Language Analysis Manager Reference

Internationalization > Localization

2003-04-01

Contents

Language Analysis Manager Reference 5

Overview 5
Functions by Task 5
- Getting The Library Version 5
- Handling Environments 5
- Opening and Closing Contexts 6
- Managing Dictionaries 6
- Analyzing Text 6

Data Types 6
- HomographAccent 6
- HomographDicInfoRec 7
- HomographWeight 7
- JapanesePartOfSpeech 7
- LAContextRef 7
- LAEnvironmentRef 8
- LAHomograph 8
- LAMorpheme 8
- LAMorphemeBundle 9
- LAMorphemePath 9
- LAMorphemeRec 10
- LAMorphemesArray 11
- LAPropertyKey 11
- LAPropertyType 12
- MorphemePartOfSpeech 12
- MorphemeTextRange 12

Constants 12
- File Creator Constants 12
- Analysis Engine Keywords 13
- Analysis Results Constants 13
- Morpheme Key Values 13
- All Morphemes Constant 13
- Leading and Trailing Constants 13
- Converting Mask 14
- Morphemes Array Version 14
- Conjugation Constants 14
- Parts of Speech Constants 14
- Parts of Speech Masks 15
- Engine Limitations 16
- Analysis Engine Type Definitions 16
- Morpheme Types 16
- Morpheme Type Analysis Constants 16

Default Environment Names 16

Result Codes 17

Appendix A **Deprecated Language Analysis Manager Functions 19**

Deprecated in Mac OS X v10.5 19

LAAddNewWord 19

LACloseAnalysisContext 19

LACloseDictionary 20

LAContinuousMorphemeAnalysis 20

LACreateCustomEnvironment 22

LADeleteCustomEnvironment 23

LAGetEnvironmentList 23

LAGetEnvironmentName 24

LAGetEnvironmentRef 25

LAGetMorphemes 25

LALibraryVersion 26

LAListAvailableDictionaries 26

LAMorphemeAnalysis 27

LAOpenAnalysisContext 29

LAOpenDictionary 29

LAResetAnalysis 30

LAShiftMorphemes 31

LATextToMorphemes 32

Document Revision History 35

Index 37

Language Analysis Manager Reference

Framework: ApplicationServices/ApplicationServices.h

Declared in LanguageAnalysis.h

Overview

The Language Analysis Manager application programming interface (API) is a shared library designed to analyze morphemes in text. It is a general-purpose API that does not rely on languages, algorithms of morpheme analysis, or their applications. Language Analysis Manager is not a framework for creating International-aware applications. To make your applications work correctly with various languages, you can use APIs such as Script Manager and Text Utilities.

The Language Analysis Manager (LAM) provides your application with morphological analysis capability, and is designed to work with a language analysis engine. Using the Language Analysis Manager, your application can manage an analysis engine and create environments and contexts in which morpheme analysis can occur. This version of the Language Analysis Manager works only with a Japanese analysis engine.

Functions by Task

Getting The Library Version

`LALibraryVersion` (page 26) Deprecated in Mac OS X v10.5
    Returns the version of the Language Analysis Manager installed.

Handling Environments

`LACreateCustomEnvironment` (page 22) Deprecated in Mac OS X v10.5
    Creates a new environment with the specified name.

`LADeleteCustomEnvironment` (page 23) Deprecated in Mac OS X v10.5
    Disposes of a reference to a custom language analysis environment.

`LAGetEnvironmentList` (page 23) Deprecated in Mac OS X v10.5
    Obtains a list of the available language analysis environments.

`LAGetEnvironmentName` (page 24) Deprecated in Mac OS X v10.5
    Obtains the name of an environment.

`LAGetEnvironmentRef` (page 25) Deprecated in Mac OS X v10.5
    Obtains the language analysis environment reference associated with an environment name

Opening and Closing Contexts

`LACloseAnalysisContext` (page 19) Deprecated in Mac OS X v10.5
  Closes the specified language analysis context.

`LAOpenAnalysisContext` (page 29) Deprecated in Mac OS X v10.5
  Creates a language analysis context from a specified language analysis environment.

Managing Dictionaries

`LAAddNewWord` (page 19) Deprecated in Mac OS X v10.5
  Adds a new word to a dictionary.

`LACloseDictionary` (page 20) Deprecated in Mac OS X v10.5
  Closes a dictionary in the specified environment.

`LAListAvailableDictionaries` (page 26) Deprecated in Mac OS X v10.5
  Obtains the number of dictionaries available in a specified environment.

`LAOpenDictionary` (page 29) Deprecated in Mac OS X v10.5
  Opens a dictionary for the specified environment.

Analyzing Text

`LAContinuousMorphemeAnalysis` (page 20) Deprecated in Mac OS X v10.5
  Performs a continuous morphological analysis of Unicode text.

`LAGetMorphemes` (page 25) Deprecated in Mac OS X v10.5
  Reads the results of a continuous morpheme analysis.

`LAMorphemeAnalysis` (page 27) Deprecated in Mac OS X v10.5
  Performs a morphological analysis of the specified Unicode text.

`LAResetAnalysis` (page 30) Deprecated in Mac OS X v10.5
  Clears the internal status of the analysis context.

`LAShiftMorphemes` (page 31) Deprecated in Mac OS X v10.5
  Shifts the read out of continuous morpheme analysis.

`LATextToMorphemes` (page 32) Deprecated in Mac OS X v10.5
  Performs a morphological analysis of the specified text.

Data Types

HomographAccent

Defines a data type for a homographic accent.

```
typedef UInt8 HomographAccent;
```

Availability
Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

HomographDicInfoRec

Contains dictionary information for a homograph.

```
struct HomographDicInfoRec {
    DCMDictionaryID dictionaryID;
    DCMUniqueID uniqueID;
};
typedef struct HomographDicInfoRec HomographDicInfoRec;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

HomographWeight

Defines a data type for a homographic weighting value.

```
typedef UInt16 HomographWeight;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

JapanesePartOfSpeech

Defines a data type for a Japanese part of speech.

```
typedef MorphemePartOfSpeech JapanesePartOfSpeech;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAContextRef

A reference to an opaque language analysis context.

```
typedef struct OpaqueLAContextRef * LAContextRef;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAEnvironmentRef

A reference to an opaque language analysis environment structure.

```
typedef struct OpaqueLAEnvironmentRef * LAEnvironmentRef;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAHomograph

Defines a data types for a homograph node.

```
typedef AERecord LAHomograph;
```

Discussion
The Apple event record (`AERecord`) is the data type upon which many Language Analysis Manager data types are based. A homograph node is the minimum unit of analysis and is representative of an individual language. Typically a homograph node corresponds to one word obtained from the dictionary.

Homograph nodes include the character string which represents this language, but the content varies according to the type of analysis stipulated in the analysis environment. Depending on the type of environment, additional information may be included for a specific language.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAMorpheme

Defines a data type for a morpheme node.

```
typedef AERecord LAMorpheme;
```

Discussion
The Apple event record (`AERecord`) is the data type upon which many Language Analysis Manager data types are based. Morpheme nodes display the language of a specific part of speech for a particular text character strings, and have corresponding character string range, part of speech and homograph nodes within text character strings as attributes.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAMorphemeBundle

Defines a data type for a morpheme bundle.

```
typedef AERecord LAMorphemeBundle;
```

Discussion
The Apple event record (`AERecord`) is the data type upon which many Language Analysis Manager data types are based. Morpheme bundles are a collection of different solutions to morpheme analysis on one character string. The "different solutions" referred to here means that two solutions have different morpheme delimiters, or the same morpheme delimiters, but the parts of speech are not the same. Morpheme bundles have each of these different solutions in the from of a morpheme path. Morpheme bundles normally have multiple paths in the "most likely" order.

Within morpheme bundles, morpheme paths do not directly include morpheme nodes. Morpheme bundles have a list of morpheme nodes as one of their attributes distinct from the morpheme path, and morpheme paths have an index to that list. In this way, it is possible to share a morpheme node from one or more paths by indirectly indicating the morpheme node. In most cases, multiple paths within one bundle resemble one another to some extent, and multiple paths may be deemed to have the same morpheme node. One morpheme node may include many homograph nodes, making it bigger, so a mechanism such as this which allows sharing is important in maintaining a small data size.

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAMorphemePath

Defines a data type for a morpheme path.

```
typedef AERecord LAMorphemePath;
```

Discussion

The Apple event record (`AERecord`) is the data type upon which many Language Analysis Manager data types are based. A morpheme path defines a single solution for the analysis of a morpheme. The path has an individual morphme delmiinter and part of speech.

There are two types of variation of morpheme paths which have a different way of holding the lower-place morpheme nodes, and in some cases they are used for different purposes. One is the morpheme path within the morpheme bundle mentioned earlier, where the path does not directly include morpheme nodes.

The other form is the morpheme path which can be used alone, and in this case, it is more convenient for it to be closed in that unit. If an application changes the operation of a morpheme node, the morpheme node must not be being shared. Therefore, for single morpheme paths, morpheme nodes are directly included in the morpheme path.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In

`LanguageAnalysis.h`

LAMorphemeRec

Contains results of the analayis of one morpheme.

```
struct LAMorphemeRec {
    ByteCount sourceTextLength;
    LogicalAddress sourceTextPtr;
    ByteCount morphemeTextLength;
    LogicalAddress morphemeTextPtr;
    UInt32 partOfSpeech;
};
typedef struct LAMorphemeRec LAMorphemeRec;
```

Fields

`sourceTextLength`

The length of the source text for this morpheme.

`sourceTextPtr`

A pointer to the source text.

`morphemeTextLength`

The length of the result text for this morpheme.

`morphemeTextPtr`

A pointer to the result text.

`partOfSpeech`

The part of speech of this morpheme.

Discussion

This structure is an entry in the `LAMorphemesArray` data structure.

Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAMorphemesArray

Contains the results of high-level morphological analysis.

```
struct LAMorphemesArray {
    ItemCount morphemesCount;
    ByteCount processedTextLength;
    ByteCount morphemesTextLength;
    LAMorphemeRec morphemes[1];
};
typedef struct LAMorphemesArray LAMorphemesArray;
typedef LAMorphemesArray * LAMorphemesArrayPtr;
```

Fields

`morphemesCount`
    The number of morphemes included.

`processedTextLength`
    The processed source character length.

`morphemesTextLength`
    The overall length of the result string.

`morphemes`
    An array of morpheme records.

Discussion
When you perform high-level analysis, you can analyze stream-format text and obtain the results as an array of morpeme information.

Availability
Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAPropertyKey

Defines a data type for a language analysis property key.

```
typedef AEKeyword LAPropertyKey;
```

Availability
Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAPropertyType

Defines a data type for a language analysis property type.

```
typedef DescType LAPropertyType;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

MorphemePartOfSpeech

Defines a data type for a morpheme part of speech.

```
typedef UInt32 MorphemePartOfSpeech;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

MorphemeTextRange

Contains a range of text associated with a morpheme.

```
struct MorphemeTextRange {
    UInt32 sourceOffset;
    UInt32 length;
};
typedef struct MorphemeTextRange MorphemeTextRange;
```

Availability
Available in Mac OS X v10.0 and later.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

Constants

File Creator Constants

Specify file creator for dictionary of Apple Japanese access methods.

```
enum {
    kAppleJapaneseDictionarySignature = 'jlan'
};
```

Analysis Engine Keywords

Specify analysis engine keywords for morpheme/homograph information.

```
enum {
    keyAEHomographDicInfo = 'lahd',
    keyAEHomographWeight = 'lahw',
    keyAEHomographAccent = 'laha'
};
```

Analysis Results Constants

Specify the nodes associated with analysis resutls.

```
enum {
    keyAELAMorphemeBundle = 'lmfb',
    keyAELAMorphemePath = 'lmfp',
    keyAELAMorpheme = 'lmfn',
    keyAELAHomograph = 'lmfh'
};
```

Morpheme Key Values

Specify key values used for morpheme/homgraph information.

```
enum {
    keyAEMorphemePartOfSpeechCode = 'lamc',
    keyAEMorphemeTextRange = 'lamt'
};
```

All Morphemes Constant

Specifies to use all morphemes.

```
enum {
    kLAAllMorphemes = 0
};
```

Leading and Trailing Constants

Specify constraints to apply to a string.

```
enum {
    kLADefaultEdge = 0,
    kLAFreeEdge = 1,
    kLAIncompleteEdge = 2
};
```

Converting Mask

Defines a mask for high-level API conversion flags.

```
enum {
    kLAEndOfSourceTextMask = 0x00000001
};
```

Morphemes Array Version

Specifies the version of the array used to hold morpheme analysis results.

```
enum {
    kLAMorphemesArrayVersion = 0
};
```

Conjugation Constants

Specify Japanese conjugations.

```
enum {
    kLASpeechKatsuyouGokan = 0x00000001,
    kLASpeechKatsuyouMizen = 0x00000002,
    kLASpeechKatsuyouRenyou = 0x00000003,
    kLASpeechKatsuyouSyuushi = 0x00000004,
    kLASpeechKatsuyouRentai = 0x00000005,
    kLASpeechKatsuyouKatei = 0x00000006,
    kLASpeechKatsuyouMeirei = 0x00000007
};
```

Parts of Speech Constants

Specify Japanese parts of speech.

```
enum {
    kLASpeechMeishi = 0x00000000,
    kLASpeechFutsuuMeishi = 0x00000000,
    kLASpeechJinmei = 0x00000100,
    kLASpeechJinmeiSei = 0x00000110,
    kLASpeechJinmeiMei = 0x00000120,
    kLASpeechChimei = 0x00000200,
    kLASpeechSetsubiChimei = 0x00000210,
    kLASpeechSoshikimei = 0x00000300,
    kLASpeechKoyuuMeishi = 0x00000400,
    kLASpeechSahenMeishi = 0x00000500,
    kLASpeechKeidouMeishi = 0x00000600,
    kLASpeechRentaishi = 0x00001000,
    kLASpeechFukushi = 0x00002000,
    kLASpeechSetsuzokushi = 0x00003000,
    kLASpeechKandoushi = 0x00004000,
    kLASpeechDoushi = 0x00005000,
    kLASpeechGodanDoushi = 0x00005000,
    kLASpeechKagyouGodan = 0x00005000,
    kLASpeechSagyouGodan = 0x00005010,
    kLASpeechTagyouGodan = 0x00005020,
    kLASpeechNagyouGodan = 0x00005030,
    kLASpeechMagyouGodan = 0x00005040,
    kLASpeechRagyouGodan = 0x00005050,
    kLASpeechWagyouGodan = 0x00005060,
    kLASpeechGagyouGodan = 0x00005070,
    kLASpeechBagyouGodan = 0x00005080,
    kLASpeechIchidanDoushi = 0x00005100,
    kLASpeechKahenDoushi = 0x00005200,
    kLASpeechSahenDoushi = 0x00005300,
    kLASpeechZahenDoushi = 0x00005400,
    kLASpeechKeiyoushi = 0x00006000,
    kLASpeechKeiyoudoushi = 0x00007000,
    kLASpeechSettougo = 0x00008000,
    kLASpeechSuujiSettougo = 0x00008100,
    kLASpeechSetsubigo = 0x00009000,
    kLASpeechJinmeiSetsubigo = 0x00009100,
    kLASpeechChimeiSetsubigo = 0x00009200,
    kLASpeechSoshikimeiSetsubigo = 0x00009300,
    kLASpeechSuujiSetsubigo = 0x00009400,
    kLASpeechMuhinshi = 0x0000A000,
    kLASpeechTankanji = 0x0000A000,
    kLASpeechKigou = 0x0000A100,
    kLASpeechKuten = 0x0000A110,
    kLASpeechTouten = 0x0000A120,
    kLASpeechSuushi = 0x0000A200,
    kLASpeechDokuritsugo = 0x0000A300,
    kLASpeechSeiku = 0x0000A400,
    kLASpeechJodoushi = 0x0000B000,
    kLASpeechJoshi = 0x0000C000
};
```

Parts of Speech Masks

Specify masks for parts of speech.

```
enum {
    kLASpeechRoughClassMask = 0x0000F000,
    kLASpeechMediumClassMask = 0x0000FF00,
    kLASpeechStrictClassMask = 0x0000FFF0,
    kLASpeechKatsuyouMask = 0x0000000F
};
```

Engine Limitations

Specify language analysis engine limitations.

```
enum {
    kMaxInputLengthOfAppleJapaneseEngine = 200
};
```

Analysis Engine Type Definitions

Specify language analysis engine type definitions for morpheme/homograph information.

```
enum {
    typeAEHomographDicInfo = 'lahd',
    typeAEHomographWeight = typeShortInteger,
    typeAEHomographAccent = 'laha'
};
```

Morpheme Types

Specify data types for morphemes.

```
enum {
    typeAEMorphemePartOfSpeechCode = 'lamc',
    typeAEMorphemeTextRange = 'lamt'
};
```

Morpheme Type Analysis Constants

Specify types used in morphological analysis.

```
enum { typeLAMorphemeBundle = typeAERecord, typeLAMorphemePath = typeAERecord,
typeLAMorpheme = typeAEList, typeLAHomograph = typeAEList };
```

Default Environment Names

Specify names for default environments for Japanese analysis.

```
#define kLAJapaneseKanaKanjiEnvironment "\pKanaKanjiConversion"
#define kLAJapaneseMorphemeAnalysisEnvironment
                                   "\pJapaneseMorphemeAnalysis"
#define kLAJapaneseTTSEnvironment          "\pJapaneseTextToSpeech"
```

Result Codes

The most common result codes retuned by the Language Analysis Manager are listed in the table below.

Result Code	Value	Description
`laTooSmallBufferErr`	-6984	The output buffer is too small to store any result. Available in Mac OS X v10.0 and later.
`laEnvironmentBusyErr`	-6985	The specified environment is used. Available in Mac OS X v10.0 and later.
`laEnvironmentNotFoundErr`	-6986	The specified environment can’t be found. Available in Mac OS X v10.0 and later.
`laEnvironmentExistErr`	-6987	An environment by the same name already exists. Available in Mac OS X v10.0 and later.
`laInvalidPathErr`	-6988	The path is not correct. Available in Mac OS X v10.0 and later.
`laNoMoreMorphemeErr`	-6989	There is nothing to read. Available in Mac OS X v10.0 and later.
`laFailAnalysisErr`	-6990	The analysis failed. Available in Mac OS X v10.0 and later.
`laTextOverFlowErr`	-6991	The text is too long. Available in Mac OS X v10.0 and later.
`laDictionaryNotOpenedErr`	-6992	The dictionary is not opened. Available in Mac OS X v10.0 and later.
`laDictionaryUnknownErr`	-6993	This dictionary can’t be used with this environment. Available in Mac OS X v10.0 and later.
`laDictionaryTooManyErr`	-6994	There are too many dictionaries. Available in Mac OS X v10.0 and later.
`laPropertyValueErr`	-6995	Invalid property value. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
`laPropertyUnknownErr`	-6996	The property is unknown to this environment. Available in Mac OS X v10.0 and later.
`laPropertyIsReadOnlyErr`	-6997	The property is read only. Available in Mac OS X v10.0 and later.
`laPropertyNotFoundErr`	-6998	The property can't be found. Available in Mac OS X v10.0 and later.
`laPropertyErr`	-6999	There is an error in the property. Available in Mac OS X v10.0 and later.
`laEngineNotFoundErr`	-7000	The engine can't be found. Available in Mac OS X v10.0 and later.

Deprecated Language Analysis Manager Functions

A function identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.5

LAAddNewWord

Adds a new word to a dictionary. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAAddNewWord (
   LAEnvironmentRef environ,
   const FSSpec *dictionary,
   const AEDesc *dataList
);
```

Parameters

environ

A reference to the language analysis environment for the dictionary you want to modify.

dictionary

The file specification for the dictionary you want to modify.

dataList

A pointer to an `AEDesc` data structure that specifies the word you want to add to the dictionary. See the Apple Event Manager documentation for more information on Apple Event descriptor records.

Return Value

A result code. See "Result Codes" (page 17).

Availability

Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`LanguageAnalysis.h`

LACloseAnalysisContext

Closes the specified language analysis context. (Deprecated in Mac OS X v10.5.)

```
OSStatus LACloseAnalysisContext (
   LAContextRef context
);
```

Parameters

context

A reference to the language analysis context you want to close.

Return Value

A result code. See "Result Codes" (page 17).

Availability

Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`LanguageAnalysis.h`

LACloseDictionary

Closes a dictionary in the specified environment. (Deprecated in Mac OS X v10.5.)

```
OSStatus LACloseDictionary (
   LAEnvironmentRef environ,
   const FSSpec *dictionary
);
```

Parameters

environ

A reference to the language analysis environment for which you want to close a dictionary.

dictionary

The file specification for the dictionary you want to close.

Return Value

A result code. See "Result Codes" (page 17).

Availability

Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`LanguageAnalysis.h`

LAContinuousMorphemeAnalysis

Performs a continuous morphological analysis of Unicode text. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAContinuousMorphemeAnalysis (
   LAContextRef context,
   ConstUniCharArrayPtr text,
   UniCharCount textLength,
   Boolean incrementalText,
   LAMorphemePath *leadingPath,
   LAMorphemePath *trailingPath,
   Boolean *modified
);
```

Parameters

context

A reference to the language analysis context whose text you want to analyze. You can obtain a language analysis context by calling the function `LAOpenAnalysisContext`.

text

A pointer to the Unicode text string you want to analyze.

textLength

The length of the Unicode text string specified in the `text` parameter. This value must specify the number of `UniChar` (double-byte) values in the string.

incrementalText

A Boolean value that indicates the method for passing text. Pass `false` to specify you want the text to be analyzed as a whole and the analysis started. Pass `true` if the text is a continuation of the text currently held by the context, and should be added to the context before undergoing analysis.

leadingPath

A pointer to the morpheme path that specifies the results of analyzing the text just previous to the string specified by the `text` parameter. The Langauage Analysis Manager uses this string to restrict the analysis. For example, if the previous section ends with a noun, the text that follows can begin with a verb. If no valid leading path is available you can pass `NULL` or a "Leading and Trailing Constants" (page 13)—`kLAFreeEdge` or `kLADefaultEdge`. Pass `kLAFreeEdge` if it is possible for an optional morpheme to come at the start or the end of analysis. Pass `kLADefaultEdge` if you want the analysis is carried out so that the start/end of analysis becomes the start of the sentence/end of sentence or the start of the segment/end of segment. Definitions for start of sentence/end of sentence and start of segment/end of segment depend on the engine.

trailingPath

A pointer to the morpheme path that specifies the results of analyzing the text that follows the string specified by the `text` parameter. When performing a continuous analysis, you must pass the constant `kLAIncompleteEdge` to indicate that the string is not complete. Note that the function `LAGetMorphemes` only returns the results it has completed analyzing, not the analysis of the complete source text. If you want to obtain all of the analysis results up to a point, (if a user expressly indicates a conversion with the space bar in a kana-kanji conversion program, and so forth) then you can pass a value other than `kLAIncompleteEdge`. Then, when you call the function `LAGetMorphemes` you obtain analysis results for that portion of the string that has been analyzed to that point.

modified

On output, `true` if the internal state of the context is changed (new analyzed morphemes are generated); otherwise `false`. When `true` is returned, you should call the function `LAGetMorphemes` and update the display. If modified is specified as `NULL`, values are not returned.

Return Value

A result code. See "Result Codes" (page 17).

Discussion
The function `LAContinuousMorphemeAnalysis` does not return analysis results, but holds them internally. You can obtain the results by calling the functions `LAGetMorphemes` or `LAShiftMorphemes`. In contrast to the function `LAMorphemeAnalysis` you cannot obtain multiple paths for an analysis done using the function `LAContinuousMorphemeAnalysis`.

You can obtain the same results as calling the function `LAResetAnalysis` by calling `LAContinuousMorphemeAnalysis` with the `text` parameter set to "" and the `incrementalText` parameter set to `false`.

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LACreateCustomEnvironment

Creates a new environment with the specified name. (Deprecated in Mac OS X v10.5.)

```
OSStatus LACreateCustomEnvironment (
   LAEnvironmentRef baseEnvironment,
   ConstStr63Param newEnvironmentName,
   Boolean persistent,
   LAEnvironmentRef *newEnvironment
);
```

Parameters

baseEnvironment
A reference to the language analysis environment that you want to use as the base environment.

newEnvironmentName
The name for the newly-created environment. This name must be unique. If an environment with the same name already exists, the function returns the result code `laEnvironmentExistErr`.

persistent
A Boolean value that specifies whether the environment should be persistent (`true`) or not (`false`). If you pass `true`, the newly-created environment is saved to disk, and it can be referred to at any time subsequently by using the name. If you pass `false`, the newly-created environment can only be used during that session. Additionally, environments created with persistent set to `false` are not returned in the list provided by the function `LAGetEnvironmentList`, so these environments can be used only as private environments. If you create a private environment, you must call the function `LADeleteCustomEnviroment` to dispose of it before you terminate your application.

newEnvironemnt
On output, a reference to the newly-created language analysis environment.

Return Value
A result code. See "Result Codes" (page 17).

Discussion
If you open or close dictionaries for custom environments, it is possible to create independent environments without interfering with existing environments.

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LADeleteCustomEnvironment

Disposes of a reference to a custom language analysis environment. (Deprecated in Mac OS X v10.5.)

```
OSStatus LADeleteCustomEnvironment (
   LAEnvironmentRef environment
);
```

Parameters

environment

A reference to the language analysis environment you want to dispose of.

Return Value
A result code. See "Result Codes" (page 17).

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAGetEnvironmentList

Obtains a list of the available language analysis environments. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAGetEnvironmentList (
   UInt32 maxCount,
   UInt32 *actualCount,
   LAEnvironmentRef environmentList[]
);
```

Parameters

maxCount

The maximum number of environments provided by the system. To determine this value, see the Discussion.

actualCount

On output, the actual number of environments.

environmentList

On output, a list of the available environments. You must allocate a buffer of the appropriate size. If you are uncertain of how much memory to allocate for this array, see the Discussion.

Return Value
A result code. See "Result Codes" (page 17).

Discussion
Typically, you use the function `LAGetEnvironmentList` by calling it twice, as follows:

1. Pass `0` for the `maxCount` parameter and `NULL` for the `environmentList` parameter.

2. Allocate enough space for an array of the size specified by `actualCount`, then call the function `LAGetEnvironmentList` again. This time, provide a count of the actual number of environments as the `maxCount` parameter, and a pointer to a buffer of the correct size for the `environmentList` parameter. On output, the pointer points to an array of the available environments.

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAGetEnvironmentName

Obtains the name of an environment. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAGetEnvironmentName (
   LAEnvironmentRef environment,
   Str63 environmentName
);
```

Parameters

environment

A reference to the language analysis environment whose name you want to obtain.

environmentName

On return, the environment name.

Return Value
A result code. See "Result Codes" (page 17).

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAGetEnvironmentRef

Obtains the language analysis environment reference associated with an environment name (Deprecated in Mac OS X v10.5.)

```
OSStatus LAGetEnvironmentRef (
   ConstStr63Param targetEnvironmentName,
   LAEnvironmentRef *environment
);
```

Parameters

targetEnvironmentName
The environment name whose language analysis environment reference you want to obtain.

environment
On output, a reference to the language analysis environment associated with the environment name.

Return Value
A result code. See "Result Codes" (page 17).

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAGetMorphemes

Reads the results of a continuous morpheme analysis. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAGetMorphemes (
   LAContextRef context,
   LAMorphemePath *result
);
```

Parameters

context
A reference to the language analysis context whose result you want to obtain. You can obtain a language analysis context by calling the function `LAOpenAnalysisContext`.

result
On output, points to the morpheme bundle that contains the results of the analysis. You are responsible for disposing of this structure by calling the Apple Event Manager function `AEDisposeDesc`.

Return Value
A result code. See "Result Codes" (page 17).

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`LanguageAnalysis.h`

LALibraryVersion

Returns the version of the Language Analysis Manager installed. (Deprecated in Mac OS X v10.5.)

```
UInt32 LALibraryVersion (
   void
);
```

Return Value

Returns the version of Language Analysis manager that is installed.

Discussion

The function `LALibraryVersion` returns the version of the Language Analysis Manager installed in the same format as `'vers'` resource. That is to say, the version number is returned in BCD (Binary-Coded Decimal) format to higher-place words, while release stage information is returned to lower-place words. For example, version 1.1.1 final release library returns 0x01118000.

Availability

Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`LanguageAnalysis.h`

LAListAvailableDictionaries

Obtains the number of dictionaries available in a specified environment. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAListAvailableDictionaries (
   LAEnvironmentRef environ,
   ItemCount maxCount,
   ItemCount *actualCount,
   FSSpec dictionaryList[],
   Boolean opened[]
);
```

Parameters

`environ`

A reference to the language analysis environment for which you want to obtain a list of available dictionaries.

`maxCount`

The maximum number of available dictionaries. To determine this value, see the Discussion.

actualCount

On output, the actual number of available dictionaries.

dictionaryList

On output, points to a list of available dictionaries. You must allocate a buffer of the appropriate size. If you are uncertain of how much memory to allocate for this array, see the Discussion.

opened

On output, points to a list of Boolean values that specify whether the available dictionaries are open. This array is parallel to the `dictionaryList` array. A dictionary file whose associated value is `true` is open and `false` if it is not open. You must allocate a buffer of the appropriate size. If you are uncertain of how much memory to allocate for this array, see the Discussion.

Return Value

A result code. See "Result Codes" (page 17).

Discussion

Typically, you use the function `LAListAvailableDictionaries` by calling it twice, as follows:

1. Pass `0` for the `maxCount` parameter, `NULL` for the `dictionaryList` parameter, and `NULL` for the `opened` parameter.

2. Allocate enough space for arrays of the size specified by `actualCount`, then call the function `LAListAvailableDictionaries` again. This time, provide a count of the actual number of dictionaries as the `maxCount` parameter, and a pointer to buffers of the correct size for the `dictionaryList` and `opened` parameters. On output, `dictionaryList` points to an array of the available dictionaries and `opened` points to an array that specifies whether each dictionary is opened or closed.

Availability

Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`LanguageAnalysis.h`

LAMorphemeAnalysis

Performs a morphological analysis of the specified Unicode text. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAMorphemeAnalysis (
   LAContextRef context,
   ConstUniCharArrayPtr text,
   UniCharCount textLength,
   LAMorphemePath *leadingPath,
   LAMorphemePath *trailingPath,
   ItemCount pathCount,
   LAMorphemeBundle *result
);
```

Parameters

context

A reference to the language analysis context whose text you want to analyze. You can obtain a language analysis context by calling the function `LAOpenAnalysisContext`.

text

A pointer to the Unicode text string you want to analyze.

textLength

The length of the Unicode text string specified in the `text` parameter. This value must specify the number of `UniChar` (double-byte) values in the string.

leadingPath

A pointer to the morpheme path that specifies the results of analyzing the text just previous to the string specified by the `text` parameter. The Language Analysis Manager uses this string to restrict the analysis. For example, if the previous section ends with a noun, the text that follows can begin with a verb. If no valid leading path is available you can pass `NULL` or a "Leading and Trailing Constants" (page 13)—`kLAFreeEdge` or `kLADefaultEdge`. Pass `kLAFreeEdge` if it is possible for an optional morpheme to come at the start or the end of analysis. Pass `kLADefaultEdge` if you want the analysis is carried out so that the start/end of analysis becomes the start of the sentence/end of sentence or the start of the segment/end of segment. Definitions for start of sentence/end of sentence and start of segment/end of segment depend on the engine.

trailingPath

A pointer to the morpheme path that specifies the results of analyzing the text that follows the string specified by the `text` parameter. The Language Analysis Manager uses this string to restrict the analysis. For example, if the following section begins with a verb, the text that precedes it can begin with a noun. If no valid trailing path is available you can pass `NULL` or a "Leading and Trailing Constants" (page 13)—`kLAFreeEdge` or `kLADefaultEdge`. Pass `kLAFreeEdge` if it is possible for an optional morpheme to come at the start or the end of analysis. Pass `kLADefaultEdge` if you want the analysis is carried out so that the start/end of analysis becomes the start of the sentence/end of sentence or the start of the segment/end of segment. Definitions for start of sentence/end of sentence and start of segment/end of segment depend on the engine.

pathCount

On output, specifies the maximum rank of the returned path.

result

On output, points to the morpheme bundle that contains the results of the analysis. You are responsible for disposing of this structure by calling the Apple Event Manager function `AEDisposeDesc`.

Return Value

A result code. See "Result Codes" (page 17).

Discussion
If you have previously called the function `LAContinuousMorphemeAnalysis`, and you the call the function `LAMorphemeAnalysis`, the internal state maintained by the function `LAContinuousMorphemeAnalysis` is disposed of. Then, if you call the functions `LAGetMorphemes` and `LAShiftMorphemes` the result code `laNoMoreMorphemeErr` is returned .

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAOpenAnalysisContext

Creates a language analysis context from a specified language analysis environment. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAOpenAnalysisContext (
   LAEnvironmentRef environ,
   LAContextRef *context
);
```

Parameters

environ

A reference to the language analysis environment for which you want to open a context.

context

On output, a language analysis context derived from the specified language analysis environment.

Return Value
A result code. See "Result Codes" (page 17).

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAOpenDictionary

Opens a dictionary for the specified environment. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAOpenDictionary (
   LAEnvironmentRef environ,
   const FSSpec *dictionary
);
```

Parameters

environ

A reference to the language analysis environment for which you want to open the dictionary.

dictionary

The file specification for the dictionary you want to open.

Return Value

A result code. See "Result Codes" (page 17).

Discussion

The environment makes an appropriate assessment of type of dictionary, user dictionary, option dictionary and so forth, before carrying out necessary operations.

Availability

Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

Declared In

`LanguageAnalysis.h`

LAResetAnalysis

Clears the internal status of the analysis context. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAResetAnalysis (
   LAContextRef context
);
```

Parameters

context

A reference to the language analysis context whose analysis you want to reset. You can obtain a language analysis context by calling the function `LAOpenAnalysisContext`.

Return Value

A result code. See "Result Codes" (page 17).

Discussion

Clear the internal status of analysis context. This is accessed before the continuous analysis by the next LAContinuousMorphemeAnalysis. Accessing LAGetMorphemes and LAShiftMorphemes immediately after this call will fail.

The same result will be achieved even if LAContinuousMorphemeAnalysis is accessed as text = "", incrementalText = false

Availability

Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LAShiftMorphemes

Shifts the read out of continuous morpheme analysis. (Deprecated in Mac OS X v10.5.)

```
OSStatus LAShiftMorphemes (
   LAContextRef context,
   ItemCount morphemeCount,
   LAMorphemePath *path,
   UniCharCount *shiftedLength
);
```

Parameters

context
A reference to the language analysis context whose read-out you want to shift. You can obtain a language analysis context by calling the function `LAOpenAnalysisContext`.

morphemeCount
The number of morphemes to be shifted. If you pass `kAllMorphemes`, all morphemes which are analized are returned.

path
If you pass `typeNull` a new path is created. If you pass a valid path, the morpheme read out at the end of the path is added. This is handy when this path is to be used as the leading edge the next time `LAContinuousMorphemeAnalysis` is accessed. In both cases, when you are done using the path, you must dispose of it by calling the Apple Event Manager function `AEDispose`.

shiftedLength
A pointer to the input character string length (in `UniChars`) corresponding to the morpheme read out. If you pass `NULL`, it is not returned.

Return Value
A result code. See "Result Codes" (page 17).

Discussion
When you call the function `LAShiftMorphemes`, the results of the analysis performed by the function `LAContinuousMorphemeAnalysis` are returned from the start to the number of morpheme readout paths specified in the `morphemeCount` parameter . Morphemes which have been read out are deleted from analysis context. For example, if "AABBCC" represents the internal status, after you fetch the morpheme "AA" by calling `LAShiftMorphemes`, the internal status becomes "BBCC".

The results you obtain by calling the funciton `LAShiftMorphemes`, are impacted by the `trailingEdge` parameter of the function `LAContinuousMorphemeAnalysis`. If the value of the `trailingEdge` parameter is `kLAIncompleteEdge`, the function `LAShiftMorphemes` might not return all morphemes and non-converted sections. The analysis engine only returns morphemes with a high degree of certainty. For example, those morphemes which are likely to change if text is added, and `laNoMoreMorphemeErr` is returned in subsequent accesses. If something other than `kLAIncompleteEdge` is passed as the `trailingEdge` parameter, it is possible to fetch morphemes up to the final morpheme. After all morphemes are fetched, the result code `laNoMoreMorphemeErr` is returned to indicate that nothing remains. This is the same as the status returned after calling the function `LAResetAnalysis`.

You can carry out a continuous analysis using the functions `LAContinuousMorphemeAnalysis` and `LAShiftMorphemes` in two ways. The first method leaves as much text as possible within the analysis engine. That is, continue to provide text to the fucntoin `LAContinuousMorphemeAnalysis` until you encounter the result code `laTextOverflowErr` , and call the function `LAShiftMorphemes` once when the error is returned. The second method leaves as little text as possible within the analysis search engine. That is, continue to provide text to the function `LAContinuousMorphemeAnalysis` until `true` is returned to the `modified` parameter. When `true` is returned, call the function `LAShiftMorphemes` with the `morphemeCount` parameter set to `kAllMorphemes`.

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

LATextToMorphemes

Performs a morphological analysis of the specified text. (Deprecated in Mac OS X v10.5.)

```
OSStatus LATextToMorphemes (
   LAContextRef context,
   TextEncoding preferedEncoding,
   ByteCount textLength,
   ConstLogicalAddress sourceText,
   ByteCount bufferSize,
   OptionBits convertFlags,
   UInt32 structureVersion,
   ByteCount *acceptedLength,
   LAMorphemesArrayPtr resultBuffer
);
```

Parameters

`context`
A reference to the language analysis context whose text you want to analyze. You can obtain a language analysis context by calling the function `LAOpenAnalysisContext`.

`preferedEncoding`
A value of type `TextEncoding` that specifies the encoding of text to use for both input and output. The text and length included in the `results` parameter are adjusted in accordance with the encoding specified here.

`textLength`
The length, in bytes, of the text you want to analyze.

`sourceText`
A pointer to the text you want to analyze.

`bufferSize`
The size of the buffer pointed to by the `resultBuffer` parameter.

convertFlags

An `OptionBits` value that specifies how to proceed with the analysis. Currently the only option you can set is `kLAEndOfSourceTextMask`. If this bit is set, the source text is analyzed to the end, and then results are generated. If this bit is not set, there is a possibility that the end portion of the source text is not yet analyzed when results are available. For example, when a large text file is analyzed it may be preferable to analyze it in chunks, returning results as each chunk is analyzed. You can specify this by passing 0 for the `convertFlags` parameter, advancing the analysis, and the setting `kLAEndOfSourceTextMask` when the whole file has been read.

structureVersion

The current version of `LAMorphemesArrayPtr`. You should pass `kLAMorphemesArrayVersion`.

acceptedLength

On output, the length of the source text that is accepted by the analysis engine.

resultBuffer

On output, a pointer to an array of `LAMorphmesArray` structures that contain the results of the morphological analysis.

Return Value

A result code. See "Result Codes" (page 17).

Discussion

The function `LATextToMorphemes` analyzes the text specified in `textLength` and `sourceText`, and returns the results to `resultBuffer` in the form of `LAMorphemesArray`. While there are no restrictions on the length of text specified, the length in byte units of `sourceText` received in this call is set to `acceptedLength` at the point where the output buffer becomes full, or until all text provided has been analyzed. In practice, sections currently being analyzed exist within the analysis context, so be aware that the length returned may not necessarily be the same as the section included in analysis results. This means that if the length of the returned text is shorter than the source text, analysis results are not complete. In this case, fetch the `results`, increment the `sourceText` by the `acceptedLength`, shorten `textLength` by the `acceptedLength`, and repeatedly call `LATextToMorphemes` until all the text is analyzed. The sample code below shows how to analyze text while loading it from a file.

```
while ( fileErr == noErr )
{
    fileErr = ReadFile (readBufferSize, &actualReadSize, readBuffer);
    if ( fileErr == eofErr )
        analyzeOption = kLAEndOfSourceTextMask;
    else
        analyzeOption = 0;
    analyzeLen = actualReadSize;
    analyzeText = readBuffer;
    result->morphemesCount = 0;
    while (analyzeLen || result->morphemesCount)
    {
        err = LATextToMorphemes (context, kTextEncodingMacJapanese,
                        analyzeLen, analyzeText, resultBufferSize,
                        analyzeOption, kLAMorphemesArrayVersion,
                        &acceptedLen, result);
        if (result->morphemesCount > 0 )
        {
                //
                // Retrieve result here...
                //
        }
        analyzeText += acceptedLen; // Increment source text ptr
        analyzeLen -= acceptedLen; // Decrement source text length
```

```
            }
}
```

If `kLAEndOfSourceTextMask` is specified and the analysis of all of the source text is done, the context becomes empty. If the analysis is suspended under this or other conditions (including errors), you must call the function `LAResetAnalysis` to clear the context.

Availability
Available in CarbonLib 1.0 and later when LanguageAnalysis 1.0 or later is present.
Available in Mac OS X 10.0 and later.
Deprecated in Mac OS X v10.5.
Not available to 64-bit applications.

Declared In
`LanguageAnalysis.h`

Document Revision History

This table describes the changes to *Language Analysis Manager Reference*.

Date	Notes
2003-04-01	Added documentation for functions, data types, and constants.

Index

A

All Morphemes Constant 13
Analysis Engine Keywords 13
Analysis Engine Type Definitions 16
Analysis Results Constants 13

C

Conjugation Constants 14
Converting Mask 14

D

Default Environment Names 16

E

Engine Limitations 16

F

File Creator Constants 12

H

`HomographAccent` data type 6
`HomographDicInfoRec` structure 7
`HomographWeight` data type 7

J

`JapanesePartOfSpeech` data type 7

L

`LAAddNewWord` function (Deprecated in Mac OS X v10.5) 19
`LACloseAnalysisContext` function (Deprecated in Mac OS X v10.5) 19
`LACloseDictionary` function (Deprecated in Mac OS X v10.5) 20
`LAContextRef` data type 7
`LAContinuousMorphemeAnalysis` function (Deprecated in Mac OS X v10.5) 20
`LACreateCustomEnvironment` function (Deprecated in Mac OS X v10.5) 22
`LADeleteCustomEnvironment` function (Deprecated in Mac OS X v10.5) 23
`laDictionaryNotOpenedErr` constant 17
`laDictionaryTooManyErr` constant 17
`laDictionaryUnknownErr` constant 17
`laEngineNotFoundErr` constant 18
`laEnvironmentBusyErr` constant 17
`laEnvironmentExistErr` constant 17
`laEnvironmentNotFoundErr` constant 17
`LAEnvironmentRef` data type 8
`laFailAnalysisErr` constant 17
`LAGetEnvironmentList` function (Deprecated in Mac OS X v10.5) 23
`LAGetEnvironmentName` function (Deprecated in Mac OS X v10.5) 24
`LAGetEnvironmentRef` function (Deprecated in Mac OS X v10.5) 25
`LAGetMorphemes` function (Deprecated in Mac OS X v10.5) 25
`LAHomograph` data type 8
`laInvalidPathErr` constant 17
`LALibraryVersion` function (Deprecated in Mac OS X v10.5) 26

LAListAvailableDictionaries function (Deprecated in Mac OS X v10.5) 26
LAMorpheme data type 8
LAMorphemeAnalysis function (Deprecated in Mac OS X v10.5) 27
LAMorphemeBundle data type 9
LAMorphemePath data type 9
LAMorphemeRec structure 10
LAMorphemesArray structure 11
laNoMoreMorphemeErr constant 17
LAOpenAnalysisContext function (Deprecated in Mac OS X v10.5) 29
LAOpenDictionary function (Deprecated in Mac OS X v10.5) 29
laPropertyErr constant 18
laPropertyIsReadOnlyErr constant 18
LAPropertyKey data type 11
laPropertyNotFoundErr constant 18
LAPropertyType data type 12
laPropertyUnknownErr constant 18
laPropertyValueErr constant 17
LAResetAnalysis function (Deprecated in Mac OS X v10.5) 30
LAShiftMorphemes function (Deprecated in Mac OS X v10.5) 31
laTextOverFlowErr constant 17
LATextToMorphemes function (Deprecated in Mac OS X v10.5) 32
laTooSmallBufferErr constant 17
Leading and Trailing Constants 13

M

Morpheme Key Values 13
Morpheme Type Analysis Constants 16
Morpheme Types 16
MorphemePartOfSpeech data type 12
Morphemes Array Version 14
MorphemeTextRange structure 12

P

Parts of Speech Constants 14
Parts of Speech Masks 15