	Increments property for each word spellchecked or replaced.
ResultCode	Sets property to status of spellchecking when CheckWord is complete.	Sets property to current status of spellchecking throughout a CheckText.

Because word checking has little effect on text checking properties, it is possible to check individual words behind the scenes during a text spellcheck. The only properties affected by this action are **ReplacementWord**, **CheckWord**, **IRAction**, and **ResultCode**. These properties must be saved prior to the **CheckWord** action if their contents are important.

Properties That Control Spellchecking

Events

Many activities in an application occur in response to specific actions or situations. One way to time such activities is to monitor the result code. Another is to respond to events. You can configure to generate events during any **CheckText** action. Events other than **Found** have little effect on performance. The **Found** event, however, slows spellchecking significantly. It is normally disabled. No events are generated during a CheckWord action.

Most events occur during a **CheckText** action. Because of this, you cannot use **CheckText** inside an event procedure. Doing so generates a VSR_IN_EVENT error. You can use the **CheckWord** property inside an event if you save and restore any **ReplacementWord** or **IRAction** values that are needed on exit from the event. These values are needed when the Word Not Found In Dictionary dialog box is displayed.

AfterPopup Event

AfterReplace Event

BeforeReplace Event

CheckError Event

CheckStatus Event

Complete Event

Found Event

Misspelled Event

Spellchecking a Single Word

When you spellcheck a single word, it is passed directly to . No interpretation occurs, except for certain uppercase checks. If the word has spaces or unusual characters, they are passed without modification.

A **CheckWord** action does not affect the **MisspelledWord** property, and does not automatically generate replacement suggestions. This allows you to check a single word during a CheckText action without affecting most context data. Only **ReplacementWord**, **Hyphenation**, **IRAction**, and the **CheckWord** property itself are affected by a **CheckWord** action.

Basic Spellcheck Procedure

Spellchecking a Text Block

Spellchecking a Text Block

When you check a block of text, VisualSpeller parses the block into separate words and checks them in order, one at a time. Depending on the values stored in the context, the spellcheck control either informs the application of the misspelling or informs the user directly. In this process, control may return to the application many times. It is up to the application to examine the return code and determine whether spellchecking should resume. View on of the following examples for more information:

CheckText Example

Word Not Found Example

Global Replace Example

Number of Words Processed Example

Return After Each Word Example

CheckText Example

With these settings, a **CheckText** action runs to completion. Misspelled words are displayed automatically in the   dialog box, and replacement words are automatically replaced.

Property	**Value**
AutoPopup	True
AutoReplace	True
BreakWordCount	0
SpellOptions (VSOPT_RETURN_EACH_WORD)	False

Control returns to the application in only four situations (result code is shown in parentheses):

1. All words in the text block are checked (0).
2. The user cancels the spellcheck (VSR_CHECK_CANCELED).
3. A replacement fails because the text buffer is too small (VSR_REPLACE_OVERFLOW).
4. An error such as a dictionary read failure occurs (VSR_ . . .).

In the third situation (text buffer overflow), you must determine whether to cancel the spellcheck or to continue. To continue, move the text to a larger buffer, perform the replacement, adjust **WordOffset** and **ResumeOffset** if necessary, and set **ResumeCheck**.

Spellchecking a Text Block

Word Not Found Example

In this case, control returns to the application when a word is not found in a dictionary or ignore/replace list, and it is not tagged for automatic replacement.

Property	Value
AutoPopup	False
AutoReplace	True
BreakWordCount	0
SpellOptions (VSOPT_RETURN_EACH_WORD)	False

When control returns, **ResultCode** is VSR_WORD_MISSPELLED if the word was misspelled. It is VSR_IGNORE_REPLACE if the word was found in an ignore/replace list and a VSIR_PROMPT action is needed. (VSIR_GLOBAL actions are handled automatically.)

The application must take appropriate action, usually with a **PopUpWordMisspelled**, or with a custom dialog box. After handling the word, the application normally uses a **ResumeCheck** action. If **WordOffset** and **ResumeOffset** are unchanged, spellchecking proceeds normally. Eventually, one of the situations listed in the CheckText example arises. Note that with these settings, **ResultCode** can never be VSR_CHECK_CANCELED. Only events and popup dialog boxes can generate this status.

Spellchecking a Text Block

Global Replace Example

This example is the same as the Word Not Found example, except that control also returns when a global replace flag is found in an ignore/replace list. In this case, the **ResultCode** is VSR_IGNORE_REPLACE and **IRAction** is VSIR_GLOBAL + VSIR_REPLACE.

Property	Value
AutoPopup	False
AutoReplace	False
BreakWordCount	0
SpellOptions (VSOPT_RETURN_EACH_WORD)	False

The application must handle the replacement. This can be accomplished using the **ReplaceLastWord** property because all pertinent information is stored in the context.

Spellchecking a Text Block

Number of Words Processed Example

With these settings, control returns to the application and **ResultCode** is set to VSR_BREAK after 20 words are handled by . This type of control keeps the spellchecker from maintaining exclusive control of the system.

Property	Value
AutoPopup	False
AutoReplace	False
BreakWordCount	20
SpellOptions (VSOPT_RETURN_EACH_WORD)	False

When control returns to the application, use the DoEvents to relinquish control to other Windows applications.

Spellchecking a Text Block

Return After Each Word Example

In this example, the VSOPT_RETURN_EACH_WORD option of the **SpellOptions** property is enabled. This causes   to return control to the application after every word, whether it is misspelled or not.

Property	Value
AutoPopup	False
AutoReplace	False
BreakWordCount	0
SpellOptions (VSOPT_RETURN_EACH_WORD)	True

When control returns to the application, **ResultCode** is VSR_REPLACED if the words was automatically replaced. It is VSR_FOUND if the word was found in a dictionary or ignore/replace list. If the word was found, the **WhereFound** and possibly **IRWhereFound** properties are valid. One reason to use the settings in this example, is to generate an analysis of the text block based on property settings at the time control is returned.

Spellchecking a Text Block

Language Considerations

VisualSpeller is designed to allow spellchecking in multiple languages. Note, however, that as a Windows application it assumes text passed to it conforms to the ANSI character set. Checking text that originates outside of Windows may generate false misspellings.

AboutBox Property

Description

Displays the About VisualSpeller dialog box. This dialog box contains information about your version of VisualSpeller, including your serial number. You must have your serial number to receive technical support or upgrade pricing on future product releases.

Syntax

*spellcontrol.***AboutBox**

AddSuggestion Property

Description

Adds a word you specify to the word suggestion list.

Syntax

spellcontrol.**AddSuggestion** (*matchCode*) = *text*

Part	Type	Description
matchCode	Integer	A value that VisualSpeller uses to determine the placement of text in the suggestion list. Larger values (up to 32,767) force text higher in the list. If *matchCode* is zero, VisualSpeller places the word using the same algorithm it uses for internal suggestions.
text	String	The word to add to the suggestion list.

Remarks

Use the **MaxSuggestions** property to control how many suggestions VisualSpeller generates and displays to the user.

Example

The following example adds the string assigned to the variable MySuggestion$ to the top of the suggestion list.

```
    VSpell1.AddSuggestion (32767) = MySuggestion$
```

AddToCommonIRList   Property

Description

Stores current **MisspelledWord** in the common ignore/replace list.

Syntax

spellcontrol.**AddToCommonIRList** = *code*

Part	Type	Description
code	Integer	Determines the action that should be taken if the word is found again. Following are the valid settings for code:

Constant	Description
VSIR_SPELLED_OK	Treat the word as though it is spelled correctly without asking the user for confirmation.
VSIR_REPLACE_ALL	Replace all future occurrences of the word with **ReplacementWord** without asking the user for confirmation.
VSIR_MISSPELLED	Ask user to confirm whether word is misspelled.
VSIR_PROMPT_REPLACE	Ask the user before replacing the word with **ReplacementWord**.
VSIR_HYPHENATION	Hyphenation information for the word can be added to the list. This code, if used, must be added to one of the codes listed above and the **Hyphenation** property must be set.

Example

The following example specifies that the last misspelled word should be added to the common ignore/replace list and flagged for global replacement by the **ReplacementWord**:

```
VSpell1.AddToCommonIRList = VSIR_REPLACE_ALL
```

AddToCustom Property

Description

Adds the last misspelled word to the specified custom dictionary.

Syntax

spellcontrol.**AddToCustom** (*index*) = code

Part	Type	Description
index	Integer	Indicates the dictionary in the DictionaryName array to which a word is added. Note that since this must always be a custom dictionary, the index is always a negative number. However, you can use a dictionary category, such as VSCAT_CUSTOM to instruct VisualSpeller to add the word to all custom libraries. VS_ALL, VSCAT_ENABLED and VSCAT_DISABLED are also valid variables for *index*.
code	Integer	Determines the action taken when the word is found in text. Following are the valid constants for *code*:

Constant	Description
VSIR_SPELLED_OK	Treat the word as though it is spelled correctly without asking the user for confirmation.
VSIR_REPLACE_ALL	Replace all future occurrences of the word with **ReplacementWord** without asking the user for confirmation.
VSIR_MISSPELLED	Ask user to confirm whether word is misspelled.
VSIR_PROMPT_REPLACE	Ask the user before replacing the word with **ReplacementWord**.
VSIR_HYPHENATION	Hyphenation information for the word can be added to the list. This code, if used, must be added to one of the codes listed above and the **Hyphenation** property must be set.

Example

The following example specifies that the last misspelled word should be added to the first custom dictionary as spelled.

```
VSpell1.AddToCustom (-1) =VSIR_SPELLED_OK
```

AddToStandardIRList Property

Description

Adds the last misspelled word to the ignore/replace list associated with the specified standard dictionary.

Syntax

spellcontrol.**AddToStandardIRList** (*index*) = *code*

Part	Type	Description
index	Integer	Specifies a dictionary in the DictionaryName array. This must be a positive number or a dictionary category such as VSCAT_STANDARD, VSCAT_ENABLED, VSCAT_DISABLED, and VS_ALL.
code	Integer	Determines the action that should be taken if the word is found again. Following are the valid constants for *code*:

Constant	Description
VSIR_SPELLED_OK	Treat the word as though it is spelled correctly without asking the user for confirmation.
VSIR_REPLACE_ALL	Replace all future occurrences of the word with **ReplacementWord** without asking the user for confirmation.
VSIR_MISSPELLED	Ask user to confirm whether word is misspelled.
VSIR_PROMPT_REPLACE	Ask the user before replacing the word with **ReplacementWord**.
VSIR_HYPHENATION	Hyphenation information for the word can be added to the list. This code, if used, must be added to one of the codes listed above and the **Hyphenation** property must be set.

AfterPopup Event

Description

This event occurs after a word has been handled by the Word Not Found dialog box.

Syntax

Private Sub *spellcontrol*_**AfterPopup** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	**Description**
VS_DEFAULT_HANDLING	Spellcheck continues.
VS_EVENT_HANDLED	Spellcheck continues.
VS_CANCEL_SPELLCHECK	Spellcheck halts and control returns to the application with VSR_CHECK_CANCELED the result. A **Complete** event is generated.

Remarks

This event occurs only if popup events are enabled and **AutoPopup** is True. It provides a convenient place for updating a text control when the user chooses to replace the misspelled word. On entry to the **AfterPopup** event, *EventAction* is set to VS_DEFAULT_HANDLING. To cancel the spellcheck, the event code should set *EventAction* to VS_CANCEL_SPELLCHECK.

AfterReplace Event

Description

This event occurs after an automatic global replacement of a word with an entry from an ignore/replace list or extended custom dictionary.

Syntax

Private Sub *spellcontrol*_**AfterReplace** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	**Description**
VS_DEFAULT_HANDLING	Spellcheck continues.
VS_EVENT_HANDLED	Spellcheck continues.
VS_CANCEL_SPELLCHECK	Spellcheck halts and control returns to the application with VSR_CHECK_CANCELED the result. A **Complete** event is generated.

Remarks

This event occurs only if replace events are enabled and **AutoReplace** is True. It provides a convenient place for updating a text control after an automatic replacement action. On entry to the **AfterReplace** event, *EventAction* is set to VS_DEFAULT_HANDLING, and may be changed to VS_CANCEL_SPELLCHECK to cancel the spellcheck.

AllowJoinedWords Property

Description

Determines whether hyphenated words are considered to be spelled correctly if the component words are individually spelled correctly.

Syntax

spellControl.**AllowJoinedWords** [= *boolean*]

Remarks

If this property is True, VisualSpeller considers a hyphenated word to be spelled correctly if its component words are individually spelled correctly. If it is False, hyphenated words must be explicitly found in a dictionary to be considered spelled correctly.

Example

The following example specifies that hyphenated words must be explicitly found in the dictionary to be considered correct:

```
VSpell1.AllowJoinedWords = False   'space-heater would fail
VSpell1.AllowJoinedWords = True    'space-heater would check OK
```

AutoPopup Property

Description

Determines whether a dialog box appears automatically when a misspelling occurs.

Syntax

spellcontrol.**AutoPopup** [= *boolean*]

Remarks

If this property is True, the Word Not Found In Dictionary dialog box appears automatically the first time VisualSpeller encounters a potentially misspelled word. It remains visible until the spellcheck is complete, until it is canceled, or until some other condition returns control to the application.

If it is False, the dialog box appears only when explicitly called with the **PopupWordMisspelled** property. It disappears when the user makes a decision.

When **AutoPopup** is True and popup events are enabled, VisualSpeller generates an **AfterPopup** event each time the user makes a decision on a misspelled word.

Use **EnableEventOptions** to enable popup events.

AutoReplace Property

Description

Determines whether global replace conditions in ignore/replace lists and extended custom dictionaries are handled automatically by VisualSpeller.

Syntax

spellcontrol.**AutoReplace** [= *boolean*]

Remarks

If this property is True, suggestions are generated after the dialog box appears if **AutoPopup** is enabled. During this process, "Searching..." appears in the suggestion list box. If **AutoPopup** is disabled, suggestions are generated before control returns to the application and before the **Misspelled** event occurs.

If this property if False, suggestions are not generated automatically and must be explicitly generated via **FindSuggestions** or via the suggestions buttons in the dialog box.

If **AutoReplace** is True and replace events are enabled, VisualSpeller generates **BeforeReplace** and **AfterReplace** events.

Use **EnableEventOptions** to enable replace events.

AutoSuggest Property

Description
Determines whether VisualSpeller automatically generates the suggestion list for a misspelled word.

Syntax
spellcontrol.**AutoSuggest** [= *boolean*]

Remarks
If this property is True, suggestions are generated after the dialog box appears. During this process Searching... appears in the suggestion list box.

If this property is False, suggestions are generated before control returns to the application and before the **Misspelled** event occurs.

BeforeReplace Event

Description

The **BeforeReplace** event occurs when a global automatic replacement is about to occur.

Syntax

Private Sub *spellcontrol*_**BeforeReplace** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	**Description**
VS_DEFAULT_HANDLING	Word is replaced and spellchecking continues.
VS_EVENT_HANDLED	Spellchecking continues with the next word. This is useful when the event rather than VisualSpeller handles the replacement. **Text** and **ResumeOffset** properties can be used to adjust the text being spellchecked. The replacement word is not spellchecked unless the ResumeOffset is adjusted appropriately.
VS_CANCEL_SPELLCHECK	Replacement does not occur. Spellchecking halts and control is returned to the calling procedure with VSR_CHECK_CANCELED the result.

Remarks

This event occurs in place of the **Misspelled** event for automatic replacements. It does not occur if **AutoReplace** is False or replace events are disabled (The **Misspelled** event occurs instead.) *EventAction* is preset to VS_DEFAULT_HANDLING on entry to the **BeforeReplace** procedure. You can change the setting of *EventAction* during the procedure.

Use **EnableEventOptions** to enable replace events.

BeginCheck Property

Description

Begins spellchecking current text in buffer.

Syntax

resultCode = *spellcontrol.***BeginCheck**

Part	Type	Description
resultCode	Integer	Variable that receives the returned value of the **ResultCode**. Test this property to determine if the **BeginCheck** was successful.

Remarks

Calling this property resets all offset values to zero and initiates a spellcheck from the beginning of the current buffer text. If called during a spellcheck, **BeginCheck** restarts the spellcheck from the beginning of the text.

CheckText is equivalent to "**Text** = *text*" followed by **BeginCheck**.

Example

The following example initiates a spellcheck and assigns the result code to the variable MyResult.

```
Dim MyResult As Integer

MyResult = VSpell1.BeginCheck
```

BreakWordCount Property

Description

Sets or returns the interval at which   returns control to the application during processing.

Syntax

spellcontrol.**BreakWordCount** [= *count*]

Part	**Type**	**Description**
count	Integer	The number of words to process before returning control to the application.

Remarks

This property provides for a periodic break in the spellcheck process during a long spellcheck that has few misspelled words. After checking the number of words specified by *count*, VisualSpeller returns control to the application and/or triggers a **CheckStatus** event. Typically, control is given back to Windows to process pending events. Then the spellcheck is resumed with **ResumeCheck**.

Example

The following example causes VisualSpeller to return control to the application after every 100th word processed.

```
    VSpell1.BreakWordCount = 100
```

CacheHits Property

Description

Returns the number of times data was found in the cache.

Syntax

number = *spellcontrol*.**CacheHits**

Part	Type	Description
number	Long	Variable that receives the returned number of cache hits.

Remarks

This property returns information you can use to fine tune the performance of cache.

See Also

CacheMisses property

CacheMisses Property

Description

Returns the number of times data was not found in cache. If the data is not found in the cache, a disk access is required to find the data.

Syntax

number = *spellcontrol*.**CacheMisses**

Part	**Type**	**Description**
number	Long	Variable that receives the returned number of cache misses.

Remarks

This property returns information you can use to fine tune the performance of the cache.

See Also

CacheHits property

CacheSize Property

Description

Sets or returns the size of the internal memory cache used to manage blocks of data from standard dictionaries.

Syntax

spellcontrol.**CacheSize** [**=** *size*]

Part	**Type**	**Description**
size	Integer	Size of the internal memory cache.

Remarks

The number of memory blocks assigned to the cache is shared among all loaded dictionaries and all users of the spellchecker. The user that specifies the largest cache controls its size. A value of zero specifies the default cache size. The larger the cache size, the better the overall spellchecking performance. **DictionaryBlockCount** returns the cache size required to fit the entire dictionary in memory.

CheckedWord Property

Description
Returns the last word checked in a **CheckWord** or **CheckText** operation.

Syntax
word = *spellcontrol*.**CheckedWord**

Part	Type	Description
word	String	Variable that receives the returned, last checked word.

Remarks
The word returned will be the last variant of the word spellchecked. This may be a different case than the original word, and reflects the actual word found if the word was not misspelled.

CheckError Event

Error Conditions

Description

The **CheckError** event occurs when there is an error condition during a **CheckText**, **BeginCheck**, or **ResumeCheck** action.

Syntax

Private Sub *spellcontrol*_**CheckError** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	Description
VS_DEFAULT_HANDLING	Spellcheck halts and control returns to the application.
VS_EVENT_HANDLED	Spellcheck continues.
VS_CANCEL_SPELLCHECK	Spellcheck halts and control returns to the application with VSR_CHECK_CANCELED the result. A **Complete** event is generated.

Remarks

This event occurs only if error events are enabled. On entry to the **CheckError** event, *EventAction* is set to VS_DEFAULT_HANDLING. It can be changed to one of the other settings during the process.

Use **EnableEventOptions** to enable error events.

CheckStatus Event

Status Conditions

Description

The **CheckStatus** event occurs when a status condition in a **CheckText**, **BeginCheck**, or **ResumeCheck** action is not handled by a different event.

Syntax

Private Sub *spellcontrol*_**CheckStatus** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	**Description**
VS_DEFAULT_HANDLING	Spellcheck halts and control returns to the application.
VS_EVENT_HANDLED	Spellcheck continues.
VS_CANCEL_SPELLCHECK	Spellcheck halts and control returns to the application with VSR_CHECK_CANCELED the result. A **Complete** event is generated.

Remarks

This event occurs only if status events are enabled. Click here for information about Status Conditions. On entry to the **CheckStatus** event, *EventAction* is set to VS_DEFAULT_HANDLING. It can be changed to any of the other settings during the process.

Use **EnableEventOptions** to enable status events.

CheckText Property

Description
Specifies the text to be spellchecked and evokes a spellcheck session.

Syntax
spellcontrol.**CheckText** = *text*

Part	Type	Description
text	String	Text to be spellchecked.

Remarks
CheckText is one of the basic spellcheck properties. Assigning a buffer of text to it spellchecks the text. If **MultiLine** is True, text can contain multiple lines.

When you start a spellcheck, it proceeds until VisualSpeller encounters a condition that needs attention. If control returns to the application, it must examine **ResultCode** and take appropriate action. If the spellcheck is to continue, the application must initiate a **ResumeCheck**. When a misspell condition occurs, **MisspelledWord**, **WordOffset**, and **ResumeOffset**, will reflect the misspelled word and its position in the text buffer. If MultiLine is enabled, then **CurrentLine** and **LineOffset** will be valid. If the result is a VSR_IGNORE_REPLACE condition, then **ReplacementWord** may be set if the **IRAction** includes VSIR_REPLACE.

To get the last variant of the current word that was checked, read **CheckedWord**.

Example
The following example checks the content of a text box without using events or automatically displayed dialog boxes.

```
Dim ResCode As Integer
VSpell1.EventOptions = 0
VSpell1.AutoPopup = False
VSpell1.CheckText = TextBox.Text              'start spellcheck
ResCode = VSpell1.ResultCode                   'get result
Do While ResCode < 0
    Select Case ResCode
        Case VSR_WORD_MISSPELLED, VSIR_IGNORE
                  VSpell1.PopupMisspelledWord  = 1      'display misspell dialog box
                  ResCode = VSpell1.ResultCode             'get new result
        Case VSR_BREAK
                  Do Events                            'give Windows a time slice
    End Select
    If ResCode - VSR_CHECK_CANCELLED       'see if cancelled
    Exit Loop
    End If
    ResCode = VSpell1.ResumeCheck                   'resume w/o rechecking
Loop
TextBox.Text = VSpell1.Text                                    'read out spellchecked text
```

CheckWord Property

Description

Spellchecks a single specified word and sets the value of **CheckedWord**.

Syntax

spellcontrol.**CheckWord** = *word*

Part	**Type**	**Description**
word	String	Word to be spellchecked.

Remarks

word is spellchecked as is with no parsing. All characters, including spaces, are considered. It is maintained in a buffer separate from that of the **CheckText**property. **CheckWord** invokes no action, generates no events or suggestions, performs no replacements, and does not affect the **MisspelledWord**property.

To return the last variant of word that was checked with **CheckWord** use **CheckedWord**.

ClearCommonIRList Property

Description

Clears the specified types of words from the common ignore/replace list.

Syntax

spellcontrol.**ClearCommonIRList** = *types*

Part	Type	Description
types	Integer	Determines which words are cleared from the list. It can be composed of one or more of the following four settings which can be combined with the OR operator or by addition:

Constant	Description
VSIR_GLOBAL	Remove words tagged for action without confirmation.
VSIR_PROMPT	Remove words tagged for action with confirmation.
VSIR_IGNORE	Remove words that are to be ignored.
VSIR_REPLACE	Remove words that are to be automatically replaced.

The following four choices should not be added to any other choice as this may change the selections.

Constant	Description
VSIR_SPELLED_OK	Remove words tagged as spelled correctly.
VSIR_REPLACE_ALL	Remove words tagged for global replacement.
VSIR_MISSPELLED	Remove misspelled words.
VSIR_PROMPT_REPLACE	Remove words tagged for replacement with confirmation

Remarks

Only those words that have all of the settings specified by types are removed from the list. For example, if types is VSIR_PROMPT, all entries that require confirmation are removed whether they are to be ignored or replaced. If types is VSIR_PROMPT + VSIR_IGNORE, only those entries that are to be ignored with confirmation are removed.

To remove all entries from the list, set types to zero.

ClearCounts Property

Description
Resets the **ReplaceCount** and **WordCount** properties to zero.

Syntax
spellcontrol.**ClearCounts** = 1

ClearOffsets Property

Description

Resets the **CurrentLine**, **LineOffset**, **WordOffset**, and **ResumeOffset** properties to zero.

Syntax

spellcontrol.**ClearOffsets** = 1

ClearStandardIRList Property

Description

Clears the specified types of words from the ignore/replace list associated with the specified standard dictionary.

Syntax

spellcontrol.**ClearStandardIRList** (*index*) = *types*

Part	**Type**	Description
index	Integer	Identifies a dictionary in the DictionaryName array. Positive numbers refer to standard dictionaries. The dictionary categories VS_ALL, VSCAT_STANDARD, VSCAT_ENABLED and VSCAT_DISABLED may be used to specify multiple lists.
types	Integer	Determines which words are cleared from the list. It can be composed of one or more of the following four settings which can be combined with the OR operator or by addition:

Constant	Description
VSIR_GLOBAL	Remove words tagged for action without confirmation.
VSIR_PROMPT	Remove words tagged for action with confirmation.
VSIR_IGNORE	Remove words that are to be ignored.
VSIR_REPLACE	Remove words that are to be

automatically replaced.

The following four choices should not be added to any other choice as this may change the selections.

Constant	Description
VSIR_SPELLED_OK	Remove words tagged as spelled correctly.
VSIR_REPLACE_ALL	Remove words tagged for global replacement.
VSIR_MISSPELLED	Remove misspelled words.
VSIR_PROMPT_REPLACE	Remove words tagged for replacement with confirmation

Remarks

Only those words that have all of the settings specified by types are removed from the list. For example, if types is VSIR_PROMPT, all entries that require confirmation are removed whether they are to be ignored or replaced. If types is VSIR_PROMPT + VSIR_IGNORE, only those entries that are to be ignored with confirmation are removed.

To remove all entries from the list, set types to zero.

ClearSuggestions Property

Description
Clears the word suggestion list.

Syntax
spellcontrol.**ClearSuggestions** = 1

ClickedButton Property

Description

Returns information about the button that was clicked during a **ClickIn** or **ClickOut** event.

Syntax

button = *spellcontrol*.**ClickedButton**

Part	Type	Description
button	Long	Variable that receives the returned information. Following are the valid constants returned to this variable:

Constant	Dialog Box	Button
VSCLICK_WORD_MISSPELLED_HELP	Word Not Found In Dictionary	Help
VSCLICK_OPTIONS	Word Not Found In Dictionary	Options
VSCLICK_PROMPT_REPLACE	Word Not Found In Dictionary	Prompt Replace
VSCLICK_ADD_TO_CUSTOM	Word Not Found In Dictionary	Add To Custom
VSCLICK_REPLACE_ALL	Word Not Found In Dictionary	Replace All
VSCLICK_REPLACE	Word Not Found In Dictionary	Replace
VSCLICK_IGNORE_ALL	Word Not Found In Dictionary	Ignore All
VSCLICK_IGNORE	Word Not Found In Dictionary	Ignore
VSCLICK_SUGGEST_NOT_FOUND	Word Not Found In Dictionary	Suggestions Not Found
VSCLICK_SUGGEST_REPLACE_WITH	Word Not Found In Dictionary	Suggestions Replace With
VSCLICK_CANCEL_SPELLCHECK	Word Not Found In Dictionary	Cancel Spellcheck
VSCLICK_THESAURUS	Not Available in this release	
VSCLICK_OPTIONS_HELP	Spell Options	Help
VSCLICK_OPTIONS_OK	Spell Options	OK
VSCLICK_OPTIONS_CANCEL	Spell Options	Cancel
VSCLICK_OPEN_CUSTOM	Spell Options	Open Custom
VSCLICK_CLOSE_CUSTOM	Spell Options	Close Custom
VSCLICK_OPEN_STANDARD	Spell Options	Open Standard
VSCLICK_CLOSE_STANDARD	Spell Options	Close Standard
VSCLICK_THESAURUS_OK	Not Available in this release	
VSCLICK_THESAURUS_SEARCH	Not Available in this release	
VSCLICK_THESAURUS_CANCEL	Not Available in this release	
VSCLICK_THESAURUS_HELP	Not Available in this release	

Remarks

These constants reflect different bit values.

ClickIn Event

Description

This event is triggered when a button is pressed on either the Word Not Found in Dictionary or Spell Options dialog box. When the event is entered, you can read the **ClickedButton** property to determine which button was clicked.

Syntax

Private Sub *spellcontrol*_**ClickIn** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	**Description**
VS_DEFAULT_HANDLING	Normal function of the button is executed upon return from this event and a **ClickOut** event is triggered.
VS_EVENT_HANDLED	Normal function of the button is bypassed and the **ClickOut** event is not triggered.
VS_CANCEL_SPELLCHECK	Spellcheck halts and control returns to the application with VSR_CHECK_CANCELED the result. No **ClickOut** event is generated.

Remarks

The **ClickIn** event is triggered before any action is taken by the spellchecker. In some cases you can read the **ClickInfo** or **ClickInfoText** properties to provide more information about the button.

ClickInfo Property

Description

Returns data related to a ClickIn or **ClickOut** event. This is only valid inside the event and only returns meaningful information for certain buttons.

Syntax

code = *spellcontrol*.**ClickInfo**

Part	Type	Description
code	Long	Variable that receives the returned data. Following are the valid constants returned to this variable:

Constant	Description
VSCLICK_ADD_TO_CUSTOM	Returns a negative number indicating the index of the custom dictionary selected after clicking the Add To Custom button on the Word Not Found in Dictionary dialog box.
VSCLICK_OPEN_CUSTOM	Returns zero or a negative number indicating the index of the custom dictionary to be opened after clicking on the Open Custom button of the Spell Options dialog box. This return code is only valid in a **ClickOut** event.
VSCLICK_CLOSE_CUSTOM	Returns a negative number indicating the index of the custom dictionary to be close after clicking on the Close Custom button of the Spell Options dialog box. This code is only valid in a **ClickIn** event.
VSCLICK_OPEN_STANDARD	Returns zero or a positive number indicating the index of the standard dictionary to be opened after clicking on the Open Standard button of the Spell Options dialog box. This code is only valid in a **ClickOut** event.
VSCLICK_CLOSE_STANDARD	Returns a positive number indicating the index of the standard dictionary to be closed after clicking on the Close Standard button of the Spell Options dialog box. This code is only valid in a **ClickIn** event.

ClickInfoText Property

Description

Returns data related to a **ClickIn** or **ClickOut** event. This property is only valid inside the event and only returns meaningful information for certain buttons.

Syntax

text = *spellcontrol*.**ClickInfoText**

Part	**Type**	**Description**
text	Long	Variable that receives the returned data. Following are the valid constants returned to this variable.

Constant	**Description**
VSCLICK_PROMPT_REPLACE	Returns the string currently in the Replace With edit box.
VSCLICK_REPLACE_ALL	Returns the string currently in the Replace With edit box.
VSCLICK_REPLACE	Returns the string currently in the Replace With edit box.
VSCLICK_SUGGEST_REPLACE_WITH	Returns the string currently in the Replace With edit box.

ClickOut Event

Description

This event is triggered when a button is pressed on either the Word Not Found in Dictionary or Spell Options dialog box. When the event is entered, you can read the **ClickedButton** property to determine which button was clicked.

Syntax

Private Sub *spellcontrol*_**ClickOut** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	Description
VS_DEFAULT_HANDLING	Continues as would normally following the button press.
VS_EVENT_HANDLED	Has no effect since the button function has already been performed. Continues same as VS_DEFAULT_HANDLING.
VS_CANCEL_SPELLCHECK	Spellcheck halts and control returns to the application with VSR_CHECK_CANCELED the result.

Remarks

The **ClickOut** event is triggered after the **ClickIn** event and after the speller has performed whatever functionality the button is supposed to do. If the **ClickIn** event overrides the normal button function by setting the *EventAction* parameter to either VS_EVENT_HANDLED or VS_CANCEL_SPELLCHECK, then the **ClickOut** event is not fired. In some cases the **ClickInfo** or **ClickInfoText** properties may contain more information about the clicked button.

CloseDictionary Property

Description

Closes the specified dictionary.

Syntax

spellcontrol.**CloseDictionary** (*index*)

Part	Type	Description
index	Integer	Specifies a dictionary in the DictionaryName array. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries. You can also use any valid dictionary category to specify multiple dictionaries.

Remarks

When the dictionary is closed, the index numbers of the remaining dictionaries may change to fill in gap left by the closed dictionary (s). For instance, if three standard dictionaries are open, and the second one is closed, the dictionary which previously used index 3 now uses index 2.

Example

The following example closes the most recently opened custom dictionary:

```
VSpell1.CloseDictionary (-CustomCount)
```

CommonIRListIsEnabled Property

Description

Returns whether the common ignore/replace list is enabled.

Syntax

boolean = *spellcontrol*.**CommonIRListIsEnabled**

Remarks

If this property is True, the common ignore/replace list is enabled. If this property is False, the common ignore/replace list is disabled.

To enable or disable the common ignore/replace list, use **EnableCommonIRList**.

Complete Event

Description
This event occurs when a spellcheck action is complete or when it is canceled by the user.

Syntax
Private Sub *spellcontrol*_**Complete** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	**Description**
VS_DEFAULT_HANDLING	VisualSpeller returns control to the calling procedure with VSVEOPT_COMPLETE the result.
VS_EVENT_HANDLED	If the procedure specifies new text using the **Text** property or changes the **ResumeOffset** property, spellchecking continues. Otherwise, control returns to the calling procedure and a VSR_NOTHING_TO_CHECK error occurs.
VS_CANCEL_SPELLCHECK	Spellchecking halts and control is returned to the calling procedure with VSR_CHECK_CANCELED the result.

Remarks
Complete events occur only when the complete events are enabled. When a spellcheck ends normally, presets *EventAction* to VS_DEFAULT_HANDLING on entry to the **Complete** procedure. When the user cancels a spellcheck before completion,   presets *EventAction* to VS_CANCEL_SPELLCHECK. You can change *EventAction* by setting it to one of the other values during the **Complete** procedure.

CreateCustom Property

Description

Creates an empty compatible custom dictionary.

Syntax

spellcontrol.**CreateCustom** = *name*

Part	**Type**	**Description**
name	String	Name of the dictionary to create.

Remarks

File must not already exist, or error VSR_FILE_EXISTS is generated. If successful, -**CustomCount** will return the index number for the new dictionary.

CreateCustomExtended Property

Description
Creates an extended custom dictionary.

Syntax
spellcontrol.**CreateCustomExtended** (*languagecode*) = *name*

Part	**Type**	**Description**
languagecode	Integer	One of the predefined VisualSpeller language codes. See **DictionaryLanguage** for a list of codes. This code is for informational purposes only, and does not affect the spellcheck or dictionary contents in any way.
name	String	Name of the dictionary to create.

Remarks
This property creates an empty extended custom dictionary. The file must not already exist, or error VSR_FILE_EXISTS is generated. If successful, -**CustomCount** gives the index number for the dictionary.

CurrentLine Property

Description

Sets or returns the line number reference for the current word.

Syntax

spellcontrol.**CurrentLine** [= *lineNumber*]

Part	Type	Description
lineNumber	Long	Identifies the line within a text block.

Remarks

This value is for the application's use and is not used by the spellchecker. If **MultiLine** is enabled, then the spellchecker increments this zero-based value on each new line (as determined by **LineBreak**) during a spellcheck. For more information, refer to **LineOffset**.

Example

The following example displays the position of the misspelled word.

```
MsgBox "Word misspelled at line "& Str(VSpell1.CurrentLine + 1)& " character " &
        Str(VSpell1.LineOffset + 1)
```

CustomCount Property

Description

Returns the number of open custom dictionaries.

Syntax

count = *spellcontrol*.**CustomCount**

Part	Type	Description
count	Integer	Variable that receives the returned number of open dictionaries.

Remarks

Since this returns the number of open custom dictionaries, -**CustomCount** (negative CustomCount) is always the index number of the most recently opened custom dictionary.

Example

The following example fills a list box with custom dictionary names:

```
Max = VSpell1.CustomCount
For X = 1 to Max
    ListBox.AddItem = VSpell1.DictionaryName (-X)
End X
```

CustomDictionary Property

Description

Sets the initial custom dictionary. This is a design-time only property.

Syntax

spellcontrol.**CustomDictionary** = *name*

Part	Type	Description
name	String	Name of the custom dictionary.

Remarks

This property is used internally by VisualSpellers property pages. At runtime, use **OpenCustom** to accomplish the same result.

CustomIsExtended Property

Description

Returns whether a custom dictionary is compatible or extended.

Syntax

status = *spellcontrol*.**CustomIsExtended** (*index*)

Part	Type	Description
index	Integer	Specifies the DictionaryName index number of the custom dictionary for which to return the type. Negative numbers indicate custom dictionaries.
status	Boolean	Variable that holds the returned information. If status is True, the dictionary is an extended custom dictionary. An extended custom dictionary can include ignore/replace and hyphenation information. The normal extension for an extended dictionary is VTC. If status is False, the dictionary is a compatible custom dictionary. These dictionaries are compatible with the dictionaries used by Microsoft Word. It is a simple text file that contains a list of correctly spelled works, arranged one per line. The normal extension for a compatible dictionary is DIC.

CustomIsReadOnly Property

Description

Returns whether a custom dictionary was opened read-only.

Syntax

status = *spellcontrol*.**CustomIsReadOnly** (*index*)

Part	Type	Description
index	Integer	Specifies the DictionaryName index number of the custom dictionary for which to return the open flag. Negative numbers indicate custom dictionaries.
status	Boolean	Variable that receives the returned information. If status is True, the dictionary is read-only and cannot be modified. If status is False, the dictionary is not read-only. If the dictionary is also updateable, it can be modified.

CustomIsUpdateable Property

Description

Returns whether a custom dictionary can be updated.

Syntax

status = *spellcontrol*.**CustomIsUpdateable** (*index*)

Part	Type	Description
index	Integer	Specifies the DictionaryName index number of the custom dictionary for which to return the updateable flag. Negative numbers indicate custom dictionaries.
status	Boolean	Variable that receives the returned information. If *status* is True, the dictionary is updateable. If *status* is False, the dictionary is not updateable.

Remarks

To enable or disable a custom dictionary, use **EnableCustomUpdate**.

DialogBGColor Property

Description

Sets or returns the background color of the Word Not Found in Dictionary and Spell Options dialog boxes.

Syntax

spellcontrol.**DialogBGColor** [= *color*]

Part	**Type**	**Description**
color	OLE_COLOR	The settings for *color are:* ● Normal RGB colors specified by using the Color palette or by using the RGB or QBColor functions in code. ● System default colors specified by system color constants listed in the Visual Basic object library in the Object Browser. The Windows operating environment substitutes the user's choices as specified in the Control Panel settings.

DialogElements Property

Description

Returns bits indicating enabled dialog elements. Dialog elements are the controls in Word Not Found in Dictionary and Spell Options dialog boxes. When disabled, a control is hidden leaving a blank spot in its location.

Syntax

elements = *spellcontrol*.**DialogElements**

Part	Type	Description
elements	Long	Variable that receives the enabled information. Valid constants for this variable can be tested by ANDing with the following values:
		Elements in Word Not Found in Dictionary Dialog
		VSD_WORD_MISSPELLED_HELP
		VSD_OPTIONS
		VSD_PROMPT_REPLACE
		VSD_ADD_TO_CUSTOM
		VSD_REPLACE_ALL
		VSD_REPLACE
		VSD_IGNORE_ALL
		VSD_IGNORE
		VSD_SUGGEST_NOT_FOUND
		VSD_SUGGEST_REPLACE_WITH
		VSD_CANCEL_SPELLCHECK
		VSD_SUGGESTIONS_LIST
		VSD_REPLACE_BOX
		VSD_CUSTOM_LIST
		VSD_THESAURUS
		Elements in Spell Options Dialog Box
		VSD_OPTIONS_HELP
		VSD_CUSTOMS
		VSD_STANDARDS
		VSD_PERFORMANCE
		VSD_SUGGESTION_LIMIT
		VSD_AUTO_SUGGESTIONS
		VSD_EXACT_MATCH
		VSD_IGNORE_FULL_CAPS
		VSD_IGNORE_PARTIAL
		VSD_IGNORE_PURE
		VSD_ALLOW_JOINED

VSD_RECHECK

Remarks

Dialog elements are enabled or disabled with the **EnableDialogElements** and **DisableDialogElements** properties. All except the THESAURUS button in the Word Not Found in Dictionary dialog box are enabled by default.

DialogHeight Property

Description

Returns the height of the Word Not Found in Dictionary dialog box.

Syntax

height = *spellcontrol*.**DialogHeight**

Part	Type	Description
height	Long	Variable that receives the current height of the dialog box expressed in pixels.

DialogLeft Property

Description

Sets or returns the initial location of the left edge of the Word Not Found in Dictionary dialog box.

Syntax

spellcontrol.**DialogLeft** = *position*

Part	**Type**	**Description**
position	Long	Initial location expressed in pixels. When position is zero, the dialog box is centered horizontally on the screen.

DialogLeftActual Property

Description
Returns the actual location of the left edge of the Word Not Found in Dictionary dialog box.

Syntax
position = *spellcontrol*.**DialogLeftActual**

Part	**Type**	**Description**
position	Long	Variable that receives the current location of the left edge of the dialog box expressed in pixels.

Remarks
When you move the dialog box using either the mouse or the **DialogLeft** property, the change is automatically reflected in this property.

DialogTop Property

Description

Sets or returns the initial location of the top edge of the Word Not Found in Dictionary dialog box.

Syntax

spellcontrol.**DialogTop** = *position*

Part	**Type**	**Description**
position	Long	Initial location expressed in pixels. When the return value is zero, the dialog box is centered vertically on the screen.

DialogTopActual Property

Description

Returns the actual location of the top edge of the Word Not Found in Dictionary dialog box.

Syntax

position = *spellcontrol*.**DialogTopActual**

Part	**Type**	**Description**
position	Long	Variable that receives the current location of the top edge of the dialog box expressed in pixels.

Remarks

This property returns the current location of the top edge of the dialog box in pixels. When you move the dialog box using either the mouse or the **DialogTop** property, the change is automatically reflected in this property.

DialogWidth Property

Description

Returns the width of the Word Not Found in Dictionary dialog box.

Syntax

width = *spellcontrol*.**DialogWidth**

Part	**Type**	**Description**
width	Long	Variable that receives the current width of the dialog box expressed in pixels.

DictionaryBlockCount Property

Description

Returns the number of cache blocks within a standard dictionary.

Syntax

blockCount = *spellcontrol*.**DictionaryBlockCount** (*index*)

Part	**Type**	**Description**
blockCount	Integer	Variable that receives the number of blocks in the specified dictionary.
index	Integer	Specifies the standard dictionary for which to return the block count.

Remarks

This data is for informational purposes only can be used to help determine optimal cache sizing. It is generally recommended that the cache be sized to at least 10% of the number of blocks contained in the dictionaries in use for reasonable performance. Returns 0 for custom dictionaries.

DictionaryBlockSize Property

Description

Returns the size of a specific standard dictionarys cache block.

Syntax

blockSize = *spellcontrol*.**DictionaryBlockSize** (*index*)

Part	**Type**	**Description**
index	Integer	Specifies the standard dictionary for which to return the block size.
blockSize	Integer	Variable that receives the size in bytes.

Remarks

This data is for informational purposes only and can be used to help determine optimal cache sizing. The dictionary cache block size multiplied by the block count approximates the amount of memory needed to hold the entire dictionary.

Returns 0 for custom dictionaries.

DictionaryCommonCount Property

Description
Returns the number of common words in a specific standard dictionary.

Syntax
count = *spellcontrol*.**DictionaryCommonCount** (*index*)

Part	**Type**	**Description**
count	Integer	Variable that receives the number of common words.
index	Integer	Specifies the standard dictionary for which to return the common word count.

Remarks
The common word list is an optional, memory-resident, word list of frequently used words, and is enabled if **DictionaryPerformance** is 0 or greater.

Returns 0 for custom dictionaries.

DictionaryCompatibilityBits Property

Description

Returns the compatibility level for a standard dictionary.

Syntax

level = *spellcontrol*.**DictionaryCompatibilityBits** (*index*)

Part	Type	Description
level	Integer	Variable that receives the returned compatibility level.
index	Integer	Specifies the standard dictionary for which to return the compatibility level.

Remarks

The only bit used is 1, which indicates a valid standard dictionary. Returns 0 for custom dictionaries.

DictionaryCopyright Property

Description
Returns copyright information for a specific standard dictionary.

Syntax
string = *spellcontrol*.**DictionaryCopyright** (*index*)

Part	**Type**	**Description**
string	Integer	Variable that receives the returned copyright string.
index	Integer	Specifies the standard dictionary for which to return the copyright information.

Remarks
Returns an empty string for custom dictionaries.

DictionaryCopyrightDerived Property

Description

Returns the original copyright information for a specific standard dictionary.

Syntax

string = *spellcontrol*.**DictionaryCopyrightDerived** (*index*)

Part	Type	Description
string	String	Variable that receives the returned copyright string.
index	Integer	Specifies the standard dictionary for which to return the copyright information.

Remarks

This string will be empty unless the standard dictionary being queried was created by merging new words into a dictionary licensed by Visual Components.

DictionaryFlags Property

Description

Returns a set of bits which indicates information about the standard dictionary.

Syntax

flags = *spellcontrol*.**DictionaryFlags** (*index*)

Part	**Type**	**Description**	
flags	Integer	Variable that returns the following bits:	
		Bit	**Information indicated**
		Bit 1 (0x1)	Dictionary created by Visual Components
		Bit 2 (0x2)	Dictionary derived from dictionary created by Visual Components
		Bit 3 (0x4)	Dictionary is exportable (can be dumped to text in the Dictionary Maker)
		Bit 4 (0x8)	Dictionary contains a common word list
		Bit 5 (0x10)	Dictionary is modifiable by the Dictionary Maker
index	Integer	Specifies the dictionary for which to return the flags.	

DictionaryIsEnabled Property

Description

Determines whether a specific dictionary is enabled for spellchecking.

Syntax

status = *spellcontrol*.**DictionaryIsEnabled** (*index*)

Part	**Type**	**Description**
index	Integer	Specifies a dictionary in the **DictionaryName** array. Positive numbers indicate standard dictionaries and negative numbers indicate custom dictionaries.
status	Boolean	Variable that receives the dictionary status. If *status* is True, the dictionary is enabled for spellchecking. If *status* is False, the dictionary is not enabled and therefore not available for spellchecking.

Remarks

Use **EnableDictionary** to enable or disable a dictionary.

DictionaryIsLoaded Property

Description

Determines whether a specific dictionary is loaded.

Syntax

status = *spellcontrol.***DictionaryIsLoaded** (*index*)

Part	**Type**	**Description**
index	Integer	Specifies a dictionary in the **DictionaryName** array. Positive numbers indicate standard dictionaries and negative numbers indicate custom dictionaries.
status	Boolean	Variable that receives the dictionary load status. If *status* is True, the dictionary is loaded. If *status* is False, the dictionary is not loaded.

Remarks

In order to use a dictionary for spellchecking, it must be opened, loaded into memory, and enabled.

You may wish to unload a dictionary and conserve memory when a spellcheck is not in progress. Use **LoadDictionary** to load a dictionary and **UnloadDictionary** to unload the dictionary.

DictionaryLanguage Property

Description

Returns a code that indicates the language of a specified dictionary.

Syntax

language = *spellcontrol.***DictionaryLanguage** (*index*)

Part	**Type**	**Description**	
index	Integer	Specifies the dictionary in the DictionaryName array for which to return the language. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries.	
language	Integer	Variable that receives one of the following language codes:	
		VSLANG_AMERICAN	VSLANG_FLEMISH
		VSLANG_ENGLISH	VSLANG_CZECH
		VSLANG_FRENCH	VSLANG_ICELANDIC
		VSLANG_GERMAN	VSLANG_ESPERANTO
		VSLANG_SPANISH	VSLANG_CATALAN
		VSLANG_PORTUGUESE	VSLANG_ROMANIAN
		VSLANG_ITALIAN	VSLANG_BULGARIAN
		VSLANG_DUTCH	VSLANG_RUSSIAN
		VSLANG_SWEDISH	VSLANG_QUECHUA
		VSLANG_FINNISH	VSLANG_TURKISH
		VSLANG_NORWEGIAN	VSLANG_INDONESIAN
		VSLANG_LATIN	VSLANG_HEBREW
		VSLANG_WELSH	VSLANG_DANISH
		VSLANG_POLISH	VSLANG_CANADIAN
		VSLANG_HUNGARIAN	

DictionaryLoadCount Property

Description

Returns the number of different speller contexts which currently have the specified dictionary loaded.

Syntax

count = *spellcontrol.*__DictionaryLoadCount__ (*index*)

Part	Type	Description
count	Integer	Variable that receives the number of speller contexts.
index	Integer	Specifies the dictionary in the DictionaryName array for which to return the language. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries.

Remarks

Valid for both standard and custom dictionaries.

DictionaryMakerVersion Property

Description

Returns a number which represents the version of the Dictionary Maker used to build the specified standard dictionary.

Syntax

version = *spellcontrol*.**DictionaryMakerVersion** (*index*)

Part	Type	Description
index	Integer	Specifies the standard dictionary for which to return the version number of the Dictionary Maker (VTDMAKER.EXE).
version	Integer	Variable that receives the version. Currently 1 for all standard dictionaries. Returns 0 for custom dictionaries

DictionaryName Property

Description

Returns the name of a specified dictionary.

Syntax

name = *spellcontrol.***DictName** (*index*)

Part	**Type**	**Description**
index	Integer	The dictionary in the DictionaryName array for which to return the name. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries.
name	String	Variable that receives the name of the specified dictionary. Other path information is not included.

DictionaryNameFull Property

Description

Returns the full name for an open dictionary, complete with drive and directory names.

Syntax

name = *spellcontrol.***DictionaryNameFull** (*index*)

Part	**Type**	**Description**
index	Integer	Specifies the dictionary for which to return the name. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries.
name	String	Variable that receives the returned file name for the specified dictionary, complete with drive and directory path.

DictionaryOpenCount Property

Description

Returns the number of speller contexts which currently have this dictionary open.

Syntax

count = *spellcontrol*.**DictionaryOpenCount** (*index*)

Part	Type	Description
index	Integer	The dictionary in the DictionaryName array for which to return the name. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries.
count	Integer	Variable that receives the returned number of speller contexts.

Remarks

Valid for both standard and custom dictionaries.

DictionaryPerformance Property

Description

Sets or returns the performance level of spellchecking requested by the current context for a specific standard dictionary.

Syntax

spellcontrol.**DictionaryPerformance** (*index*) [= *performance*]

Part	**Type**	**Description**
index	Integer	Specifies the standard dictionary for which to return the performance information. Setting Index to the dictionary category VS_ALL sets the performance level for all dictionaries at once.
performance	Integer	Specifies the performance level. This level can be a value between -7 and 7. Lower values degrade performance but conserve memory. Higher values increase performance but require more memory. Actual performance is also affected by the dictionary itself and by system memory. If memory is low, for example, a value of 7 may force a dictionary into virtual memory and result in poorer performance.

Remarks

The performance level indicated may be different than the actual performance level if more than one context has this dictionary open. Returns 0 for custom dictionaries.

DictionaryPerformanceActual Property

Description

Returns the actual performance level for a specific standard dictionary.

Syntax

level = *spellcontrol.***DictionaryPerformanceActual** (*Index*)

Part	**Type**	**Description**
index	Integer	Specifies the standard dictionary for which to return the performance information.
level	Integer	Variable that receives the returned performance value. This can be a value between -7 and 7. Lower values degrade performance but conserve memory. Higher values increase performance but require more memory.

Remarks

This property returns the highest performance level used when multiple speller contexts specify different performance levels for the same dictionary.

Returns 0 for custom dictionaries.

DictionaryStatus Property

Description

Returns status information for the specified dictionary.

Syntax

status = *spellcontrol*.**DictionaryStatus** (*index*)

Part	Type	Description
index	Integer	Specifies the dictionary for which to return the status. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries.
status	Integer	Variable that receives the returned status information. This information is a set of bits that indicate the status of the dictionary. It can consist of one or more of the following constants combined with the Or operator or by addition:

Constant	Description
VSDSTAT_ENABLED	Dictionary is enabled.
VSDSTAT_LOADED	Dictionary is loaded.
VSDSTAT_IRLIST_ENABLED	Dictionarys ignore/replace list is enabled.
VSDSTAT_UPDATEABLE	Updating is enabled
VSDSTAT_READONLY	Dictionary is read-only
VSDSTAT_EXTENDED	Extended custom dictionary
VSDSTAT_HYPHENATION	Dictionary contains hyphenation information
VSDSTAT_PHONETICS	Dictionary contains phonetic search information.
VSDSTAT_COMMONLIST	Dictionary includes common word list

Example

The following example assigns to the variable Custom1Status, a value that indicates the status of the first custom dictionary and then uses it to test whether the dictionary is read-only:

```
Custom1Status = VSpell1.DictionaryStatus (-1)
If Custom1Status and VSDSTAT_READONLY Then
    MsgBox "Dictionary is read-only"
End If
```

DictionarySymbolSetSize Property

Description

Returns the size of a standard dictionarys symbol set.

Syntax

size = *spellcontrol*.**DictionarySymbolSetSize** (*index*)

Part	**Type**	**Description**
index	Integer	Specifies the dictionary for which to return the symbol set information.
size	Integer	Variable that receives the returned size of the symbol set in bytes.

Remarks

The symbol set size is the number of unique letters contained in the entire dictionary. Returns 0 for custom dictionaries.

DictionaryWordCount Property

Description

Returns the number of words in a specific dictionary.

Syntax

count = *spellcontrol*.**DictionaryWordCount** (*index*)

Part	**Type**	**Description**
index	Integer	Specifies the dictionary for which to return the status. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries.
count	Long	Variable that receives the number of words in the specified dictionary.

Remarks

Valid for both standard and custom dictionaries.

DisableDialogElements Property

Description

Disables one or more dialog elements in the Word Not Found in Dictionary dialog and the Spell Options dialog.

Syntax

spellcontrol.**DisableDialogElements** = *elements*

Part	Type	Description
elements	Long	Dialog box elements to disable. One or more of the following elements combined by addition:
		Elements in Word Not Found in Dictionary Dialog
		VSD_WORD_MISSPELLED_HELP
		VSD_OPTIONS
		VSD_PROMPT_REPLACE
		VSD_ADD_TO_CUSTOM
		VSD_REPLACE_ALL
		VSD_REPLACE
		VSD_IGNORE_ALL
		VSD_IGNORE
		VSD_SUGGEST_NOT_FOUND
		VSD_SUGGEST_REPLACE_WITH
		VSD_CANCEL_SPELLCHECK
		VSD_SUGGESTIONS_LIST
		VSD_REPLACE_BOX
		VSD_CUSTOM_LIST
		VSD_THESAURUS
		Elements in Spell Options Dialog Box
		VSD_OPTIONS_HELP
		VSD_CUSTOMS
		VSD_STANDARDS
		VSD_PERFORMANCE
		VSD_SUGGESTION_LIMIT
		VSD_AUTO_SUGGESTIONS
		VSD_EXACT_MATCH
		VSD_IGNORE_FULL_CAPS
		VSD_IGNORE_PARTIAL
		VSD_IGNORE_PURE
		VSD_ALLOW_JOINED
		VSD_RECHECK

Remarks

Any elements not referenced are left unchanged, except that setting to 0 or VS_ALL disables all elements.

DisableEventOptions Property

Description

Disables spellcheck events.

Syntax

spellcontrol.**DisableEventOptions** = *events*

Part	Type	Description
events	Integer	Determines which events are disabled. This can be composed of multiple settings combined with the OR operator or by addition. Events can assume the following constants:

Constant	Description
VSEVOPT_COMPLETE	Disables the **Complete** event.
VSEVOPT_FOUND	Disables the **Found** event.
VSEVOPT_MISSPELLED	Disables the **Misspelled** event.
VSEVOPT_BEFORE_REPLACE	Disables the **BeforeReplace** event.
VSEVOPT_POPUP	Disables the Popup event.
VSEVOPT_CHECK_STATUS	Disables the **CheckStatus** event.
VSEVOPT_CHECK_ERROR	Disables the **CheckError** event.
VSEVOPT_AFTER_REPLACE	Disables the **AfterReplace** event.
VSEVOPT_CLICK_IN	Disables the **ClickIn** event.
VSEVOPT_CLICK_OUT	Disables the **ClickOut** event.
VSEVOPT_DEFAULTS	Disables all events except the **Found** event.

Remarks

The state of events not specified are left unchanged. Setting to 0 or VS_ALL will disable all events.

Use **EnableEventOptions** to enable events and **EventOptions** to return the enabled status of events.

DisableSpellOptions Property

Description

Disables options defining how a spellcheck is performed.

Syntax

spellcontrol.**DisableSpellOptions** = *options*

Part	**Type**	**Description**
options	Integer	Determines which spell options are disabled. This can be composed of multiple settings combined with the OR operator or by addition. options can assume the following constants:

Constant	**Description**
VSOPT_AUTO_REPLACE	Sets the **AutoReplace** property to False. This prevents VisualSpeller from automatically generating a suggestion list for a misspelled word.
VSOPT_AUTO_SUGGEST	Sets the **AutoSuggest** property to False. This prevents a dialog box from automatically appearing when a misspelled word is encountered.
VSOPT_AUTO_POPUP	Sets the **AutoPopUp** property to False. This prevents a dialog box from automatically appearing when a misspelled word is encountered.
VSOPT_IGNORE_FULL_CAPS	Sets the **IgnoreFullCaps** property to False. This causes VisualSpeller to look for fully capped words in the dictionary(s).
VSOPT_IGNORE_PARTIAL_NUMBERS	Sets the **IgnorePartialNumbers** property to False. This causes VisualSpeller to question words that contain a mixture of letters and numbers during a spellcheck.
VSOPT_MULTILINE	Sets the **MultiLine** property to False. This prevents VisualSpeller from keeping track of line breaks during a spellcheck.
VSOPT_IGNORE_PURE_NUMBERS	Causes VisualSpeller to question all-numeral words.
VSOPT_ALLOW_JOINED_WORDS	Sets the **AllowJoinedWords** property to False. This prevents VisualSpeller from considering that hyphenated words are spelled correctly if the component words are individually spelled correctly.

VSOPT_EXACT_MATCH	Prevents VisualSpeller from requiring an exact case match.
VSOPT_RETURN_EACH_WORD	Prevents from returning after each word when spellchecking a block of text with the **CheckText** property.
VSOPT_REPLACE_RECHECK	Sets the **ReplaceRecheck** property to False. This prevents Visual Speller from checking words typed by the user in the Word Not Found in Dictionary dialog box.

Remarks

Any options not referenced are left unchanged. To disable all options, set options to zero or VS_ALL.

DisableSuggestOptions Property

Description

Disables options defining how word suggestions are generated.

Syntax

spellcontrol.**DisableSuggestOptions** = *options*

Part	**Type**	**Description**
options	Integer	Determines how options are cleared. options can be composed of multiple settings combined with the OR operator or by addition. Following are the valid constants for this argument:

Constant	**Descriptions**
VSSUGOPT_CAPITALIZATION	Disables the option that checks the word in lowercase, initial caps, and all caps.
VSSUGOPT_CHARSWAP	Disables the option that swaps each successive pair of letters and checks the result.
VSSUGOPT_DELETES	Disables the option that removes each letter in the word in sequence (checks for accidental extra character).
VSSUGOPT_DOUBLES	Disables the option that moves double letters to adjacent letters. For example, seetings is tried as ssetings and settings.
VSSUGOPT_HYPHENS	Disables the option that inserts a hyphen between each successive pair of letters (checks for missing hyphen).
VSSUGOPT_SPLITS	Disables the option that inserts a space between each successive pair of letters in the word and checks the resulting two words.
VSSUGOPT_EXCHANGES	Disables the option that replaces each letter in the word with all other alphabetic characters and checks the result.
VSSUGOPT_INSERTIONS	Disables the option that inserts an extra letter between each successive pair of letters, running through the entire alphabet in each instance.

Remarks

VisualSpeller creates suggested spellings by modifying the misspelled word and checking the result using techniques controlled by the suggestion options. All techniques are enabled by default.

To disable all suggestion options, set to 0 or VS_ALL.

DLLHandle Property

Description
Returns the handle to the VisualSpeller control.

Syntax
*handle = spellcontrol.***DLLHandle**

Remarks
This property is not needed in general usage. However, if you write your own DLL, you can use this property to return the handle to the speller control.

EnableCommonIRList Property

Description

Enables or disables the common ignore/replace list.

Syntax

spellcontrol.**EnableCommonIRList** = *boolean*

If this property is True, the common ignore/replace list is enabled. If this property is False, the common ignore/replace list is disabled.

EnableCustomUpdate Property

Description
Enable or disable the ability to update the specified custom dictionary.

Syntax
spellcontrol.**EnableCustomUpdate** (*index*) = *status*

Part	**Type**	**Description**
index	Integer	Specifies a dictionary. Negative numbers refer to custom dictionaries. The dictionary category VSCAT_CUSTOM can be used to enable or disable update for all custom dictionaries.
status	Boolean	If *status* is True, the dictionary can be updated. If *status* is False, the dictionary cannot be updated.

EnableDialogElements Property

Description

Enables one or more dialog elements in the Word Not Found in Dictionary dialog box and the Spell Options dialog box.

Syntax

spellcontrol.**EnableDialogElements** = *elements*

Part	**Type**	**Description**
elements	Long	Determines which dialog box elements are enabled. This can be one or more of the following elements combined by addition:
		Elements in Word Not Found in Dictionary Dialog
		VSD_WORD_MISSPELLED_HELP
		VSD_OPTIONS
		VSD_PROMPT_REPLACE
		VSD_ADD_TO_CUSTOM
		VSD_REPLACE_ALL
		VSD_REPLACE
		VSD_IGNORE_ALL
		VSD_IGNORE
		VSD_SUGGEST_NOT_FOUND
		VSD_SUGGEST_REPLACE_WITH
		VSD_CANCEL_SPELLCHECK
		VSD_SUGGESTIONS_LIST
		VSD_REPLACE_BOX
		VSD_CUSTOM_LIST
		VSD_THESAURUS
		Elements in Spell Options Dialog Box
		VSD_OPTIONS_HELP
		VSD_CUSTOMS
		VSD_STANDARDS
		VSD_PERFORMANCE
		VSD_SUGGESTION_LIMIT
		VSD_AUTO_SUGGESTIONS
		VSD_EXACT_MATCH
		VSD_IGNORE_FULL_CAPS
		VSD_IGNORE_PARTIAL
		VSD_IGNORE_PURE
		VSD_ALLOW_JOINED
		VSD_RECHECK

Remarks

Any elements not referenced are left unchanged. Set to 0 or VS_ALL to enable all elements.

EnableDictionary Property

Description

Enables or disables the specified dictionary.

Syntax

spellcontrol.**EnableDictionary** (*index*) = *status*

Part	**Type**	**Description**
index	Integer	Specifies a dictionary. Positive numbers refer to standard dictionaries. Negative numbers refer to custom dictionaries. You can also use the dictionary categories to enable or disable a set of dictionaries.
status	*Boolean*	Determines the status of the specified dictionary. If *status* is True, the dictionary is enabled. If *status* is False, the dictionary is disabled.

Remarks

In order to use a dictionary for spellchecking, it must be opened, loaded into memory, and enabled.

EnableEventOptions Property

Description

Enables spellcheck events.

Syntax

spellcontrol.**EnableEventOptions** = *events*

Part	**Type**	**Description**
events	Integer	Determines which events are enabled. This can be composed of multiple settings combined with the OR operator or by addition. events can assume the following constants:

Constant	**Description**
VSEVOPT_COMPLETE	Enables the **Complete** event.
VSEVOPT_FOUND	Enables the **Found** event.
VSEVOPT_MISSPELLED	Enables the **Misspelled** event.
VSEVOPT_BEFORE_REPLACE	Enables the **BeforeReplace** event.
VSEVOPT_POPUP	Enables the **AfterPopup** event.
VSEVOPT_CHECK_STATUS	Enables the **CheckStatus** event.
VSEVOPT_CHECK_ERROR	Enables the **CheckError** event.
VSEVOPT_AFTER_REPLACE	Enables the **AfterReplace** event.
VSEVOPT_CLICK_IN	Enables the **ClickIn** event.
VSEVOPT_CLICK_OUT	Enables the **ClickOut** event.
VSEVOPT_DEFAULTS	Enables all events except the **Found** event.

Remarks

The state of events not specified are left unchanged. Set to 0 or VS_ALL to enable all events.

Use **DisableEventOptions** to disable events and **EventOptions** to return the enabled status of events.

EnableSpellOptions Property

Description

Enables the options for how a spellcheck is performed.

Syntax

spellcontrol.**EnableSpellOptions** = *options*

Part	**Type**	**Description**
options	Integer	Determines which spell options are enabled. This can be composed of multiple settings combined with the OR operator or by addition. options can assume the following constants:

Constant	**Description**
VSOPT_AUTO_REPLACE	Sets the **AutoReplace** property to True. VisualSpeller automatically generates a suggestion list for a misspelled word.
VSOPT_AUTO_SUGGEST	Sets the **AutoSuggest** property to True. This automatically displays a dialog box when a misspelled word is encountered.
VSOPT_AUTO_POPUP	Sets the **AutoPopUp** property to True. This automatically displays a dialog box when a misspelled word is encountered.
VSOPT_IGNORE_FULL_ CAPS	Sets the **IgnoreFullCaps** property to True. This causes VisualSpeller to ignore words in all caps during a spellcheck.
VSOPT_IGNORE_PARTIAL _NUMBERS	Sets the **IgnorePartialNumbers** property to True. This causes VisualSpeller to ignore words that contain a mixture of letters and numbers during a spellcheck.
VSOPT_MULTILINE	Sets the **MultiLine** property to True. VisualSpeller keeps track of line breaks during a spellcheck.
VSOPT_IGNORE_PURE_ NUMBERS	Causes VisualSpeller to ignore all-numeral words.
VSOPT_ALLOW_JOINED_ WORDS	Sets the **AllowJoinedWords**property to True. VisualSpeller considers hyphenated words are spelled correctly if the component words are individually spelled correctly.
VSOPT_EXACT_MATCH	Forces VisualSpeller to require an exact case match.
VSOPT_RETURN_EACH_ WORD	Causes to return after each word when spellchecking a block of text with the **CheckText** property.
VSOPT_REPLACE_ RECHECK	Sets the **ReplaceRecheck** property to True. This forces Visual Speller to check words typed by the user in the Word Not Found in Dictionary dialog box.
VSOPT_DEFAULTS	Sets the **AutoReplace**, **AutoSuggest**, **AutoPopup**, and **ReplaceRecheck**

properties, as well as the ignore pure numbers option, to True. These are the default settings for a spellcheck session.

Remarks

Set to 0 or VS_ALL to enable all spell options.

EnableStandardIRList Property

Description

Enables or disables the ignore/replace list associated with the specified standard dictionary.

Syntax

spellcontrol.**EnableStandardIRList** (*index*) = *status*

Part	**Type**	**Description**
index	Integer	Specifies a dictionary. Positive numbers refer to standard dictionaries. The dictionary categories VS_ALL, VSCAT_STANDARD, VSCAT_ENABLED, and VSCAT_DISABLED can be used to enable or disable multiple standard dictionary ignore/replace lists.
status	Boolean	Sets the state of the ignore/replace list. If *status* is True, the ignore/replace list is enabled. If *status* is False, the ignore/replace list is disabled.

EnableSuggestOptions Property

Description

Enables the options for how suggestions are generated.

Syntax

spellcontrol.**EnableSuggestOptions** = *options*

Part	Type	Description
options	Integer	Determines which options are enabled. *options* can be composed of multiple settings combined with the OR operator or by addition. Following are the valid constants for this argument:

Constant	Descriptions
VSSUGOPT_CAPITALIZATION	Checks the word in lowercase, initial caps, and all caps.
VSSUGOPT_CHARSWAP	Swaps each successive pair of letters and checks the result.
VSSUGOPT_DELETES	Removes each letter in the word in sequence (checks for accidental extra character).
VSSUGOPT_DOUBLES	Moves double letters to adjacent letters. For example, seetings is tried as ssetings and settings.
VSSUGOPT_HYPHENS	Inserts a hyphen between each successive pair of letters (checks for missing hyphen).
VSSUGOPT_SPLITS	Inserts a space between each successive pair of letters in the word and checks the resulting two words.
VSSUGOPT_EXCHANGES	Replaces each letter in the word with all other alphabetic characters and checks the result.
VSSUGOPT_INSERTIONS	Inserts an extra letter between each successive pair of letters, running through the entire alphabet in each instance.
VSSUGOPT_DEFAULTS	Enables all suggestion options.

Remarks

VisualSpeller creates suggested spellings by modifying the misspelled word and checking the result using techniques controlled by the suggestion options. All techniques are enabled by default. Set to 0 or VS_ALL to enable all suggestion options.

ErrorOffset Property

Description
Sets the offset for runtime errors for VSR* error values.

Syntax
spellcontrol.**ErrorOffset** = *offset*

Part	Type	Description
offset	Integer	Sets offset for runtime errors. Runtime error = VSR_* errorcode + ErrorOffset.

Remarks
By default, the offset is set to 32350.

ErrorText Property

Description

Returns the text of the error message for the current **ResultCode**.

Syntax

string =*spellcontrol*.**ErrorText**

Part	Type	Description
string	String	Variable that receives the returned text.

ErrorTitle Property

Description

Sets or returns the title for the Error message box. Setting to the empty string "" reverts to the default title.

Syntax

spellcontrol. **ErrorTitle** [= *string*]

Part	**Type**	**Description**
string	String	Error message box title.

EventOptions Property

Description

Returns the enabled spellcheck events.

Syntax

events = *spellcontrol*.**EventOptions**

Part	Type	Description
events	Integer	Determines which events are enabled. This can be composed of multiple settings combined with the OR operator or by addition. events can assume the following constants:

Constant	Description
VSEVOPT_COMPLETE	The **Complete** event is enabled.
VSEVOPT_FOUND	The **Found** event is enabled.
VSEVOPT_MISSPELLED	The **Misspelled** event is enabled.
VSEVOPT_BEFORE_REPLACE	The **BeforeReplace** event is enabled.
VSEVOPT_POPUP	The **AfterPopup** event is enabled.
VSEVOPT_CHECK_STATUS	The **CheckStatus** event is enabled.
VSEVOPT_CHECK_ERROR	The **CheckError** event is enabled.
VSEVOPT_AFTER_REPLACE	The **AfterReplace** event is enabled.
VSEVOPT_CLICK_IN	The **ClickIn** event is enabled.
VSEVOPT_CLICK_OUT	The **ClickOut** event is enabled.

Remarks

Use **EnableEventOptions** to enable events and **DisableEventOptions** to disable events.

Example

The following example enables Misspelled and Complete events and disables all other events:

```
    VSpell1.EventOptions = VSEVOPT_MISSPELLED + VSEVOP_COMPLETE
```

FindSuggestions Property

Description

Generates a list of suggested spellings for the current misspelled word.

Syntax

spellcontrol.**FindSuggestions** = *string*

Part	Type	Description
string	String	Word for which to generate a suggestion list.

Remarks

Assigning an empty string to string causes VisualSpeller to generate suggestions for the last misspelled word.

Use **Suggestion** to access the generated suggestions.

Example

The first example finds suggestions for the word returned by the MisspelledWord property:

```
VSpell1.FindSuggestions = " "
```

The second example finds suggestions for the word "gorble":

```
VSpell1.FindSuggestions = "gorble"
```

Found Event

Description

This event occurs when a word is found in a dictionary.

Syntax

Private Sub *spellcontrol*_**Found** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	Description
VS_DEFAULT_HANDLING	Spellchecking halts and control is returned to the calling procedure with the result code VSR_FOUND.
VS_EVENT_HANDLED	Spellchecking continues.
VS_CANCEL_SPELLCHECK	Spellchecking halts and control is returned to the calling procedure with VSR_CHECK_CANCELED the result.

Remarks

Found events occur only when the found events are enabled. They are useful when you must tabulate data on found words. *EventAction* is preset to VS_EVENT_HANDLED on entry to the **Found** procedure. It is preset to VS_DEFAULT_HANDLING if the VSOPT_RETURN_EACH_WORD spell option is enabled. You can change *EventAction* by setting it to one of the other values during the Found procedure.

GetEntry Property

Description

Retrieves an entry from a custom dictionary or from a standard dictionary ignore/replace list. The retrieved word is made available through the **MisspelledWord**, **ReplacementWord**, **Hyphenation**, and **IRAction** properties.

Syntax

spellcontrol.**GetEntry** (*index*) = *entryNumber*

Part	**Type**	**Description**
index	Integer	Dictionary in the DictionaryName array from which to return an entry. Negative numbers specify custom dictionaries. Positive numbers specify ignore/replace lists associated with standard dictionaries.
entryNumber	Long	Identifies the word to retrieve from the list. Words are identified by number, with 1 referring to the first entry in the list.

Remarks

By getting successive entries, you can read out the entire dictionary. When you reach the end of a dictionary, the **ResultCode** is VSR_END_LIST. Because dictionaries are shared among applications, it is conceivable that another application could change a custom dictionary. If this were to happen, the **ResultCode** would be VSR_CHANGED. The probability of such a change is low. If one should occur, however, you should re-read the dictionary beginning with the first entry. Reading the first entry resets the VSR_CHANGED status.

Example

The following example places in a list box all words in the common ignore replace list that age to be ignored without confirmation:

```
DIM X As Integer
DIM ResCode As Integer
DIM IRAction As Integer


Do
    ListBox.Clear
    X = 1
        Do
            VSpell1.GetEntry(VSCAT_IRLIST) = X
            ResCode = VSpell1.ResultCode
            IRAction = VSpell1.IRAction And (VSR_IGNORE + VSR_GLOBAL)
            If ResCode = 0 Then
               If IRAction = VSR_IGNORE + VSR_GLOBAL Then
                   ListBox.AddItem VSpell1.MisspelledWord
               End If
            Else
               Exit Do
            End If
            X = X + 1
        Loop
    If ResCode = VSR_END_OF_LIST Then
        Exit Do
    End If
Loop
```

Hyphenation Property

Description

Sets or returns hyphenation information for the last correctly spelled word.

Syntax

spellcontrol.**Hyphenation** [= *string*]

Part	Type	Description
string	String	Special string in which each character can be interpreted as a binary number representing the length of a syllable. A null string indicates that no hyphenation is possible or that hyphenation information is not available in the dictionary.

Example

The following example assigns to the variable Last$ the hyphenation information for the last correctly spelled word and displays the length of the word's first syllable.

```
Last$ = VSpell1.Hyphenation
If Last$ < > " " Then
    MsgBox "Length of first syllable of " &VSpell1.ChecKWord &" = " & Str (Asc(Last$))
Else
    MsgBox "No hyphenation available for word"
End If
```

IgnoreFullCaps Property

Description

Determines whether VisualSpeller ignores words in all caps during a spellcheck.

Syntax

spellcontrol.**IgnoreFullCaps** = *boolean*

Remarks

If this property is True, all-caps words are ignored during a spellcheck. If this property is False, all-caps words are not ignored.

IgnorePartialNumbers Property

Description

Determines whether VisualSpeller ignores words that contain a mixture of letters and numbers during a spellcheck.

Syntax

spellcontrol.**IgnorePartialNumbers** = *boolean*

Remarks

If this property is True, words with numbers are ignored during a spellcheck. If this property is False, words with numbers are not ignored.

IRAction Property

Description

Returns a code that specifies what action to take when a word is found in the ignore/replace list.

Syntax

action = *spellcontrol*.**IRAction**

Part	Type	Description
action	Integer	Variable that receives one of the following codes, indicating the action VisualSpeller will take regarding the misspelled word.

Constant	Description
VSIR_MISSPELLED	Treats word as a normal misspelling - generally with an automatically displayed dialog box.
VSIR_PROMPT_REPLACE	Treats word as a misspelling, but does not generate suggestions. Offers **ReplacementWord** as the default replacement.
VSIR_REPLACE_ALL	Replaces word with **ReplacementWord**. Normally, this occurs automatically, but if **AutoReplace** is False, replacement must be handled by the program.

Remarks

This property is relevant when the value of **ResultCode** is VSR_IGNORE_REPLACE at the end of a **CheckText**, **BeginCheck**, **ResumeCheck**, or **CheckWord** action or inside a **Misspelled** event.

IRWhereFound Property

Description

Returns an index to the DictionaryName array that specifies in which ignore/replace list the last correctly spelled word was found.

Syntax

list = *spellcontrol*.**IRWhereFound**

Part	Type	Description
list	Integer	Variable that receives the returned ignore/replace list index. This variable contains 0 if the word was found in the common ignore/replace list. Otherwise, it contains the index of the standard dictionary associated with the ignore/replace list that contained the word.

Example

For words found in an ignore/replace list, the following example displays a message that indicates which list contained the word:

```
If VSpell1.WhereFound = 0 Then
    If VSpell1.IRWhereFound = 0 Then
        MsgBox "Word found in common IR list."
    Else
        MsgBox "Word found in IR list for dictionary " & DictionaryName
                (VSpell1.IRWhereFound)
    End If
End If
```

LineBreak Property

Description

Sets or returns the characters and character combinations that are treated as line breaks during a multiline spellcheck.

Syntax

spellcontrol.**LineBreak** [= *breakString*]

Part	Type	Description
breakString	String	A string consisting of one or more groups of decimal numbers. The first in each group is the number of characters that follow in the group. Succeeding numbers in each group are decimal values for non-alphanumeric characters which define a possible end of line sequence. The string ends with a zero-length group. The default string specifies CR and LF in all variants. The default breakstring is 2 13 10 2 10 13 1 13 1 10 0.

Example

The following example specifies that only CRLF be treated as a line break:

```
VSpell1.LineBreak = Chr(2) & Chr(13) & Chr(10)
```

LineOffset Property

Description

Sets or returns the character offset within the current line at which spellchecking occurs or resumes.

Syntax

spellcontrol.**LineOffset** [= *line*]

Part	**Type**	**Description**
line	Integer	The character offset (based at 0) within the current line of the last misspelled word.

Example

The following example displays the position of the misspelled word.

```
MsgBox "Word misspelled at line " & Str (VSpell1.CurrentLine + 1) & " character " &
        Str(VSpell1.LineOffset + 1)
```

LoadDictionary Property

Description

Loads the specified dictionary.

Syntax

spellcontrol.**LoadDictionary** (*index*) = 1

Part	Type	Description
index	Integer	Specifies a dictionary in the DictionaryName array. Positive numbers indicate standard dictionaries and negative numbers indicate custom dictionaries.

Remarks

In order to use a dictionary for spellchecking, it must be opened, loaded into memory, and enabled.

You may wish to unload a dictionary and conserve memory when a spellcheck is not in progress. Use **DictionaryIsLoaded** to determine whether a dictionary is loaded and **UnloadDictionary** to unload the dictionary.

MaxSuggestions Property

Description

Sets or returns the maximum number of correction suggestions that VisualSpeller generates and displays.

Syntax

spellcontrol.**MaxSuggestions** = *max*

Part	Type	Description
max	Integer	Maximum number of displayed suggestions.

Misspelled Event

Description

This event occurs when a potentially misspelled word is encountered during a spellcheck.

Syntax

Private Sub *spellcontrol*_**Misspelled** (*EventAction* As Integer)

Following are the valid constants for *EventAction:*

Constant	**Description**
VS_DEFAULT_HANDLING	If **AutoPopUp** is True, the Word Not Found in Dictionary dialog box appears. Otherwise, control returns to the calling procedure.
VS_EVENT_HANDLED	Spellchecking continues with the next word. **Text** and **ResumeOffset** properties can be used to adjust the text being spellchecked.
VS_CANCEL_SPELLCHECK	Spellchecking halts and control is returned to the calling procedure with VSR_CHECK_CANCELED the result.

Remarks

This event occurs for each misspelled word, including ignore/replace prompting. The event occurs before a dialog box popup and thus acts in place of a BeforePopUp event. *EventAction* is preset to VS_DEFAULT_HANDLING on entry to the **Misspelled** procedure. After responding to the misspelled word, you can change *EventAction* by setting it to one of the other values.

MisspelledWord Property

Description
Sets or holds the text of the last misspelled word.

Syntax
spellcontrol.**MisspelledWord** = *word*

Part	Type	Description
word	String	Text of the last misspelled word.

Remarks
Generally, VisualSpeller keeps track of the last misspelled word. However, there are times when you may want to provide a word that VisualSpeller treats as the last misspelled word.

Since actions such as **AddToCommonIRList**, **AddToStandardIRList**, and **AddToCustom** use the last misspelled word, setting the misspelled word can be useful for automatically generating custom dictionaries or preset ignore/replace lists.

Example
The following example displays the most recent misspelled word.

```
MsgBox "Misspelled: " & VSpell1.MisspelledWord
```

MultiLine Property

Description

Determines whether VisualSpeller keeps track of line breaks during a spellcheck.

Syntax

spellcontrol.**MultiLine** [= *boolean*]

Remarks

If this property is True, **CurrentLine** returns the number of the line that contains the last misspelled word and **LineOffset**returns the position within that line of the misspelled word.

If this property is False, VisualSpeller does not keep track of line breaks.

MultiLine settings are valid only during **CheckText** actions.

OpenCustom Property

Description

Opens a specified custom dictionary.

Syntax

spellcontrol.**OpenCustom** (*loadit*) = *DictionaryName*

Part	**Type**	**Description**
loadit	Boolean	Indicates whether the dictionary is immediately loaded into memory when it is opened. If True, the dictionary is loaded; if False, it is not loaded.
DictionaryName	String	Identifies the dictionary to open.

Remarks

This property opens the custom dictionary specified by **DictionaryName**for spellchecking. Check **ResultCode** after calling this property. If the operation was successful, -**CustomCount** returns the dictionary index number.

OpenStandard Property

Description

Opens a specified standard dictionary.

Syntax

spellcontrol.**OpenStandard** (*loadit*) = *DictionaryName*

Part	**Type**	**Description**
loadit	Boolean	Indicates whether the dictionary is immediately loaded into memory when it is opened. If True, the dictionary is loaded; if False, it is not loaded.
DictionaryName	String	Identifies the dictionary to open.

Remarks

This property opens the standard dictionary specified by DictionaryName for spellchecking. Check **ResultCode** after calling this property. If the operation was successful, **StandardCount** returns the dictionary index number.

OptionsHelpFile Property

Description

Sets or returns the name and path of the help file associated with the Spell Options dialog box.

Syntax

spellcontrol.**OptionsHelpFile** [= *string*]

Part	**Type**	**Description**
string	String	Name and path of the help file displayed when the dialog box help button is clicked. The default is "VSPELLER.HLP".

OptionsTitle Property

Description

Sets or returns the title for the Spell Options dialog box.

Syntax

spellcontrol.**OptionsTitle** = *string*

Part	**Type**	**Description**
string	String	Dialog box title. Setting this property to the empty string "" reverts to the default title.

PopupError Property

Description
Displays the internal Error dialog box, which presents the text of the error contained in the **ResultCode**.

Syntax
spellcontrol.**PopupError** = 1

PopupOptions Property

Description

Displays the internal Spell Options dialog box, which will use the current context information to display spellchecker options and allow the user to take action.

Syntax

spellcontrol.**PopupOptions** = 1

PopupWordMisspelled Property

Description

Displays the internal Word Not Found in Dictionary dialog box, which will use the current context information to display the misspelled word and allow the user to take action.

Syntax

spellcontrol.**PopupWordMisspelled** = 1

RemoveFromCommonIRList Property

Description
Removes an entry from the common ignore/replace list.

Syntax
spellcontrol.**RemoveIRListEntry** = *text*

Part	**Type**	**Description**
text	String	Entry to remove from the list.

Remarks
If *text* is not found in the common ignore/replace list, **ResultCode** is set to the constant VSR_ENTRY_NOT_FOUND.

RemoveFromStandardIRList Property

Description

Removes an entry from the ignore/replace list associated with a specific standard dictionary.

Syntax

spellcontrol.**RemoveIRListEntry** (*index*) = *text*

Part	**Type**	**Description**
index	Integer	Specifies a dictionary in the DictionaryName array or a category of dictionaries. Setting *index* to the dictionary category VS_ALL removes *text* from all standard ignore/replace lists at once.
text	String	Entry to remove from the list.

Remarks

If *text* is not found in the specified ignore/replace list(s), **ResultCode** is set to the constant VSR_ENTRY_NOT_FOUND.

ResultCode reflects a failure to find the word in any of the dictionaries or common ignore/replace list. Therefore, the result code is VSR_ENTRY_NOT_FOUND, unless the word is found in all the lists. To more accurately check for errors, individually remove words from each list in sequence. With only one targeted list on each call, the result code correctly reflects whether the word is found and removed in that list.

ReplaceCount Property

Description

Sets or returns the number of words replaced during the current spellcheck.

Syntax

spellcontrol.**ReplaceCount** [= *count*]

Part	Type	Description
count	Long	The number of words.

Remarks

This property counts global replacements, replacements made in response to a popup and replacements made using the **ReplaceLastWord** property.

Example

The following example displays the number of replacements made during a spellcheck.

```
VSpell1.ReplaceCount = 0
.
.  'normal spellcheck
.
MsgBox "Replaced " & Str(VSpell1.ReplaceCount) & " words"
```

ReplaceLastWord Property

Description

Replaces the current misspelled word in the buffer text with **ReplacementWord**.

Syntax

spellcontrol.**ReplaceLastWord** = 1

Remarks

Using this property does not cause a **BeforeReplace** or **AfterReplace** event. Replacement location is based on **WordOffset** and **ResumeOffset**, so changing these properties can adversely affect the operation of **ReplaceLastWord**.

ReplacementWord Property

Description

Sets or returns the word to be used when a replacement is requested.

Syntax

spellcontrol.**ReplacementWord** [= *word*]

Part	**Type**	**Description**
word	String	The word to be used as a replacement.

Remarks

Following a misspell condition with a VSR_IGNORE_REPLACE status and **IRAction** set to VSIR_REPLACE, this property contains the word that replaces the **MisspelledWord**. You can change the word before replacement occurs, usually in response to a popup dialog box. **ReplacementWord** is also used by **ReplaceLastWord**, **AddToCustom**, **AddToStandardIRList**, and **AddToCommonIRList** You can change it prior to calling any of these properties.

Examples

The following example sets the replacement word to that entered in a text box and then replaces the misspelled word.

```
VSpell1.ReplacementWord = TextBox.Text

VSpell1.ReplaceLastWord = True
```

ReplaceOccurred Property

Description

Indicates whether VisualSpeller made replacements since the last time this property was called.

Syntax

replacement = *spellcontrol*.**ReplaceOccurred**

Part	Type	Description
replacement	Integer	Variable that receives the state of the property. 0 is False and 1 is True.

Remarks

This property is cleared when it is read or when new text is loaded.

Example

The following example places the changed text in a text box if replacments were made during a spellcheck:

```
If VSpell1.ReplaceOccurred Then '
    TextBox.Text = VSpell1.Text
End If
```

ReplaceRecheck Property

Description

Determines whether VisualSpeller checks words typed by the user in an automatically displayed Word Not Found in Dictionary dialog box.

Syntax

spellcontrol.**ReplaceRecheck** = *boolean*

Remarks

If this property is True, replacement words typed by the user in the Word Not Found in Dictionary dialog box are spellchecked. This does not affect words chosen from the suggestion list.

If this property is False, the replacement words are not spellchecked.

ResultCode Property

Error Conditions Status Conditions

Description

Sets or returns the result of the last spellcheck action.

Syntax

spellcontrol.**ResultCode** = *code*

Part	Type	Description
code	Integer	When *code* is 0, the spellcheck was successful. A nonzero value indicates failure or a warning condition.

Remarks

ResultCode is not changed by properties that to do not invoke an action or take a dictionary index.

Example

The following example displays the error code when a VisualSpeller error occurs:

```
If VSpell1.ResultCode > 0 Then
    MsgBox "Error code = " + Str$(VSpell1.ResultCode)
End If
```

ResumeCheck Property

Error Conditions Status Conditions

Description

Resumes a spellcheck after control has been passed to the application.

Syntax

resultcode = *spellcontrol*.**ResumeCheck**

Part	**Type**	**Description**
ResultCode	Integer	Variable that receives the current value of the **ResultCode** property. When *ResultCode* is 0, the spellcheck was successful. A nonzero value indicates failure or a warning condition.

Remarks

ResumeCheck is most often used to continue a spellcheck begun with a **CheckText** or **BeginCheck** action. It can however, be used to begin a spellcheck when you need specific control over where spellchecking begins.

This property continues the spellcheck at **ResumeOffset**. In general, use this property to resume spellchecking at the next word. Use **ResumeWithRecheck** to resume with first rechecking a replaced word.

The example checks the content of a text box without using events or automatically displayed dialog boxes.

```
DIM ResCode As Integer

VSpell1.EventOptions = 0
VSpell1.AutoPopUp =False
VSpell1.CheckText = TextBox.Text     ' start spellcheck
ResCode = VSpell1.ResultCode         ' get result
Do While ResCode < 0
    Select Case ResCode
        Case VSR_WORD_MISSPELLED, VSR_IGNORE_REPLACE
            VSpell1.PopUpWordMisspelled  ' display misspell dialog
            ResCode = VSpell1.ResultCode    ' get new result
        Case VSR_BREAK
            Do Events                    ' give Windows a time slice
    End Select
    If ResCode = VSR_CHECK_CANCELED   ' see if canceled
        Exit Loop
    End If
    ResCode = VSpell1.ResumeCheck     ' resume w/o rechecking
Loop
TextBox.Text = VSpell1.Text           ' read out spellchecked text
```

ResumeOffset Property

Description

Sets or returns the zero-based offset within a text block where spellchecking resumes when the **ResumeCheck** property is called.

Syntax

offset = *spellcontrol*.**ResumeOffset**

Part	**Type**	**Description**
Offset	Long	Variable that receives the zero-based offset within text block.

Remarks

This property is valid during a **CheckText** action when a misspelling has occurred. It is indeterminate otherwise. It is set to zero by **BeginCheck** and **CheckText** actions. Changing **ResumeOffset** changes where VisualSpeller resumes spellchecking after handling a misspelled word.

Example

The example starts a spellcheck on a text block at the current insertion point, if there is one.

```
If TextBox.SelText = "" Then         ' Null string means SelStart
    VSpell1.Text = TextBox.Text      ' will be insertion point
    VSpell1.ResumeOffset = TextBox.SelStart
    VSpell1.ResumeCheck = VSR_NOTHING_TO_CHECK
    .
    .
    .
End If
```

ResumeWithRecheck Property

Error Conditions   Status Conditions

Description

Resumes a spellcheck and rechecks the previous word.

Syntax

resultCode = *spellcontrol*.**ResumeWithRecheck**

Part	Type	Description
resultCode	Integer	Variable that receives the value of the **ResultCode** property. When *resultCode* is 0, the spellcheck was successful. A nonzero value indicates failure or a warning condition.

Remarks

ResumeWithRecheck continues the spellcheck at **WordOffset**. The first word checked increments **WordCount** only when **ResumeOffset** and **WordCount** are the same. This situation indicates that the user deleted the misspelled word from **Text** or that the offsets were changed by the application.

SearchOrder Property

Description

Sets or returns the order in which the three word sources (standard dictionaries, custom dictionaries, and ignore/replace list) are searched during a spellcheck.

Syntax

spellcontrol.**SearchOrder** [= *setting*]

Part	Type	Description
setting	Integer	Valid settings for this property are:

Constant	Description
VSORDER_SCIR	Standard, custom, ignore/replace list
VSORDER_CIRS	Custom, ignore/replace list, standard
VSORDER_IRSC	Ignore/replace list, standard, custom
VSORDER_SIRC	Standard, ignore/replace list, custom
VSORDER_CSIR	Custom, standard, ignore/replace list
VSORDER_IRCS	Ignore/replace list, custom, standard

If a cache miss occurs on a standard dictionary, the search moves to the custom dictionaries and the ignore/replace list before returning to the standard dictionaries. You can defeat this feature by adding the following special setting to any of the order settings:

Constant	Description
VSORDER_NO_LOOKAHEAD	Disable lookahead feature.

Example

The example specifies that VisualSpeller search the ignore/replace list first, then the standard dictionaries, then the custom dictionaries and that the look ahead feature be disabled.

```
VSpell1.SearchOrder = VSORDER_IRSC + VSORDER_NO_LOOKAHEAD
```

SpellOptions Property

Description

Returns the currently enabled options that control how a spellcheck is performed.

Syntax

options = *spellcontrol*.**SpellOptions**

Part	Type	Description
options	Integer	Variable that receives information about which spell options are enabled. This can be composed of multiple settings combined with the OR operator or by addition. options can assume the following constants:

Constant	Description
VSOPT_AUTO_REPLACE	VisualSpeller automatically generates a suggestion list for a misspelled word.
VSOPT_AUTO_SUGGEST	A dialog box is automatically displayed when a misspelled word is encountered.
VSOPT_AUTO_POPUP	A dialog box is automatically displayed when a misspelled word is encountered.
VSOPT_IGNORE_FULL_CAPS	Words in all caps are ignored during a spellcheck.
VSOPT_IGNORE_PARTIAL_NUMBERS	Words that contain a mixture of letters and numbers are ignored during a spellcheck.
VSOPT_MULTILINE	VisualSpeller keeps track of line breaks during a spellcheck.
VSOPT_IGNORE_PURE_NUMBERS	All-numeral words are ignored during a spellcheck.
VSOPT_ALLOW_JOINED_WORDS	VisualSpeller considers hyphenated words are spelled correctly if the component words are individually spelled correctly.
VSOPT_EXACT_MATCH	Forces VisualSpeller to require an exact case match.
VSOPT_RETURN_EACH_WORD	Causes VisualSpeller to return after each word when spellchecking a block of text with the **CheckText** property.
VSOPT_REPLACE_RECHECK	Forces Visual Speller to check words typed by the user in the Word Not Found in Dictionary dialog box.
VSOPT_DEFAULTS	Sets the **AutoReplace**, **Suggestion**, **AutoPopup**, **IgnorePartialNumbers**, and **ReplaceRecheck** properties to True. These are the default

	settings for a spellcheck session.

Remarks

Use **EnableSpellOptions** to enable options and **DisableSpellOptions** to disable options.

StandardCount Property

Description

Returns the number of open standard dictionaries.

Syntax

count = *spellcontrol*.**StandardCount**

Part	**Type**	**Description**
count	Integer	Variable that receives the index number of the most recently opened standard dictionary.

Example

The following example fills a list box with standard dictionary names.

```
Max = VSpell1.StandardCount
For X = 1 to Max
    ListBox.AddItem = VSpell1.DictionaryName(X)
End X
```

StandardDictionary Property

Description

Sets the initial standard dictionary. This is a design-time only property.

Syntax

spellcontrol.**StandardDictionary** = *name*

Part	Type	Description
name	String	Name of the standard dictionary.

Remarks

This property is used internally by VisualSpellers property pages. At runtime, use **OpenStandard** to accomplish the same result.

StandardIRListIsEnabled Property

Description

Returns whether the ignore/replace list for a standard dictionary is enabled.

Syntax

status = *spellcontrol*.**StandardIRListIsEnabled** (*index*)

Part	**Type**	**Description**
index	Integer	The dictionary in the DictionaryName array for which to return the ignore/replace list information. Positive numbers specify standard dictionaries.
status	Boolean	Variable that receives the returned state of the ignore/replace list. If status is True, the list is enabled. If status if False, the list is disabled.:

Remarks

Use **EnableStandardIRList** to enable or disable an ignore/replace list.

Suggestion Property

Description

Returns a suggestion from the array of suggestions generated by VisualSpeller for the most recent misspelled word.

Syntax

SuggestionArray = *spellcontrol.***Suggestion** (*index*)

Part	**Type**	**Description**
index	Integer	Index number from the array of suggestions for which to return a suggestion. The values for the suggestion array can range from 1 to **SuggestionCount.**
SuggestionArray	String	Variable that receives the returned list of suggestions.

SuggestionCount Property

Description

Returns the number of suggestions generated by VisualSpeller for the most recent misspelled word.

Syntax

count = *spellcontrol*.**SuggestionCount**

Part	**Type**	**Description**
count	Integer	Variable that receives the returned size of the **Suggestion** array**.**

Example

The example fills a list box with the current suggestion list.

```
Count = VSpell1.SuggestionCount

For X = 1 to Count

    ListBox.AddItem = VSpell1.Suggestions(X)

Next X
```

SuggestionsLimit Property

Description

Sets or returns the overall number of suggestions that are investigated for each misspelled word.

Syntax

spellcontrol.**SuggestionsLimit** [= *limit*]

Part	**Type**	**Description**
limit	Integer	Number of suggestions

Remarks

When the number of suggestions specified by **SuggestionsLimit** is greater than the value set by **MaxSuggestions**, VisualSpeller investigates the number of suggestions specified by **SuggestionsLimit**. However, only the number of suggestions specified by **MaxSuggestions** are displayed.

SuggestionsMade Property

Description

Indicates whether suggestions have been generated since the misspelled word changed.

Syntax

boolean = *spellcontrol*.**SuggestionsMade**

Remarks

If this property is True, suggestions have been generated. If this property is False, suggestions have not been generated.

This property is set when either **FindSuggestions** or **AddSuggestion** is used. It is reset when **ClearSuggestions** is used or when **MisspelledWord** changes.

SuggestOptions Property

Description

Returns the enabled options that control how suggestions are generated.

Syntax

options = *spellcontrol*.**SuggestOptions**

Part	Type	Description
options	Integer	Variable that receives the enabled option information. Options can be composed of multiple settings combined with the OR operator or by addition. Following are the valid constants for this argument:

Constant	Descriptions
VSSUGOPT_CAPITALIZATION	Checks the word in lowercase, initial caps, and all caps.
VSSUGOPT_CHARSWAP	Swaps each successive pair of letters and checks the result.
VSSUGOPT_DELETES	Removes letter in the word in sequence (checks for accidental extra character).
VSSUGOPT_DOUBLES	Moves double letters to adjacent letters. For example, seetings is tried as ssetings and settings.
VSSUGOPT_HYPHENS	Inserts a hyphen between each successive pair of letters (checks for missing hyphen).
VSSUGOPT_SPLITS	Inserts a space between each successive pair of letters in the word and checks the resulting two words.
VSSUGOPT_EXCHANGES	Replaces each letter in the word with all other alphabetic characters and checks the result.
VSSUGOPT_INSERTIONS	Inserts an extra letter between each successive pair of letters, running through the entire alphabet in each instance.

Remarks

VisualSpeller creates suggested spellings by modifying the misspelled word and checking the result using techniques controlled by this property. All techniques are enabled by default. Use **EnableSuggestOptions** to enable options and to disable options you do not want.

Text Property

Description

Sets or returns the text to be spellchecked.

Syntax

spellcontrol.**Text** [= *text*]

Part	Type	Description
text	String	Holds the text to be spellchecked using a **CheckText**, **BeginCheck**, or **ResumeCheck** action. **CheckText** automatically loads **Text** with the text to be spellchecked, but you can get the same effect by loading **Text** and using the **BeginCheck** property. After a spellcheck in which replacements were made, you must update the source of the text (such as a text control) by reading the **Text** property.

Example

The example loads the **Text** property with text from a control and begins spellchecking from the beginning.

```
VSpell1.Text = TextBox.Text

VSpell1.BeginCheck = True

.

.

.
```

TimerTicks Property

Description

Returns the system timer tick count.

Syntax

count = *spellcontrol*.**TimerTicks**

Part	**Type**	**Description**
count	Long	Variable that receives the returned number of milliseconds since Windows was started.

UnloadDictionary Property

Description

Unloads the specified dictionary.

Syntax

spellcontrol.**UnloadDictionary** (*index*) = 1

Part	Type	Description
index	Integer	Specifies a dictionary in the DictionaryName array. Positive numbers indicate standard dictionaries and negative numbers indicate custom dictionaries.

Remarks

In order to use a dictionary for spellchecking, it must be opened, loaded into memory, and enabled.

You may wish to unload a dictionary and conserve memory when a spellcheck is not in progress. Use **DictionaryIsLoaded** to determine whether a dictionary is loaded and **LoadDictionary** to load the dictionary.

UserData Property

Description
A place to store any information for future use.

Syntax
spellcontrol.**UserData** [**=** *data*]

Part	**Type**	**Description**
data	Long	Variable used to store or receive information.

Version Property

Description

Returns a string that represents the version number of VisualSpeller.

Syntax

version = *spellcontrol*.**Version**

Part	**Type**	**Description**
version	String	Variable that receives the returned version number.

Remarks

This property is for reference only.

WhereFound Property

Description

Returns the dictionary index indicating where a correctly spelled word was found.

Syntax

index = *spellcontrol*.**WhereFound**

Part	**Type**	**Description**
index	Integer	Variable that receives the returned dictionary index. A positive value indicates that the word was found in a standard dictionary. A negative value indicates it was found in a custom dictionary. Zero indicates the word was found in the ignore/replace list, in which case the **IRWhereFound** property can be read to determine which list.

Example

The example displays a message indicating where a word was found in a **CheckWord** action or **Found** event.

```
If VSpell1.WhereFound > 0 Then
    MsgBox "Found in a standard dictionary"
ElseIf VSpell1.WhereFound < 0 Then
    MsgBox "Found in a custom dictionary"
ElseIf VSpell1.WhereFound = 0 Then
    MsgBox "Found in the ignore/replace list"
End If
```

WordCount Property

Description
Counts the number of words spellchecked since the word count was cleared.

Syntax
spellcontrol.**WordCount** [= *count*]

Part	**Type**	**Description**
count	Long	Number of spellchecked words.

WordLength Property

Description

Return the length of misspelled word.

Syntax

length = *spellcontrol*.**WordLength**

Part	**Type**	**Description**
length	Long	Variable that receives the returned length of the misspelled word.

Remarks

Equivalent to **ResumeOffset** - **WordOffset**.

WordMisspelledHelpFile Property

Description

Sets or returns the name and path of the help file associated with the Word Not Found in Dictionary dialog box.

Syntax

spellcontrol.**WordMisspelledHelpFile** [= *string*]

Part	**Type**	**Description**
string	String	Name and path of the help file displayed when the dialog box help button is clicked. The default is "VSPELLER.HLP".

WordMisspelledTitle Property

Description

Sets or returns the title for the Word Not Found in Dictionary dialog box.

Syntax

spellcontrol.**WordMisspelledTitle** [= *string*]

Part	Type	Description
string	String	The dialog box title. Setting this property to the empty string "" reverts to the default title.

WordOffset Property

Description
Specifies the zero-based offset from the beginning of a text block where a misspelled word was found.

Syntax
spellcontrol.**WordOffset** [= *offset*]

Part	Type	Description
offset	Long	Zero-based offset from the beginning of a text block.

Remarks
This property is valid during a **CheckText**, **BeginCheck**, or **ResumeCheck** action when a misspelling has occurred and at the end of a **CheckText** action, when it contains the final length of the text checked. It is indeterminate otherwise. **WordOffset** is used by **ResumeWithRecheck**. It is set to zero by **BeginCheck** and **CheckText** action.

Example

The example displays a misspelled word highlighted in a text control.

```
TextControl.Text = VSpell1.Text

TextControl.SelStart = VSpell1.WordOffset

TextControl.SelLength = Len(VSpell1.MisspelledWord)
```

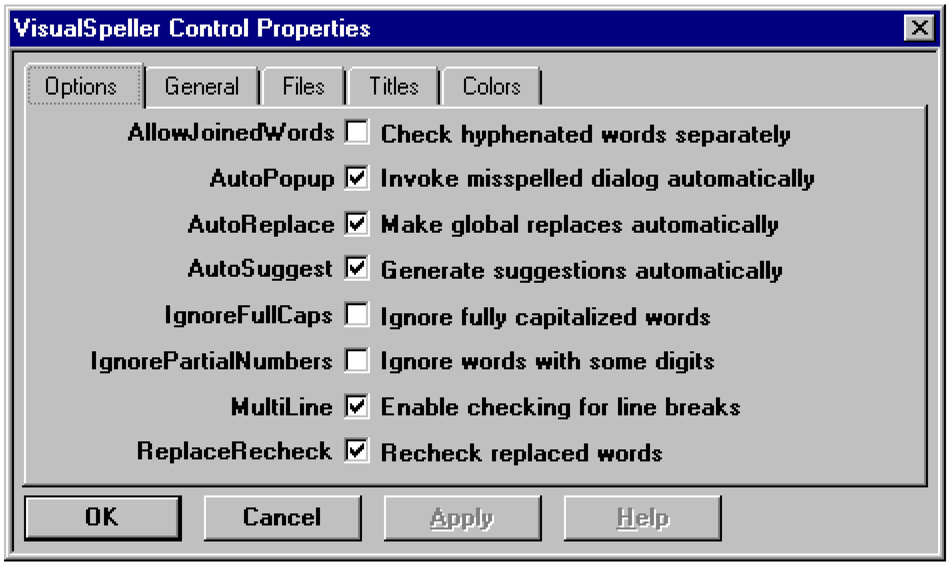
VisualSpeller Control Properties

A number of the most commonly used properties can be set in the VisualSpeller Control Properties dialog box.

To display the VisualSpeller Control Properties dialog box:

1. Right-click on the spellcheck control to display the shortcut menu.
2. Select Properties from the shortcut menu.

Click on a tab in the following illustration to view all the Property pages..



AllowJoinedWords	☐	Check hyphenated words separately
AutoPopup	☑	Invoke misspelled dialog automatically
AutoReplace	☑	Make global replaces automatically
AutoSuggest	☑	Generate suggestions automatically
IgnoreFullCaps	☐	Ignore fully capitalized words
IgnorePartialNumbers	☐	Ignore words with some digits
MultiLine	☑	Enable checking for line breaks
ReplaceRecheck	☑	Recheck replaced words

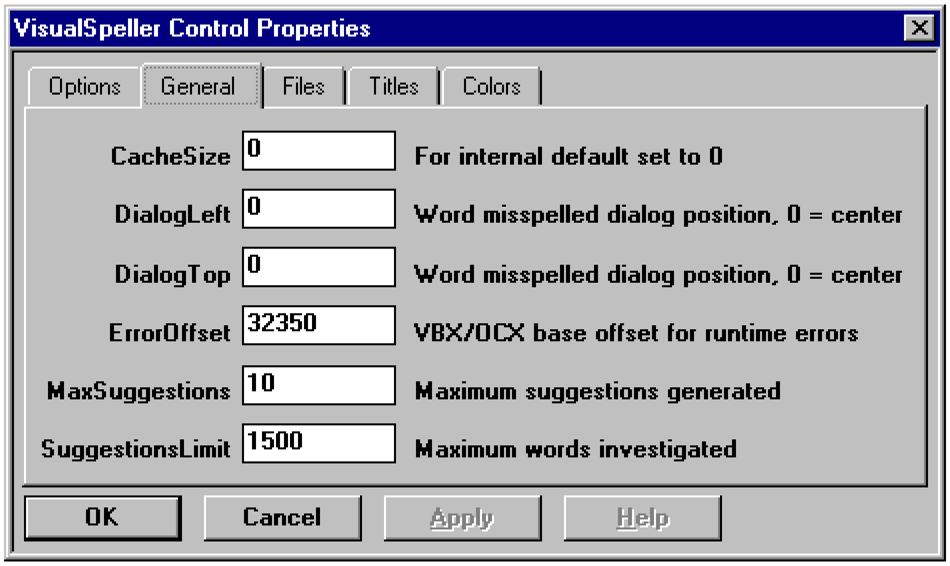
VisualSpeller Control Properties

A number of the most commonly used properties can be set in the VisualSpeller Control Properties dialog box.

To display the VisualSpeller Control Properties dialog box:

1. Right-click on the spellcheck control to display the shortcut menu.
2. Select Properties from the shortcut menu.

Click on a tab in the following illustration to view all the Property pages..



CacheSize	0	For internal default set to 0
DialogLeft	0	Word misspelled dialog position, 0 = center
DialogTop	0	Word misspelled dialog position, 0 = center
ErrorOffset	32350	VBX/OCX base offset for runtime errors
MaxSuggestions	10	Maximum suggestions generated
SuggestionsLimit	1500	Maximum words investigated

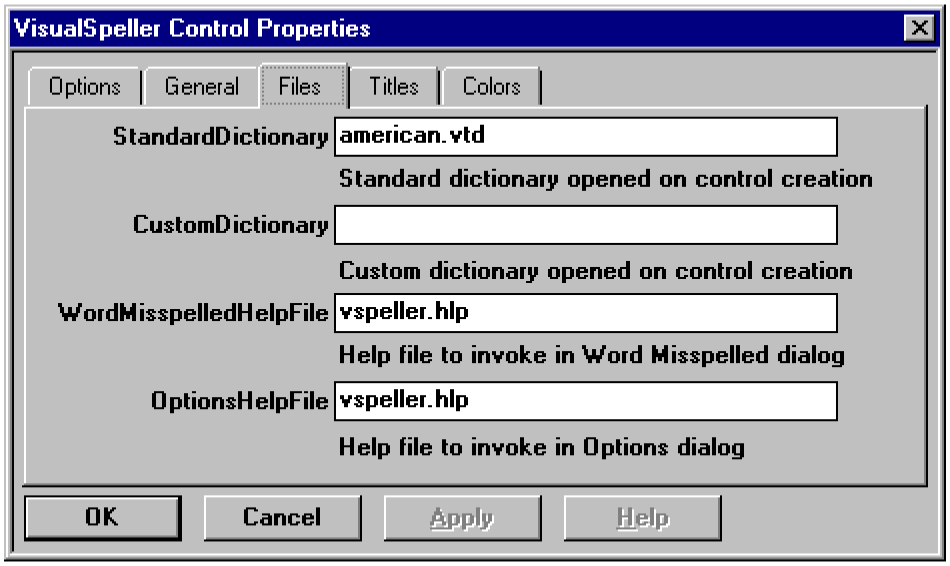
VisualSpeller Control Properties

A number of the most commonly used properties can be set in the VisualSpeller Control Properties dialog box.

To display the VisualSpeller Control Properties dialog box:

1. Right-click on the spellcheck control to display the shortcut menu.
2. Select Properties from the shortcut menu.

Click on a tab in the following illustration to view all the Property pages..



StandardDictionary: american.vtd
Standard dictionary opened on control creation

CustomDictionary:
Custom dictionary opened on control creation

WordMisspelledHelpFile: vspeller.hlp
Help file to invoke in Word Misspelled dialog

OptionsHelpFile: vspeller.hlp
Help file to invoke in Options dialog

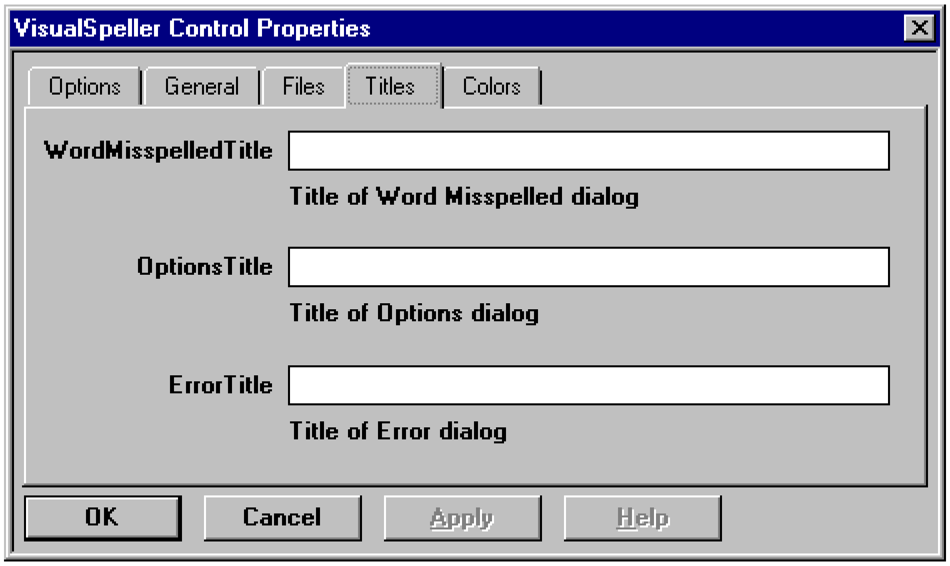
VisualSpeller Control Properties

A number of the most commonly used properties can be set in the VisualSpeller Control Properties dialog box.

To display the VisualSpeller Control Properties dialog box:

1. Right-click on the spellcheck control to display the shortcut menu.
2. Select Properties from the shortcut menu.

Click on a tab in the following illustration to view all the Property pages..



WordMisspelledTitle
Title of Word Misspelled dialog

OptionsTitle
Title of Options dialog

ErrorTitle
Title of Error dialog

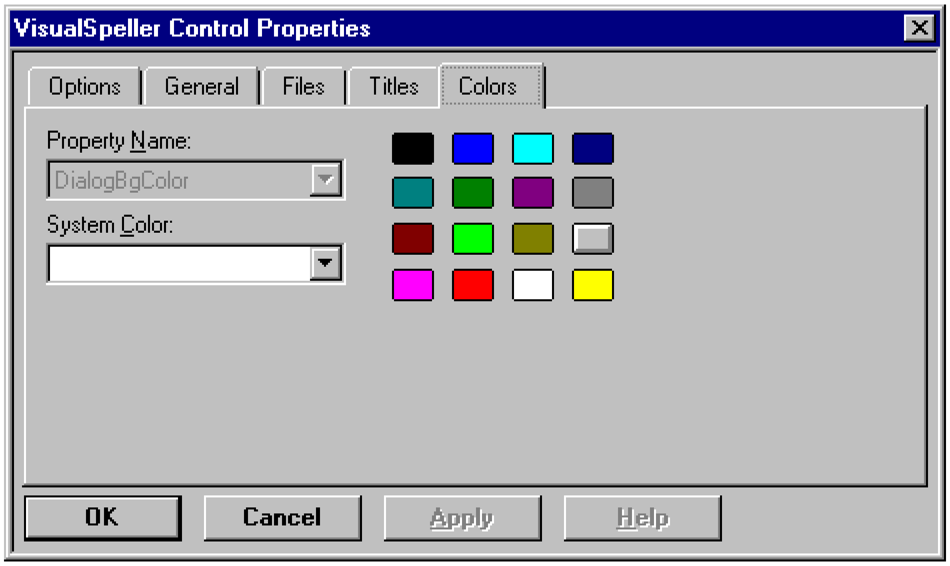
VisualSpeller Control Properties

A number of the most commonly used properties can be set in the VisualSpeller Control Properties dialog box.

To display the VisualSpeller Control Properties dialog box:

1. Right-click on the spellcheck control to display the shortcut menu.
2. Select Properties from the shortcut menu.

Click on a tab in the following illustration to view all the Property pages..



Property Name: DialogBgColor

System Color:

Error and Status Conditions

VisualSpeller sets the **ResultCode** property each time you use an action property or assign a value to a writable property. Negative values indicate status conditions those that are a normal result of an operation. Positive values indicate error conditions.   A value of zero indicates no error.

Reading properties other than action properties and properties with indices does not affect **ResultCode** except in cases where an indexed property is read and the index is invalid. This allows you to retrieve information on an operation without affecting the result code of the previous operation.

Status Constants

Error Constants

Status Constants

Error Constants

Following are the constants you can use to test for status conditions:

Status Constant	Descriptions
VSR_WORD_MISSPELLED	Word misspelled return (check properties).
VSR_IGNORE_REPLACE	Ignore/replace requested (check properties).
VSR_CHECK_CANCELED	Spellcheck cancelled.
VSR_BREAK	Spellcheck break (see BreakWordCount).
VSR_FOUND	Word found in dictionary.
VSR_REPLACED	Word was replaced.
VSR_CHANGED	List/custom changed during GetEntry.
VSR_END_LIST	End of list (GetEntry).
VSR_ENTRY_NOT_FOUND	Word not found (RemoveIREntry).
VSR_CREATED	Custom dictionary created on open.
VSR_OPENED_READONLY	Custom dictionary opened read-only.
VSR_REPLACE_OVERFLOW	Buffer too small on replace action.
VSR_NOT_ADDED	Word not added during AddSuggestion.
VSR_POPPED	Word handled by internal popup dialog box (if VSOPT_RETURN_EACH_WORD option was used).
VSR_CLICK_IN	ClickIn event was triggered.
VSR_CLICK_OUT	ClickOut event was triggered.

Error Constants

Status Constants

The following are constants that you can use to test for error conditions.

Error Constant	Description
VSR_NO_ERROR	No error was generated. Operation was successfully completed.
VSR_WORD_TOO_LONG	Word exceeded 75 characters during assignment to MisspelledWord, ReplacementWord, or Hyphenation.
VSR_NO_MISSPELL	An action was attempted that required a MisspelledWord when none existed. Actions such as AddToIRList and AddToCustom can occur only when a misspell occurs or when a word is explicitly assigned to MisspelledWord in code. MisspelledWord is cleared at the beginning of each spellcheck.
VSR_UNKNOWN_OPTION	A DLL function was called using an option that is unknown for that function.
VSR_ALREADY_OPEN	An attempt was made to open a standard or custom dictionary that was already open for the current context.
VSR_OPEN_FAILED	An attempt to open a dictionary failed because it could not be found or because it was open for exclusive use by another application.
VSR_CREATE_FAILED	An attempt to create a custom dictionary failed because the path was invalid or because the target directory was not writeable by the application.
VSR_INVALID_DICTIONARY	An attempt to open a standard dictionary failed because it was corrupt or because its format was invalid.
VSR_NO_FLOPPY_ALLOWED	An attempt was made to open a dictionary stored on a floppy diskette. (Dictionaries must be stored on non-removeable media.)
VSR_WRITE_FAILURE	An attempt to create or write to a custom dictionary failed, probably as a result of inadequate disk space.
VSR_READ_FAILURE	An attempt to open or load a dictionary failed, probably because the dictionary was corrupt or because there was a hardware problem.
VSR_BAD_FILE_NAME	An attempt to open a dictionary failed because the file name was invalid.
VSR_NOT_UPDATEABLE	An attempt to make a custom dictionary updateable failed because the dictionary file was read-only.
VSR_DUPLICATE_WORD	An attempt to add a word to a custom dictionary failed because the word was already in the dictionary.
VSR_BAD_CUSTOM	An attempt to open or load a custom dictionary failed because it contained nonASCII characters or because the format of the data was bad.
VSR_IN_EVENT	A CheckText, ResumeCheck, or BeginCheck action failed because an event was already in progress.
VSR_NOT_LOADED	An AddToCustom action failed because the specified dictionary was not loaded.
VSR_REPLACE_ERROR	An automatic replacement or ReplaceLastWord action failed because the text buffer was too short to accept the expanded text.
VSR_CACHE_ERROR	was unable to allocate the minimum cache size.
VSR_INVALID_CONTEXT	The OCX control does not contain a valid context handle, probably because a memory problem existed at the time the control was initialized.
VSR_NOTHING_TO_CHECK	A BeginCheck or ResumeCheck action failed because there was no text

	to spellcheck.
VSR_DIALOG_ERROR	An error prevented a popup dialog box from appearing.
VSR_OUT_OF_MEMORY	There was insufficient memory to complete the operation.
VSR_BAD_INDEX	The dictionary index was out of range.
VSR_OUT_OF_STRING_SPACE	Visual Basic is out of string memory. This could occur anytime you read a property that returns a string.
VSR_IN_DIALOG	An error occurred in the Word Not Found in Dictionary or Spell Options dialog box..
VSR_FILE_EXISTS	There is already an existing file with the name of the file you are trying to create.

Runtime Errors

Visual Basic generates trappable errors for all abnormal conditions. The codes for these errors are the same as their VisualSpeller counterparts but are incremented by VB_OFFSET. By default, this value is 32350. If this range conlicts with another control, you can adjust it by setting the **ErrorOffset** property. The following list shows the constants and values for the Visual Basic OCX runtime error conditions:

Error Constant	Value
VBR_WORD_TOO_LONG	1 + VB_OFFSET
VBR_NO_MISSPELL	2+ VB_OFFSET
VBR_UNKNOWN_OPTION	3+ VB_OFFSET
VBR_ALREADY_OPEN	4+ VB_OFFSET
VBR_OPEN_FAILED	5+ VB_OFFSET
VBR_CREATE_FAILED	6+ VB_OFFSET
VBR_INVALID_DICTIONARY	7+ VB_OFFSET
VBR_NO_FLOPPY_ALLOWED	8+ VB_OFFSET
VBR_WRITE_FAILURE	9+ VB_OFFSET
VBR_READ_FAILURE	10+ VB_OFFSET
VBR_BAD_FILE_NAME	11+ VB_OFFSET
VBR_NOT_UPDATEABLE	12+ VB_OFFSET
VBR_DUPLICATE_WORD	13+ VB_OFFSET
VBR_BAD_CUSTOM	14+ VB_OFFSET
VBR_IN_EVENT	15+ VB_OFFSET
VBR_NOT_LOADED	16+ VB_OFFSET
VBR_REPLACE_ERROR	17+ VB_OFFSET
VBR_CACHE_ERROR	18+ VB_OFFSET
VBR_INVALID_CONTEXT	19+ VB_OFFSET
VBR_NOTHING_TO_CHECK	20+ VB_OFFSET
VBR_DIALOG_ERROR	21+ VB_OFFSET
VBR_OUT_OF_MEMORY	22+ VB_OFFSET
VBR_BAD_INDEX	23+ VB_OFFSET
VBR_OUT_OF_STRING_SPACE	24+ VB_OFFSET
VBR_IN_DIALOG	25+ VB_OFFSET
VBR_FILE_EXISTS	26+ VB_OFFSET

Welcome to VisualSpeller

Welcome to VisualSpeller

Error and Status Conditions

Status Constants

Error Constants

Runtime Errors

OCX Property Reference

Welcome to VisualSpeller

Welcome to VisualSpeller

Welcome to VisualSpeller

Welcome to VisualSpeller

Welcome to VisualSpeller