Acrobat Spelling API Reference

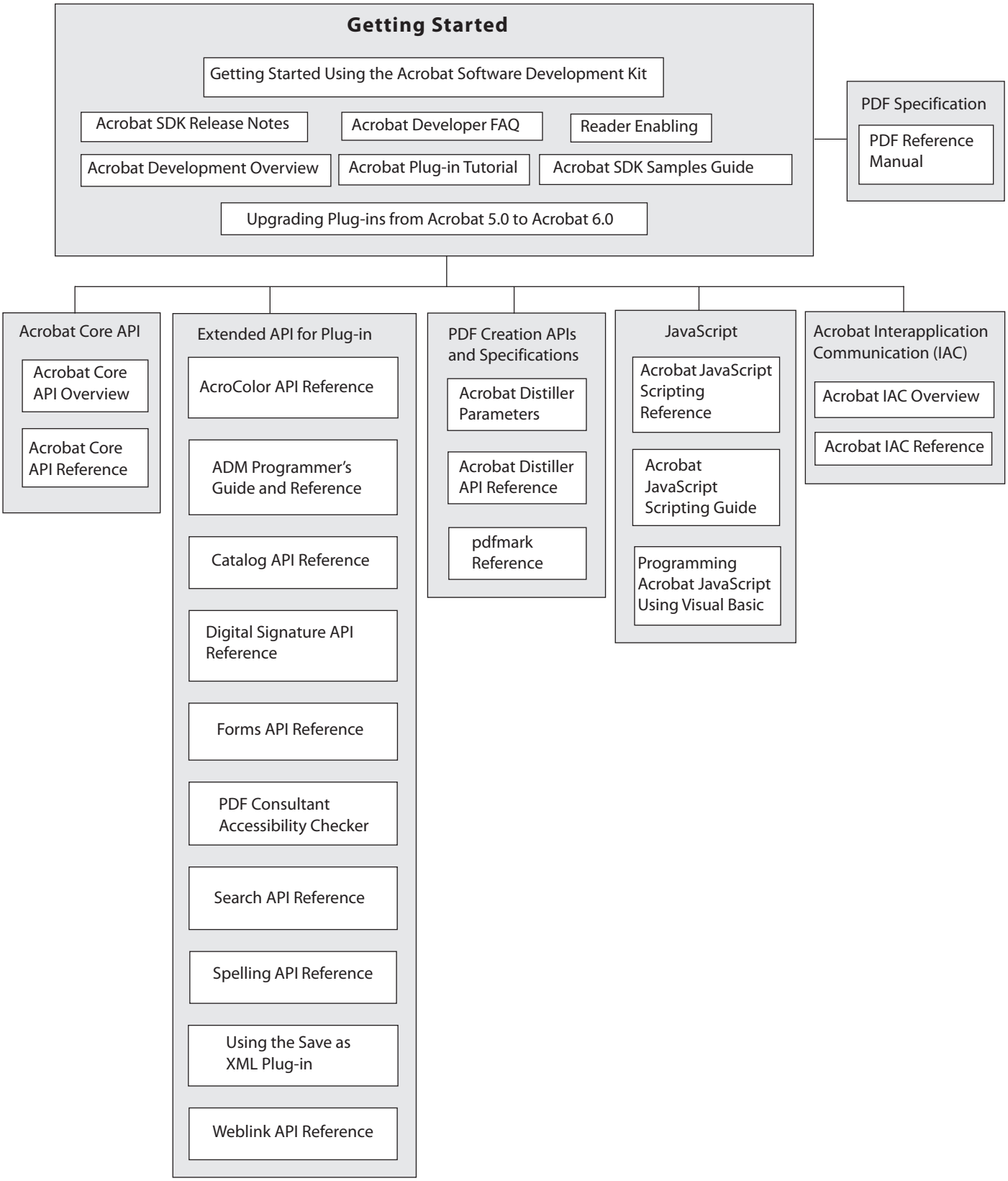
Technical Note #5416

Version : Acrobat 6.0

ADOBE SYSTEMS INCORPORATED
Corporate Headquarters
345 Park Avenue
San Jose, CA 95110-2704
(408) 536-6000
http://partners.adobe.com

May 2003

Acrobat SDK Documentation Roadmap



Contents

Introduction

Acrobat 6.0 provides a Spelling plug-in, which exports a Host Function Table (HFT) implementing a spell-check API for use by plug-in developers. The Spelling API has not changed between Acrobat 5.0 and Acrobat 6.0.

Using the Spelling HFT

To use the Spelling HFT, a plug-in must include the header file **SpellerHFT.h**, which includes **Speller_Sel.h**.

The following is a typical sequence of calls made by a plug-in using the Spelling HFT. During its **importReplaceAndRegister** callback, the plug-in should:

- Import the HFT, using **ASExtensionMgrGetHFT**, and assign the HFT returned by this call to a plug-in-defined global variable named **gSpellerHFT**. The easiest way to do this is to use the **Init_SpellerHFT** macro defined in **SpellerHFT.h**.
- Allocate and initialize one **SpellCheckParam** block for each spelling domain the client will add, if any.
- Add zero or more domains using the **SpellAddDomain** call.

  For example, Acrobat's Forms and Comments plug-ins register domains with Spelling. Spelling calls back into Forms and Comments to get the text for each form field or comment when the user invokes the Spell Check dialog.

During execution, the plug-in should:

- Respond to the following callbacks for each domain:
  - **SCEnableProc** is called by Spelling to ask if this domain has anything that needs to be checked
  - **SCGetTextProc** is called to get a text buffer to be checked.
  - **SCCompletionProc** is called after the user has modified the text buffer.
- The client may call other Spelling HFT services during execution even if it did not add a domain.

During its **unloadCallback**, the plug-in should:

- Remove all spelling domains added during initialization using the **SpellRemoveDomain** call.
- Free all memory associated with **SpellCheckParam** block(s) (**scInBuffer**, **scOutBuffer**, and **scClientData**).
- Free the **SpellCheckParam** block(s).

Text Parameters

Several of the Spelling API methods (**SpellCheck**, **SpellCheckText**, and **SpellCheckWord**) take input strings as parameters, and several methods return strings as output parameters.

Input strings are either big-endian Unicode strings with the bytes **0xFE 0xFF** prepended, or strings with **PDFDocEncoding**. In either case a string is expected to have the appropriate null-termination. If a string is UCS-2 it may have embedded language and country information.

Output text is in big-endian UCS-2 format with the bytes **0xFE 0xFF** prepended. This string can be converted to a host encoded string by using **ASTextFromPDText** and **ASTextGetEncodedCopy**. For example:

```
char **altArray = NULL;
ASInt32 altCount = 0;
ASBool status = SpellCheckWord(acd, cWord, NULL, 0, &altArray,
&altCount);
if (altCount) {
     ASText ast = ASTextFromPDText(altArray[1]);
     char* altWord = ASTextGetEncodedCopy(ast,
     (ASHostEncoding) PDGetHostEncoding() );
}
```

Spelling Declarations

SpellCheckParam

SpellCheckParamPtr

```
typedef SpellCheckParam *SpellCheckParamPtr;

struct SpellCheckParam {
    ASInt16 scVersion;;
    ASInt16 scFlags;
    char scName[kSpellMaxName];
    SCEnableProc scEnableProc;
    SCGetTextProc scGetTextProc;
    SCCompletionProc scCompletionProc;
    AVDoc scAVDoc;
    ASInt32 scPageNum;
    ASInt32 scIndex;
    char * scInBuffer;
    char * scOutBuffer;
    void * scClientData;
}; SpellCheckParam
```

Description

This parameter block is used for communication between the Spelling plug-in and a client plug-in. The client must allocate it, initialize all fields, and pass it to the Spelling plug-in when adding a domain with **SpellAddDomain**. Spelling passes it back to the client when the **SCEnableProc**, **SCGetTextProc**, and **SCCompletionProc** are called.

It is passed to the client's **SCGetTextProc** from the Spelling plug-in when it needs to request a text buffer from the client. After the user has completed the spell check on the **scInBuffer**, this same parameter block is passed to the client's **SCCompletionProc** with the result of the spell check in **scOutBuffer**.

When the **SCGetTextProc** is called, **scPageNum**, **scIndex** and **scAVDoc** are filled in by the Spelling plugin.

The **SCGetTextProc** should fill in **scInBuffer** and clear **scOutBuffer** if it is not already **NULL**.

When the spell check is completed, **SCCompletionProc** is called with **scOutBuffer** if the user made changes to **scInBuffer**.

NOTE: The spelling client is responsible for all memory allocated including the **scOutBuffer** returned from the Spelling plug-in.

The client is responsible for all parameters except where indicated.

Members

scVersion	Version number of this structure (**kSCparam_LATEST_VERSION**)
scFlags	Domain control flags; see **SpellDomainFlags**. Should normally be set to **kSpellFlagAllDomain**.
scName	This name will be returned by **SpellDomainNames**.
scEnableProc	This callback will be invoked with **scAVDoc** filled in before **scGetTextProc** is called. If **false** is returned, the Spelling plug-in will not call **scGetTextProc**.
scGetTextProc	This callback will be invoked when the user clicks the "Start" button on the Spell Check dialog. This procedure will be called by the Spelling plug-in to retrieve an **scIndex** text buffer from the **scPageNum** of **scAVDoc**. This procedure should iterate through all the items on the **scPageNum** of **scAVDoc**.
scCompletionProc	Completion callback for **scInBuffer**. When the user has completed the spell check of **scInBuffer**, this procedure will be called with **scOutBuffer** filled in by the Spelling plugin if the user made changes to **scInBuffer**. */
scAVDoc	*(Passed by Spelling plug-in)* The active **AVDoc** during this spell checking session.
scPageNum	*(Passed by Spelling plug-in)* The page number (zero based) of **scAVDoc**.
scIndex	*(Passed by Spelling plug-in)* The index of this domain item (zero based). The Spelling plug-in will set **scIndex** to zero to start/restart and request the first text buffer from this domain on **scPageNum**. The Spelling plug-in will increment **scIndex** after each call to **scGetTextProc**. The client can increment **scIndex** if desired to track non-sequential domain items.
scInBuffer	The text buffer passed from client to the Spelling plug-in when **scGetTextProc** is called by the Spelling plug-in.

scOutBuffer	*(Passed by Spelling plug-in)* The buffer returned from the Spelling plug-in to **scCompletionProc**. **NOTE:** The client must release this memory.
scClientData	Can be used by the client to store private state data. The client is responsible for allocating memory where necessary.

Header File

Speller_Sel.h

Related Methods

SpellAddDomain

SpellRemoveDomain

SpellDialogResult

```
enum SpellDialogResult
{
    kSpellDone = 0,
    kSpellCompleted,
    kSpellFailed = -1,
    kSpellResultLast;
```

Description

Possible results of the spell check when the **SpellCheck** method is called.

Members

kSpellDone	The user clicked the "Done" button to dismiss the dialog and did not complete the spell check, but may have made changes to the text buffer.
kSpellCompleted	The user completed spell checking of the input buffer, or all words were correct. A new text buffer is returned if corrections where made.
kSpellFailed	The spell check dialog failed due to an internal error

Header File

Speller_Sel.h

Related Methods

SpellCheck

SpellDomainFlags

```
enum SpellDomainFlags
{
    kSpellFlagNone = 0x0000,
    kSpellFlagAllDomain = 0x0001,
    kSpellFlagLast = 0xFFFF;
```

Description

Valid values for the **scFlags** field of the **SpellCheckParam** block. For Acrobat 5 and higher, all clients should set this field to **kSpellFlagAllDomain**. All other bits in this flag are reserved for future use. Spelling clients' private data and flags can be stored in the **scClientData** field.

Members

kSpellFlagNone	Default domain behavior.
kSpellFlagAllDomain	For Acrobat 5 and higher, all registered spelling domains should set this flag. Only the "All" domains are called by spelling when the user clicks the "Start" button on the Spell Check dialog.

Header File

Speller_Sel.h

Spelling Methods

SpellAddDictionary

```
ASBool SpellAddDictionary(char* cName, char *cFile,
ASBool bShow);
```

Description

Adds a dictionary to the list of available dictionaries.

Parameters

cName	The dictionary name to be used in the Spelling Preference Panel. It can be used as one of the **dictionaryArray** elements in the **SpellCheck**, **SpellCheckText**, **SpellCheckWord** and **SpellCountKnownWords** methods. It can be returned as one of the **dictionaryArray** elements in the **SpellDictionaryNames**, **SpellGetDocDictionaryOrder**, **SpellSetDocDictionaryOrder** and **SpellUserDictionaryOrder** methods.
cFile	Device-independent path of the dictionary files. A dictionary actually consists of four files: **DDD*xxxxx*.hyp**, **DDD*xxxxx*.lex**, **DDD*xxxxx*.clx**, and **DDD*xxxxx*.env**. (***xxxxx*** may be any value you like.) The **cFile** parameter must be the device independent path of the **.hyp** file. For example "**/c/temp/testdict/TST.hyp**". Spelling will look in the parent directory of the TST.hyp file for the other three files. All four file names must start with the same unique 3 characters to associate them with each other, and they must end with the dot-three extensions listed above, even on a Macintosh.
bShow	If **true**, Spelling will combine the **cName** parameter with "User: " and show that name in all lists and dialogs. For example if **cName** is "Test", Spelling will add "User: Test" to all lists and dialogs. When **bShow** is false, Spelling will not show this custom dictionary in any lists or dialogs.

Return Value

true if successful; the dictionary is now available for use.

Header File

SpellerHFT.h

Related Methods

SpellDictionaryNames
SpellGetDocDictionaryOrder
SpellRemoveDictionary
SpellSetDocDictionaryOrder
SpellUserDictionaryOrder

SpellAddDomain

```
ASBool SpellAddDomain(SpellCheckParamPtr scp);
```

Description

Adds a spelling domain (search scope) to the Spell Check dialog. This should be called during plug-in initialization for each domain that Spelling should iterate through when the user invokes the Spell Check dialog.

Parameters

scp	A pointer to a parameter block. See **SpellCheckParam** for details.

Return Value

true if the domain name was successfully added, **false** otherwise.

Header File

SpellerHFT.h

Related Methods

SpellRemoveDomain

SpellAddWord

```
ASBool SpellAddWord(char *cWord);
```

Description

Adds a word to the user's dictionary.

Parameters

`cWord`	The word to be added to the user's dictionary. **NOTE:** Internally the Spelling plug-in will scan the user "Not-A-Word" dictionary and remove the word if it is listed there. Otherwise, the word is added to the user dictionary.

Return Value

`true` if successful; the word has been added.

Header File

`SpellerHFT.h`

Related Methods

`SpellRemoveWord`

SpellCheck

```
char* SpellCheck (AVDoc avd, const char* textBuffer,
ASInt16* dialogResult, ASBool bReserved,
char** dictionaryArray, ASInt32 dictionaryCount);
```

Description

Calls the Spelling plug-in to have it check a text buffer and interact with the user. This service will display the Spell Check dialog.

Parameters

avd	The current active document.
textBuffer	The input text buffer to be checked for spelling.
dialogResult	*(Filled by the method)* See enum **SpellDialogResult** for possible dialog results. If **dialogResult** is **kSpellDone**, the client should terminate any domain search loops.
bReserved	Reserved for future use.
dictionaryArray	Optional array of dictionary names to be searched. When provided, this dictionary list overrides the default dictionary search list for this document.
dictionaryCount	Count of dictionary names in **dictionaryArray**.

Return Value

Anew text buffer if the user made any changes to the input text buffer, otherwise **NULL**.

Header File

SpellerHFT.h

Related Methods

SpellCheckRTF
SpellCheckText
SpellCheckWord

SpellCheckRTF

```
ASInt32 SpellCheckRTF(AVDoc avd, void* vReserved,
ASText astPlainText, char** dictionaryArray,
ASInt32 dictionaryCount, SCRTFChangeProc pRTFchangeProc,
void* vClientData);
```

Description

Calls the Spelling plug-in to have it check a text object and receive a callback for each change as the user interacts with the Spell Check dialog. This service will display the Spell Check dialog.

Parameters

avd	The current active document.
vReserved	Reserved for future use.
astPlainText	The text object to be checked for spelling.
dictionaryArray	Optional array of dictionary names to be searched. When provided, this dictionary list overrides the default dictionary search list for this document.
dictionaryCount	Count of dictionary names in **dictionaryArray**.
pRTFchangeProc	The callback procedure to call with each user change as it occurs.
vClientData	A pointer to data to pass to the callback.

Return Value

A **SpellDialogResult** value.

Header File

SpellerHFT.h

Related Methods

SpellCheck
SpellCheckText
SpellCheckWord

SpellCheckText

```
ASBool SpellCheckText(AVDoc avd, const char* textBuffer,
ASUns32* startOffset, ASUns32* endOffset,
char** dictionaryArray, ASInt32 dictionaryCount);
```

Description

Checks the spelling of the words in a text buffer starting at **startOffset** and ending at **endOffset**.

NOTE: It does not display the Spell Check dialog. Typically you would call this function when you want to silently check the spelling of a text buffer. AcroForms uses this call to underline unknown words in a form text field.

Parameters

avd	The document to check.
textBuffer	Text buffer to be checked for spelling.
startOffset	Set this value to the starting offset (zero based) in **textBuffer** where Spelling should start searching for misspelled words. (Pass zero to check all text in the buffer.) The method sets this value to the offset of the first misspelled word found.
endOffset	Set this value to the ending offset (zero based) in **textBuffer** where Spelling should stop searching for misspelled words. (Pass the length of **textBuffer** to check to the end of the buffer.) The method sets this value to the offset of the end of the first misspelled word found.
dictionaryArray	Optional array of dictionary names to be searched. When provided, this dictionary list overrides the default dictionary search list for the current active document.
dictionaryCount	Count of dictionary names in **dictionaryArray**

Return Value

Returns **true** if all words in the buffer were in the current dictionaries.

Returns **false** if a misspelled word was found and returns the **startOffset** and **endOffset** of the first misspelled word found in the text buffer.

Header File

SpellerHFT.h

Related Methods

SpellCheck
SpellCheckRTF
SpellCheckWord

SpellCheckWord

```
ASBool SpellCheckWord(AVDoc avd, const char* cWord,
char** dictionaryArray, ASInt32 dictionaryCount,
char*** alternativeArrayPtr, ASInt32 *alternativeCount);
```

Description

Calls the Spelling plug-in to have it check a word.

NOTE: It does not display the Spell Check dialog. Typically you would call this function when you want to silently check the spelling of a word.

Parameters

avd	The current active document.
cWord	A string containing the word to be checked for spelling.
dictionaryArray	Optional array of dictionary names to be searched. When provided, this dictionary list overrides the default dictionary search list for the current active document.
dictionaryCount	Count of dictionary names in **dictionaryArray**
alternativeArrayPtr	*(Filled by the method)* Optional pointer for an array of alternative spellings of an incorrect word if any. **NOTE:** The caller is responsible for freeing this memory.
alternativeCount	Count of alternative words in **alternativeArray**

Return Value

true if the word in the buffer is correct, **false** if the word is unknown.

Header File

SpellerHFT.h

Related Methods

SpellCheck
SpellCheckRTF
SpellCheckText

SpellCountKnownWords

```
ASInt32 SpellCountKnownWords(const char* textBuffer, ASInt32
dictionaryCount, char** dictionaryArray, ASInt32*
counterArray);
```

Description

Calls the Spelling plug-in to count the words in a text buffer that are contained in each of a set of dictionaries. Typically you would call this function to attempt to discover the language of a text buffer.

NOTE: It does not display the Spell Check dialog.

Parameters

textBuffer	The text buffer.
dictionaryCount	Count of dictionary names in **dictionaryArray** and counters in **counterArray**
dictionaryArray	Array of dictionary names to be searched
counterArray	An array of counters. For each dictionary in **dictionaryArray**, the Spelling plug-in increments the corresponding counter in this array when a word from the text buffer is found in that dictionary.

NOTE: All four of these parameters are required.

Return Value

The index of the dictionary with the highest correct count; if more than one dictionary has the highest correct count, the index of the first one encountered. -1 on error.

Header File

SpellerHFT.h

Related Methods

SpellCheckWord

SpellCustomDictionaryClose

```
ASBool SpellCustomDictionaryClose(ASText astName);
```

Description

Removes a custom user dictionary from the dictionary search order, but does not delete the dictionary file. Generally used by a PDF Form.

Parameters

astName	The dictionary name as an **ASText** object; must be the same name that was passed to **SpellCustomDictionaryCreate** or **SpellCustomDictionaryOpen**.

Return Value

true if successful; the dictionary is no longer available for use.

Header File

SpellerHFT.h

Related Methods

SpellCustomDictionaryCreate
SpellCustomDictionaryDelete
SpellCustomDictionaryExport
SpellCustomDictionaryOpen

SpellCustomDictionaryCreate

```
ASBool SpellCustomDictionaryCreate(ASText astName,
ASPathName folderPath);
```

Description

Creates a new custom user dictionary and adds it to the list of available dictionaries. You can add words to the new dictionary using **SpellAddWord**. When it is complete, you can export it using **SpellCustomDictionaryExport**.

This dictionary can be passed or returned as one of the **dictionaryArray** elements in Spelling functions.

A PDF Form can add a custom dictionary to its dictionary search order by opening it with **SpellCustomDictionaryOpen**.

Parameters

astName	The dictionary name to be used in the Spelling Preference Panel, as an **ASText** object.
folderPath	The dictionary folder path, as an **ASPathName** object. If the user does not have read/write permission in this folder, the method fails.

Return Value

true if successful; the dictionary is now available for use.

Header File

SpellerHFT.h

Related Methods

SpellCustomDictionaryClose
SpellCustomDictionaryDelete
SpellCustomDictionaryExport
SpellCustomDictionaryOpen

SpellCustomDictionaryDelete

```
ASBool SpellCustomDictionaryDelete(ASText astName);
```

Description

Deletes a custom user dictionary object, and deletes its file from disk.

Parameters

astName	The dictionary name as an **ASText** object; must be the same name that was passed to the creation function.

Return Value

true if successful; the dictionary is no longer available for use.

false on failure; the method fails if the user does not have read/write permission for the dictionary file.

Header File

SpellerHFT.h

Related Methods

SpellCustomDictionaryClose
SpellCustomDictionaryCreate
SpellCustomDictionaryExport
SpellCustomDictionaryOpen

SpellCustomDictionaryExport

```
ASBool SpellCustomDictionaryExport(ASText astName,
ASPathName folderPath, void* reserved);
```

Description

Exports an open custom user dictionary to the specified folder.

Parameters

astName	The dictionary name, as an **ASText** object. This becomes the file name of the exported dictionary file.
folderPath	The folder path where the dictionary file will be created, as an **ASPathName** object. If the user does not have read/write permission in this folder, the method fails.
reserved	Reserved for future use.

Return Value

true if successful; the dictionary has been exported.

Header File

SpellerHFT.h

Related Methods

SpellCustomDictionaryClose
SpellCustomDictionaryCreate
SpellCustomDictionaryDelete
SpellCustomDictionaryOpen

SpellCustomDictionaryOpen

```
ASBool SpellCustomDictionaryOpen(ASText astName,
ASPathName folderPath, ASBool bShow);
```

Description

Opens a custom user dictionary and adds it to the dictionary search order. Generally used by a PDF Form.

This dictionary can be passed or returned as one of the **dictionaryArray** elements in Spelling functions.

Parameters

astName	The dictionary name to be used in the Spelling Preference Panel, as an **ASText** object.
folderPath	The dictionary folder path, as an **ASPathName** object. If the user does not have read/write permission in this folder, the method fails.
bShow	When **true**, Spelling combines **astName** with "User: " and shows that name in all lists and menus. For example if **astName** is "Test", Spelling adds "User: Test" to all lists and menus. When **false**, Spelling does not show this custom dictionary in any lists or menus.

Return Value

true if successful; the dictionary is now available for use.

Header File

SpellerHFT.h

Related Methods

SpellCustomDictionaryClose
SpellCustomDictionaryCreate
SpellCustomDictionaryDelete
SpellCustomDictionaryExport

SpellDictionaryNames

```
ASBool SpellDictionaryNames(char*** dictionaryArrayPtr,
ASInt32 *dictionaryCount);
```

Description

Returns an array of the currently available dictionary names.

Parameters

dictionaryArrayPtr	*(Filled by the method)* A pointer to an array of the dictionary names allocated and returned by the Spelling plug-in. **NOTE:** The caller is responsible for freeing this memory.
dictionaryCount	*(Filled by the method)* A pointer to the count of names in the array, returned by the Spelling plug-in.

Return Value

true if successful; dictionary names if any are listed in the output **dictionaryArray**.

Header File

SpellerHFT.h

Related Methods

SpellAddDictionary
SpellRemoveDictionary

SpellDomainNames

```
ASBool SpellDomainNames(char*** domainArrayPtr, ASInt32
*domainCount);
```

Description

Returns an array of the current domain names.

Parameters

domainArrayPtr	*(Filled by the method)* An array of the domain names, returned by the Spelling plug-in **NOTE:** The caller is responsible for freeing this memory.
domainCount	*(Filled by the method)* The count of names in the array, returned by the Spelling plug-in.

Return Value

true if successful; domain names if any are listed in the output **domainArray**.

Header File

SpellerHFT.h

Related Methods

SpellAddDomain
SpellRemoveDomain

SpellGetDocDictionaryOrder

```
ASBool SpellGetDocDictionaryOrder(AVDoc avd, char***
dictionaryArrayPtr, ASInt32 *dictionaryCount);
```

Description

Returns the document's dictionary search order, if any. If this array is **NULL**, Spelling will use the order defined by the user in the Spelling Preferences panel. Spelling will search dictionaries in the document order first followed by the user's order.

Parameters

avd	The document.
dictionaryArrayPtr	*(Filled by the method)* An array of the dictionary names, allocated and returned by the Spelling plug-in. The order in the list is the dictionary search order for this document. **NOTE:** The caller is responsible for freeing this memory.
dictionaryCount	*(Filled by the method)* The spelling plugin will return the count of names in the array

Return Value

true if successful; dictionary search order names are listed in the output **dictionaryArray**.

Header File

SpellerHFT.h

Related Methods

SpellUserDictionaryOrder
SpellSetDocDictionaryOrder

SpellGetDocLanguageOrder
SpellLanguages
SpellSetDocLanguageOrder
SpellUserLanguageOrder

SpellGetDocLanguageOrder

```
ASBool SpellGetDocLanguageOrder (AVDoc avd,
char*** languageArrayPtr, ASInt32 *languageCount);
```

Description

Returns the document's dictionary search order, if any, as an array of ISO 639-2 and 3166 language codes, allowing you to identify a dictionary by language rather than by name.

If this array is **NULL**, Spelling uses the order defined by the user in the Spelling Preferences panel. Spelling searches dictionaries in the document's order first, followed by the user's order.

Parameters

avd	The document.
languageArrayPtr	*(Filled by the method)* An array of the language codes, allocated and returned by the Spelling plug-in. The order in the list is the dictionary search order for this document. **NOTE:** The caller is responsible for freeing this memory.
languageCount	*(Filled by the method)* The spelling plugin will return the count of language codes in the array

Return Value

true if successful.

Header File

Speller_Sel.h

Related Methods

SpellLanguages
SpellSetDocLanguageOrder
SpellUserLanguageOrder

SpellGetDocDictionaryOrder
SpellSetDocDictionaryOrder
SpellUserDictionaryOrder

SpellGetNextWord

```
ASText SpellGetNextWord (ASConstText inBuffer,
ASInt32* nStart, ASInt32* nEnd, ASBool bFilter);
```

Description

Scans a text buffer and returns the next word.

Parameters

inBuffer	The text buffer.
nStart	*(Filled by the method)* On input, the character offset in the buffer at which to start the search. On return, the offset to the start of the word, if found.
nEnd	*(Filled by the method)* On input, the character offset at which to terminate the scan, or zero to scan to the end of the text buffer. On return, the offset to the character past the last character of the word found. To scan for the next word, set **nStart** to this new offset.
bFilter	If **true**, apply standard ignore filters to the search, ignoring the following: • single character words • long words (48 characters max) • words that have digits • all UPPERCASE • roman numerals • words that have non-roman unicode (CJK) characters

Return Value

The next word as an **ASText** object.

Header File

Speller_Sel.h

Related Methods

SpellAddWord
SpellRemoveWord

SpellIgnoreAll

```
ASBool SpellIgnoreAll(AVDoc avd, char *cWord);
```

Description

Ignore all occurrences of a word in a document when spell-checking.

Parameters

avd	The document in which to ignore the word.
cWord	The word to ignore. A NULL-terminated string as **PDText**, which is either a big-endian Unicode string prepended with the bytes **0xFE 0xFF**, or a string with **PDFDocEncoding**.

Return Value

true if successful; the word will be ignored.

Header File

`Speller_Sel.h`

Related Methods

SpellAddWord
SpellRemoveWord

SpellLanguages

```
ASBool SpellLanguages( char*** languageArrayPtr,
ASInt32* languageCount);
```

Description

Returns an array of ISO 639-2 and 3166 language codes that identify the currently available dictionaries.

Parameters

languageArrayPtr	*(Filled by the method)* Spelling allocates and returns a **char*** array of the language codes. The caller is responsible for freeing the array and its elements.
languageCount	*(Filled by the method)* A pointer to the number of elements in the **languageArrayPtr** array.

Return Value

true if successful.

Header File

Speller_Sel.h

Related Methods

SpellGetDocLanguageOrder
SpellSetDocLanguageOrder
SpellUserLanguageOrder

SpellGetDocDictionaryOrder
SpellSetDocDictionaryOrder
SpellUserDictionaryOrder

Example

```
char **langArray = NULL;
ASInt32 langCount = 0;
ASBool status = SpellLanguages(&langArray, &langCount);
if (langCount) {
  ASText ast = ASTextFromPDText(langArray[0]);
  char* langName = ASTextGetEncodedCopy(ast,
                      ASHostEncoding(PDGetHostEncoding()));
}
// don't forget to free ast, langName, each element of langArray
// and the langArray itself.
```

SpellRemoveDictionary

```
ASBool SpellRemoveDictionary(char* cName);
```

Description

Removes a user dictionary that was added via **SpellAddDictionary**.

Parameters

cName	The dictionary name; must be the same name that was passed to **SpellAddDictionary**.

Return Value

true if successful; the dictionary is no longer available for use.

Header File

SpellerHFT.h

Related Methods

SpellAddDictionary

SpellRemoveDomain

```
ASBool SpellRemoveDomain(SpellCheckParamPtr scp);
```

Description

Removes a spelling domain (search scope) from the Spell Check dialog that was added with the **SpellAddDomain** call. This should be called during plug-in termination for each domain that was added by a call to **SpellAddDomain**. After calling the function, the client should free all memory associated with the **SpellCheckParam** block (**scInBuffer**, **scOutBuffer**, and **scClientData**) and then free the **SpellCheckParam** block.

Parameters

scp	A pointer to a parameter block. See **SpellCheckParam** for details.

Return Value

true if successful; domain name was removed; **false** if the domain was already removed or never added.

Header File

SpellerHFT.h

Related Methods

SpellAddDomain

SpellRemoveWord

```
ASBool SpellRemoveWord(char *cWord);
```

Description

Removes a word from the user's dictionary.

Parameters

`cWord`	The word to be removed from the user's dictionary. **NOTE:** Internally the Spelling plug-in will scan the user dictionary and remove the previously added word if it is there. Otherwise the word will be added to the user's "Not-A-Word" dictionary.

Return Value

`true` if successful; the word has been removed.

Header File

`SpellerHFT.h`

Related Methods

`SpellAddWord`

SpellSetDocDictionaryOrder

```
ASBool SpellSetDocDictionaryOrder(AVDoc avd, char**
dictionaryArray, ASInt32 dictionaryCount)
```

Description

Sets the document's dictionary search order. If this array is **NULL**, Spelling uses the order defined by the user in the Spelling Preferences panel. Spelling searches dictionaries in the document order first, followed by the user's order.

Parameters

avd	The document.
dictionaryArray	An array of ordered dictionary names to be searched before searching the list specified by the user in Spelling Preferences Panel. Use **SpellDictionaryNames** to obtain a list of the currently available dictionaries.
dictionaryCount	The number of dictionaries in the **dictionaryArray**. Pass zero to clear the document order list.

Return Value

true if successful; the dictionary search order for this document has been set.

Header File

SpellerHFT.h

Related Methods

SpellGetDocDictionaryOrder
SpellUserDictionaryOrder

SpellGetDocLanguageOrder
SpellLanguages
SpellSetDocLanguageOrder
SpellUserLanguageOrder

SpellSetDocLanguageOrder

```
ASBool SpellSetDocLanguageOrder (AVDoc avd,
char** languageArrayPtr, ASInt32 languageCount);
```

Description

Sets the document's dictionary search order from an array of ISO 639-2 and 3166 language codes, allowing you to identify a dictionary by language rather than by name. Spelling searches dictionaries in the document's order first, followed by the user's order.

Parameters

avd	The document.
languageArrayPtr	A **char*** array of ordered language codes for dictionaries to be searched before searching the list specified by the user in the Spelling Preferences Panel.
languageCount	The number of language codes in the array. Pass zero to clear the document's dictionary order list.

Return Value

true if successful.

Header File

Speller_Sel.h

Related Methods

SpellGetDocLanguageOrder
SpellLanguages
SpellUserLanguageOrder

SpellGetDocDictionaryOrder
SpellSetDocDictionaryOrder
SpellUserDictionaryOrder

SpellUserDictionaryOrder

```
ASBool SpellUserDictionaryOrder(char*** dictionaryArrayPtr,
ASInt32 *dictionaryCount);
```

Description

Returns the user's dictionary search order from the Spelling Preference Panel.

Parameters

`dictionaryArrayPtr`	*(Filled by the method)* An array of dictionary names, allocated and returned by the Spelling plug-in. The array is ordered by the dictionary search order from the Spelling Preferences panel. **NOTE:** The caller is responsible for freeing this memory.
`dictionaryCount`	*(Filled by the method)* The count of names in the array, returned by the Spelling plug-in.

Return Value

`true` if successful.

Header File

`SpellerHFT.h`

Related Methods

`SpellGetDocDictionaryOrder`
`SpellSetDocDictionaryOrder`

`SpellGetDocLanguageOrder`
`SpellLanguages`
`SpellSetDocLanguageOrder`
`SpellUserLanguageOrder`

SpellUserLanguageOrder

```
ASBool SpellUserLanguageOrder (char*** languageArrayPtr,
ASInt32 *languageCount);
```

Description

Returns the user's dictionary search order, if any, as an array of ISO 639-2 and 3166 language codes, allowing you to identify a dictionary by language rather than by name.

This is the order defined by the user in the Spelling Preferences panel. Spelling searches dictionaries in the document's order first, followed by the user's order.

Parameters

`languageArrayPtr`	*(Filled by the method)* An array of the language codes, allocated and returned by the Spelling plug-in. The order in the list is the dictionary search order for this document. **NOTE:** The caller is responsible for freeing this memory.
`languageCount`	*(Filled by the method)* Spelling returns the count of language codes in the array

Return Value

`true` if successful.

Header File

`Speller_Sel.h`

Related Methods

`SpellGetDocLanguageOrder`
`SpellLanguages`
`SpellSetDocLanguageOrder`

`SpellGetDocDictionaryOrder`
`SpellSetDocDictionaryOrder`
`SpellUserDictionaryOrder`

SpellUserWords

```
ASBool SpellUserWords(ASBool bAdded, char*** wordArrayPtr,
ASInt32 *wordCount);
```

Description

Returns an array of words from the user's "is a word" dictionary, or from the user's "not a word" dictionary.

Parameters

bAdded	If **true**, the list of added words is returned. if **false**, the list of removed words is returned.
wordArrayPtr	*(Filled by the method)* An array of the added/removed words, allocated and returned by the Spelling plug-in. **NOTE:** The caller is responsible for freeing this memory.
wordCount	*(Filled by the method)* The count of words in the array, returned by the Spelling plug-in.

Return Value

true if successful; the user words if any are listed in the output **wordArray**.

Header File

SpellerHFT.h

Spelling Callbacks

SCCompletionProc

```
ACCB1 ASBool ACCB2 SCCompletionProc (SpellCheckParamPtr scp);
```

Description

An **SCCompletionProc** must be provided for each domain a plug-in registers with **SpellAddDomain**. It is called by the Spelling plug-in when the user has completed the spell check.

Parameters

scp	The **SpellCheckParam** parameter block (which the client set up in **SpellAddDomain**) passed to the client by the Spelling plug-in after the spell check of **scInBuffer** is complete. This procedure will be called with **scOutBuffer** filled in by the Spelling plug-in if the user made changes to scInBuffer. (**scOutBuffer** could be **NULL**).

Return Value

true if the changes were successfully committed.

NOTE: In the current version of the Spelling plug-in, this value is ignored.

Header File

Speller_Sel.h

Related Callbacks

SCEnableProc
SCGetTextProc

Related Methods

SpellAddDomain
SpellRemoveDomain

SCEnableProc

```
ACCB1 ASBool ACCB2 SCEnableProc (SpellCheckParamPtr scp);
```

Description

Called by the Spelling plug-in to determine whether the Spelling menu items and toolbar button should be enabled. An **SCEnableProc** must be provided for each domain a plug-in registers with **SpellAddDomain**.

Parameters

scp	The Spelling plug-in passes this **SpellCheckParam** parameter block (which the client set up in **SpellAddDomain**) to the **SCEnableProc** after the spell check is complete. The result of the spell check is in **scOutBuffer**.

Return Value

Should return **true** if spell checking can be performed on the **scName** domain.

Header File

Speller_Sel.h

Related Callbacks

SCCompletionProc
SCGetTextProc

Related Methods

SpellAddDomain
SpellRemoveDomain

SCGetTextProc

```
ACCB1 ASBool ACCB2 SCGetTextProc (SpellCheckParamPtr scp);
```

Description

This procedure is called by Spelling to request a text buffer to be checked by the user in the Spelling Dialog.

Parameters

scp	The Spelling plug-in passes this **SpellCheckParam** parameter block, which the client set up in **SpellAddDomain**, to the **SCGetTextProc** to request a text buffer. The Spelling plug-in will fill in the **scAVDoc**, **scPageNum**, and **scIndex** members. **SCGetTextProc** should return a text buffer in **scInBuffer**.

Return Value

This callback should pass back a text buffer to be checked in the **scInBuffer** member of **scp**, and return **true**. Spelling will call the client's **SCCompletionProc** after the user has processed this buffer.

If there is no more data to be checked on this page, **SCGetTextProc** should return **false** and set **scInBuffer** to **NULL**.

Header File

Speller_Sel.h

Related Callbacks

SCCompletionProc
SCEnableProc

Related Methods

SpellAddDomain
SpellRemoveDomain

SCRTFChangeProc

```
ACCB1 ASBool ACCB2 SCRTFChangeProc (void* vReserved,
ASConstText astNewText, ASInt32 nStartIndex,
ASInt32 nEndIndex, void *vData);
```

Description

Handler called by **SpellCheckRTF** each time the user makes a change using the Spell Check dialog. This allows the caller of SpellCheckRTF to track text changes to a rich text buffer and overlay the plain text change onto the rich text spans to perserve formatting.

Parameters

vReserved	Reserved for future use.
astNewText	The text object containing the new plain text of the change.
nStartIndex	The character offset of the change start from the start of the current plain text buffer in **astNewText**.
nEndIndex	The character offset of the change end from the start of the current plain text buffer in **astNewText**.
vData	The client data passed to the calling function.

Return Value

true to continue the dialog with the user, **false** to terminate.

Header File

Speller_Sel.h

Related Methods

SpellCheckRTF