Locale Utilities Reference

Internationalization > Localization

2002-01-24

Contents

Locale Utilities Reference 5

Overview 5
Functions by Task 5
- Obtaining Available Locales 5
- Obtaining Localized Locale Names 5
- Converting Among Locale Data Formats 6

Functions 6
- LocaleCountNames 6
- LocaleGetIndName 7
- LocaleGetName 8
- LocaleOperationCountLocales 10
- LocaleOperationCountNames 10
- LocaleOperationGetIndName 11
- LocaleOperationGetLocales 12
- LocaleOperationGetName 13
- LocaleRefFromLangOrRegionCode 14
- LocaleRefFromLocaleString 16
- LocaleRefGetPartString 16
- LocaleStringToLangAndRegionCodes 17

Data Types 18
- LocaleAndVariant 18
- LocaleNameMask 19
- LocaleOperationClass 19
- LocaleOperationVariant 20
- LocalePartMask 20
- LocaleRef 21

Constants 21
- Locale Name Masks 21
- Locale Part Masks 22

Appendix A Deprecated Locale Utilities Functions 25

Deprecated in Mac OS X v10.5 25
- LocaleGetRegionLanguageName 25

Document Revision History 27

Index 29

Locale Utilities Reference

Framework:	CoreServices/CoreServices.h
Declared in	MacLocales.h

Overview

Unicode operations such as collation and text boundary determination can be affected by the conventions of a particular language or region. You can use Locale Utilities to specify language-specific or region-specific information for locale-sensitive Unicode operations. The Locale Utilities provide support for obtaining available locales, obtaining localized locale names, and converting among locale data formats.

Carbon supports the Locales Utilities.

Functions by Task

Obtaining Available Locales

`LocaleOperationGetLocales` (page 12)
Obtains the list of available locale-and-variant combinations for a given class of operation.

`LocaleOperationCountLocales` (page 10)
Obtains the total count of locale-and-variant combinations available for a given class of operation.

Obtaining Localized Locale Names

`LocaleGetName` (page 8)
Obtains the localized name for a locale and/or operation variant, based on the requested language for the localized name.

`LocaleGetIndName` (page 7)
Obtains a localized name for a locale and/or operation variant, based on an index into the list of all available localized names.

`LocaleCountNames` (page 6)
Obtains the total count of all available localized names for a locale and/or operation variant.

`LocaleOperationGetName` (page 13)
Obtains the localized name for an operation class, based on the requested language for the localized name.

`LocaleOperationGetIndName` (page 11)
: Obtains a localized name for an operation class, based on an index into the list of all available localized names.

`LocaleOperationCountNames` (page 10)
: Obtains the total count of all available localized names for an operation class.

`LocaleGetRegionLanguageName` (page 25) Deprecated in Mac OS X v10.5
: Obtains the localized language name for a region.

Converting Among Locale Data Formats

`LocaleRefFromLangOrRegionCode` (page 14)
: Converts a Mac OS language or region code to a locale reference.

`LocaleRefFromLocaleString` (page 16)
: Converts a string containing locale data to a locale reference.

`LocaleRefGetPartString` (page 16)
: Converts a locale reference to a string containing locale data.

`LocaleStringToLangAndRegionCodes` (page 17)
: Converts a string containing locale data to Mac OS language and region codes.

Functions

LocaleCountNames

Obtains the total count of all available localized names for a locale and/or operation variant.

```
OSStatus LocaleCountNames (
   LocaleRef locale,
   LocaleOperationVariant opVariant,
   LocaleNameMask nameMask,
   ItemCount *nameCount
);
```

Parameters

locale
: A `LocaleRef` value identifying the locale for which you are requesting a localized name.

opVariant
: A `LocaleOperationVariant` value identifying the operation variant for which you are requesting a localized name. Pass 0 to obtain the default operation variant.

nameMask
: A `LocaleNameMask` value specifying the parts of the name that you are requesting: the name of the locale alone, the name of the operation variant alone, or the name for the combination of the two.

nameCount
: A pointer to an `ItemCount` value. On return, the value is set to the total count of all available localized names for the locale and/or operation variant.

Return Value

A result code. The `LocaleCountNames` function can return `paramErr` if `nameCount` is `NULL`.

Discussion

This function provides a count of all available localized names for a given locale, operation variant, or locale-and-variant combination, for use in the function `LocaleGetIndName` (page 7). You must call this function prior to calling `LocaleGetIndName`.

Special Considerations

This function can move memory in Carbon and Mac OS 9.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleGetIndName

Obtains a localized name for a locale and/or operation variant, based on an index into the list of all available localized names.

```
OSStatus LocaleGetIndName (
   LocaleRef locale,
   LocaleOperationVariant opVariant,
   LocaleNameMask nameMask,
   ItemCount nameIndex,
   UniCharCount maxNameLen,
   UniCharCount *actualNameLen,
   UniChar displayName[],
   LocaleRef *displayLocale
);
```

Parameters

locale

A `LocaleRef` value identifying the locale for which you are requesting a localized name.

opVariant

A `LocaleOperationVariant` value identifying the operation variant for which you are requesting a localized name. Pass 0 to obtain the default operation variant.

nameMask

A `LocaleNameMask` value specifying the parts of the name that you are requesting: the name of the locale alone, the name of the operation variant alone, or the name for the combination of the two.

nameIndex

An `ItemCount` index value identifying an entry in the list of available localized names. You can obtain a count of all available localized names from the `nameCount` parameter of the function `LocaleCountNames` (page 6). Note that index values must be in the range from 0 to `nameCount` -1. To create a list of all available localized names that correspond to the locale and operation variant you request, you can call the `LocaleGetIndName` function starting with an index value of 0, then increment the index value with each successive call to `LocaleGetIndName`, continuing to `nameCount`-1.

maxNameLen

A `UniCharCount` value specifying the length of the `UniChar` array being passed in the `displayName` parameter. An array of 32 characters is typically adequate if you request just a single name (that is, the name for a locale or an operation variant), or 64 characters if you request both names.

actualNameLen

A pointer to a `UniCharCount` value. On return, the value contains the actual length of the name produced in the `displayName` parameter.

displayName

An array of `UniChar` values. On return, the array contains the requested name as localized for a particular display locale.

displayLocale

A pointer to a `LocaleRef` value. On return, the value indicates the language of the name produced in the `displayName` parameter.

Return Value

A result code. If the value specified in the `maxNameLen` parameter is too small for the requested string, `LocaleGetIndName` returns `kLocalesBufferTooSmallErr` (-30001).

Discussion

The `LocaleGetIndName` function obtains a localized name for a locale, operation variant, or locale-and-variant combination. `LocaleGetIndName` produces the name based on a requested locale, an operation variant, and an index into the count of all available localized names for the locale and/or operation variant. The language for the obtained name is produced on return.

You can use repeated calls to the `LocaleGetIndName` function to create a list of all available localized names (that is, the names from all available display locales), that correspond to a given locale and operation variant. Alternately, for a particular locale, operation variant, or locale-and-variant combination, the function `LocaleGetName` (page 8) obtains the corresponding name as localized for a particular display locale.

Note that, in some languages, the name for a language or locale may have several different grammatical forms, where the correct form depends on the usage or context. For example, Swedish uses different forms of a language name depending on whether the name is applied to a collation order, to text break rules, or to keyboard layouts. However, the Locale Utilities functions only return one form of a language or locale name—typically that name which would be used for the language name alone. Therefore, this name may not be the correct form for some usages in some languages.

Special Considerations

This function can move memory in Carbon and Mac OS 9.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleGetName

Obtains the localized name for a locale and/or operation variant, based on the requested language for the localized name.

```
OSStatus LocaleGetName (
   LocaleRef locale,
   LocaleOperationVariant opVariant,
   LocaleNameMask nameMask,
   LocaleRef displayLocale,
   UniCharCount maxNameLen,
   UniCharCount *actualNameLen,
   UniChar displayName[]
);
```

Parameters

locale

A `LocaleRef` value identifying the locale for which you are requesting a localized name.

opVariant

A `LocaleOperationVariant` value identifying the operation variant for which you are requesting a localized name. Pass 0 to obtain the default operation variant.

nameMask

A `LocaleNameMask` value specifying the parts of the name that you are requesting: the name of the locale alone, the name of the operation variant alone, or the name for the combination of the two.

displayLocale

A `LocaleRef` value indicating the requested language for the name produced in the `displayName` parameter. If `LocaleGetName` cannot find a name in the requested language, it produces a name in the default display locale.

maxNameLen

A `UniCharCount` value specifying the length of the `UniChar` array being passed in the `displayName` parameter. An array of 32 characters is typically adequate if you request just a single name (that is, the name for a locale or an operation variant), or 64 characters if you request both names.

actualNameLen

A pointer to a `UniCharCount` value. On return, the value contains the actual length of the name produced in the `displayName` parameter.

displayName

An array of `UniChar` values. On return, the array contains the requested name as localized for a particular display locale.

Return Value

A result code. If `LocaleGetName` cannot find a name that matches the language specified in the `displayLocale` parameter, it returns `kLocalesDefaultDisplayStatus` (-30029). If the value specified in the `maxNameLen` parameter is too small for the requested string, `LocaleGetName` returns `kLocalesBufferTooSmallErr` (-30001).

Discussion

For a particular locale, operation variant, or locale-and-variant combination, the `LocaleGetName` function obtains the corresponding name as localized for a particular display locale. Alternately, you can use repeated calls to the function `LocaleGetIndName` (page 7) to iterate through all of the available display locales to create a list of the corresponding names as localized in each of the display locales.

Note that, in some languages, the name for a language or locale may have several different grammatical forms, where the correct form depends on the usage or context. For example, Swedish uses different forms of a language name depending on whether the name is applied to a collation order, to text break rules, or to keyboard layouts. However, the Locale Utilities functions only return one form of a language or locale name—typically that name which would be used for the language name alone. Therefore, this name may not be the correct form for some usages in some languages.

Special Considerations

This function can move memory in Carbon and Mac OS 9.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleOperationCountLocales

Obtains the total count of locale-and-variant combinations available for a given class of operation.

```
OSStatus LocaleOperationCountLocales (
   LocaleOperationClass opClass,
   ItemCount *localeCount
);
```

Parameters

opClass

A `LocaleOperationClass` value identifying the operation class for which you are requesting availability information.

localeCount

A pointer to an `ItemCount` value. On return, the value is set to the total count of all available locale-and-variant combinations for the specified class of operation.

Return Value

A result code. The `LocaleOperationCountLocales` function returns `paramErr` if the `opClass` parameter is 0 or if the `localeCount` parameter is `NULL`.

Discussion

The `LocaleOperationCountLocales` function counts the total number of locale-and-variant combinations available for a given operation class. You should call `LocaleOperationCountLocales` before the function `LocaleOperationGetLocales` (page 12) in order to determine how much memory to allocate for the list `LocaleOperationGetLocales` produces.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleOperationCountNames

Obtains the total count of all available localized names for an operation class.

```
OSStatus LocaleOperationCountNames (
   LocaleOperationClass opClass,
   ItemCount *nameCount
);
```

Parameters

opClass

A `LocaleOperationClass` value identifying the operation class for which you are requesting a localized name.

nameCount

A pointer to an `ItemCount` value. On return, the value is set to the total count of all available localized names for the operation class.

Return Value

A result code. The `LocaleOperationCountNames` function returns `paramErr` if `nameCount` is `NULL`.

Discussion

This function provides a count of all available localized names for a given operation class, for use in the function `LocaleOperationGetIndName` (page 11). You must call this function prior to calling `LocaleOperationGetIndName`.

Special Considerations

This function can move memory in Carbon and Mac OS 9.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleOperationGetIndName

Obtains a localized name for an operation class, based on an index into the list of all available localized names.

```
OSStatus LocaleOperationGetIndName (
   LocaleOperationClass opClass,
   ItemCount nameIndex,
   UniCharCount maxNameLen,
   UniCharCount *actualNameLen,
   UniChar displayName[],
   LocaleRef *displayLocale
);
```

Parameters

opClass

A `LocaleOperationClass` value identifying the operation class for which you are requesting a localized name.

nameIndex

An `ItemCount` index value identifying an entry in the list of available localized names. You can obtain a count of all available localized names from the `nameCount` parameter of the function `LocaleOperationCountNames` (page 10). Note that index values must be in the range from 0 to `nameCount` -1. To create a list of all available localized names that correspond to the requested operation class, you can call the `LocaleOperationGetIndName` function starting with an index value of 0, then increment the index value with each successive call to `LocaleOperationGetIndName`, continuing to `nameCount`-1.

maxNameLen

A `UniCharCount` value specifying the length of the `UniChar` array being passed in the `displayName` parameter.

actualNameLen

A pointer to a `UniCharCount` value. On return, the value contains the actual length of the name produced in the `displayName` parameter.

displayName

An array of `UniChar` values. On return, the array contains the requested name as localized for a particular display locale.

displayLocale

A pointer to a `LocaleRef` value. On return, the value indicates the language of the name produced in the `displayName` parameter.

Return Value

A result code. If the value specified in the `maxNameLen` parameter is too small for the requested string, `LocaleOperationGetIndName` returns `kLocalesBufferTooSmallErr` (-30001).

Discussion

The `LocaleOperationGetIndName` function obtains a localized name for an operation class based on an index into the count of all available localized names for the operation class. The language for the obtained name is produced on return.

You can use repeated calls to the `LocaleOperationGetIndName` function to create a list of all available localized names (that is, the names from all available display locales), that correspond to a given operation class. Alternately, for a particular operation class, the function `LocaleOperationGetName` (page 13) obtains the corresponding name as localized for a particular display locale.

Special Considerations

This function can move memory in Carbon and Mac OS 9.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleOperationGetLocales

Obtains the list of available locale-and-variant combinations for a given class of operation.

```
OSStatus LocaleOperationGetLocales (
   LocaleOperationClass opClass,
   ItemCount maxLocaleCount,
   ItemCount *actualLocaleCount,
   LocaleAndVariant localeVariantList[]
);
```

Parameters

opClass

A `LocaleOperationClass` value identifying the operation class for which you are requesting availability information.

maxLocaleCount

An `ItemCount` value specifying the size of the array passed in the `localeVariantList` parameter. To determine how much memory to allocate for the list, you can call the function `LocaleOperationCountLocales` (page 10). `LocaleOperationCountLocales` produces a count of the locale-and-variant combinations available for a given operation class in its `localeCount` parameter.

actualLocaleCount

A pointer to an `ItemCount` value. On return, the value contains the actual length of the list produced in the `localeVariantList` parameter.

localeVariantList

An array of `LocaleAndVariant` values. On return, the array contains a list of the available combinations of locale and operation variant for a specified operation class. The `LocaleAndVariant.opVariant` field may be 0 for entries that do not have a specific operation variant.

Return Value

A result code. If the value specified in the `maxLocaleCount` parameter is too small for the complete list, `LocaleOperationGetLocales` returns `kLocalesBufferTooSmallErr` (-30001). The function returns `paramErr` if `opClass` is 0 or if either the `actualLocaleCount` or `localeVariantList` parameter is `NULL`.

Discussion

The `LocaleOperationGetLocales` function provides a list of all the combinations of locales and operation variants that are available for a given class of operations, such as collation.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleOperationGetName

Obtains the localized name for an operation class, based on the requested language for the localized name.

```
OSStatus LocaleOperationGetName (
   LocaleOperationClass opClass,
   LocaleRef displayLocale,
   UniCharCount maxNameLen,
   UniCharCount *actualNameLen,
   UniChar displayName[]
);
```

Parameters

opClass

A `LocaleOperationClass` value identifying the operation class for which you are requesting a localized name.

displayLocale

A `LocaleRef` value indicating the requested language for the name produced in the `displayName` parameter. If `LocaleOperationGetName` cannot find a name in the requested language, it produces a name in the default display locale.

maxNameLen

A `UniCharCount` value specifying the length of the `UniChar` array being passed in the `displayName` parameter. An array of 64 characters is typically adequate.

actualNameLen

A pointer to a `UniCharCount` value. On return, the value contains the actual length of the name produced in the `displayName` parameter.

displayName

An array of `UniChar` values. On return, the array contains the requested name as localized for a particular display locale.

Return Value

A result code. If `LocaleOperationGetName` cannot find a name that matches the language specified in the `displayLocale` parameter, it returns `kLocalesDefaultDisplayStatus` (-30029). If the value specified in the `maxNameLen` parameter is too small for the requested string, `LocaleOperationGetName` returns `kLocalesBufferTooSmallErr` (-30001).

Discussion

For a particular operation class, the `LocaleOperationGetName` function obtains the corresponding name as localized for a particular display locale. Alternately, you can use repeated calls to the function `LocaleOperationGetIndName` (page 11) to iterate through all of the available display locales to create a list of the corresponding names as localized in each of the display locales.

Special Considerations

This function can move memory in Carbon and Mac OS 9.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleRefFromLangOrRegionCode

Converts a Mac OS language or region code to a locale reference.

```
OSStatus LocaleRefFromLangOrRegionCode (
   LangCode lang,
   RegionCode region,
   LocaleRef *locale
);
```

Parameters

lang

A Mac OS `LangCode` value, specifying the language for which to create a `LocaleRef`. If you wish to specify only a region, not a language as well, you can pass the constant `kTextLanguageDontCare` (from TextCommon.h). The `LocaleRefFromLangOrRegionCode` function requires values for either the `lang` or `region` parameters, or both.

region

A Mac OS `RegionCode` value, specifying the region for which to create a `LocaleRef`. If you wish to specify only a language, not a region as well, you can pass the constant `kTextRegionDontCare` (from TextCommon.h). The `LocaleRefFromLangOrRegionCode` function requires values for either the `lang` or `region` parameters, or both.

locale

A pointer to a value of type `LocaleRef`. On return, the `LocaleRef` value contains a valid reference to locale data. Note that the `LocaleRefFromLangOrRegionCode` function can move memory, so the `locale` parameter should not point to memory that can move.

Return Value

A result code. The `LocaleRefFromLangOrRegionCode` function returns `paramErr` if no value is provided for either the `lang` or `region` parameters, or if both values are provided but they are inconsistent, or if a value provided is invalid. The function returns `paramErr` if the `locale` parameter is `NULL`. It can also return memory errors or resource errors. Finally, if the tables used for mapping language and region codes are invalid, the function returns `kLocalesTableFormatErr` (-30002) in CarbonLib and Mac OS 9 (in Mac OS 8.6, `paramErr` is returned).

Discussion

A `LocaleRef` is an opaque type that references static locale data. You typically provide a `LocaleRef` value to the locale-sensitive Unicode Utilities functions and to some Locale Utilities functions such as `LocaleGetName` (page 8) and `LocaleOperationGetName` (page 13).

You can use the `LocaleRefFromLangOrRegionCode` function to convert a Mac OS language and/or region code to a `LocaleRef` value. Additionally, you can use the function `LocaleRefFromLocaleString` (page 16) to convert a locale part string—or an Internet language tag or POSIX/Java locale string—to a `LocaleRef` value.

You should not save `LocaleRef` values in a file or other persistent storage, because a given `LocaleRef` is not guaranteed to be valid across multiple launches of your application. For persistent storage, you can either save the original value or string used to construct the `LocaleRef`, or you can use the function `LocaleRefGetPartString` (page 16) to convert a `LocaleRef` to a locale part string and save that.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleRefFromLocaleString

Converts a string containing locale data to a locale reference.

```
OSStatus LocaleRefFromLocaleString (
   const char localeString[],
   LocaleRef *locale
);
```

Parameters

localeString

An ASCII string containing an Internet RFC 1766-style language tag, or a POSIX-style or Java-style locale string, or a Mac OS locale part string. A locale part string is an ASCII string whose full form is "lan-var.sc-v_rg-v", where "lan-var" specifies the language and variant, ".sc-v" specifies the script & variant, and "_rg-v" specifies the region and variant. Any of those three parts can be omitted furthermore, the variant part "-var" or "-v" within any of those parts may be omitted.

locale

A pointer to a value of type `LocaleRef`. On return, the `LocaleRef` value contains a valid reference to locale data. Note that the `LocaleRefFromLocaleString` function can move memory, so the `locale` parameter should not point to memory that can move.

Return Value

A result code. The function returns `paramErr` if the `localeString` or `locale` parameters are `NULL`, or if the value provided in `localeString` is invalid. It can also return memory errors.

Discussion

A `LocaleRef` is an opaque type that references static locale data. You typically provide a `LocaleRef` value to the locale-sensitive Unicode Utilities functions and to some Locale Utilities functions such as `LocaleGetName` (page 8) and `LocaleOperationGetName` (page 13).

You can use the `LocaleRefFromLocaleString` function to convert a locale part string—or an Internet language tag or POSIX/Java locale string—to a `LocaleRef` value. Additionally, you can use the function `LocaleRefFromLangOrRegionCode` (page 14) to convert a Mac OS language and/or region code to a `LocaleRef` value.

You should not save `LocaleRef` values in a file or other persistent storage, because a given `LocaleRef` is not guaranteed to be valid across multiple launches of your application. For persistent storage, you can either save the original value or string used to construct the `LocaleRef`, or you can use the function `LocaleRefGetPartString` (page 16) to convert a `LocaleRef` back to a locale part string and save that.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

LocaleRefGetPartString

Converts a locale reference to a string containing locale data.

```
OSStatus LocaleRefGetPartString (
   LocaleRef locale,
   LocalePartMask partMask,
   ByteCount maxStringLen,
   char partString[]
);
```

Parameters

`locale`
: The `LocaleRef` value to convert to string format. You can pass `NULL` for the default system locale.

`partMask`
: A `LocalePartMask` value specifying the kinds of locale information to include in the string. Available kinds of locale information include language, script, region, and their respective variants.

`maxStringLen`
: A `ByteCount` value specifying the length of the `char` array being passed in the `partString` parameter. The amount of storage should typically be at least 4 times the number of parts requested.

`partString`
: An array of `char` values. On return, the array contains the new part string. The full form of the returned string is "lan-var.sc-v_rg-v" where "lan-var" specifies the language and variant, ".sc-v" specifies the script & variant, and "_rg-v" specifies the region and variant. Fields not specified by the `partMask` parameter (as well as any field separator that precedes them) are not included in the returned string.

Return Value
A result code. The function returns `paramErr` if the `partString` parameter is `NULL`, or if the locale is invalid. It returns `kLocalesBufferTooSmallErr` (-30001) if the value supplied in the `maxStringLen` parameter is too small for the requested string.

Discussion
The `LocaleRefGetPartString` function converts a `LocaleRef` value to a string containing locale information. You can use a locale part string to tag or specify language or locale in persistent storage.

Availability
Available in CarbonLib 1.0 and later when running Mac OS 8.6 or later.

Available in Mac OS X 10.0 and later.

Declared In
`MacLocales.h`

LocaleStringToLangAndRegionCodes

Converts a string containing locale data to Mac OS language and region codes.

```
OSStatus LocaleStringToLangAndRegionCodes (
   const char localeString[],
   LangCode *lang,
   RegionCode *region
);
```

Parameters

localeString

An ASCII string containing an Internet RFC 1766-style language tag, or a POSIX-style or Java-style locale string, or a Mac OS locale part string. If you have a `LocaleRef` value, you can obtain a locale string from the function `LocaleRefGetPartString` (page 16). A locale part string is an ASCII string whose full form is "lan-var.sc-v_rg-v", where "lan-var" specifies the language and variant, ".sc-v" specifies the script & variant, and "_rg-v" specifies the region and variant. Any of those three parts can be omitted furthermore, the variant part "-var" or "-v" within any of those parts may be omitted.

lang

A pointer to a `LangCode` value, or pass `NULL`, if you do not want to obtain a language code. On return, `LocaleStringToLangAndRegionCodes` produces the appropriate Mac OS language code for the locale part string provided in the `localeString` parameter. Note that you can set either the `lang` or the `region` parameter to `NULL`, but not both.

region

A pointer to a `RegionCode` value or pass `NULL`, if you do not want to obtain a region code. On return, `LocaleStringToLangAndRegionCodes` produces the appropriate Mac OS region code for the locale part string provided in the `localeString` parameter. Note that you can set either the `region` or the `lang` parameter to `NULL`, but not both.

Return Value

A result code. If the `localeString` parameter is `NULL` or if the string cannot be mapped to an existing language code and region code, the `LocaleStringToLangAndRegionCodes` function returns `paramErr`. If the required resource is invalid, the function can return `kLocalesTableFormatErr` (-30002). The function can also return Resource Manager errors.

Discussion

The `LocaleStringToLangAndRegionCodes` function maps from a locale string to a combination of Mac OS language code and region code.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 9 or later.

Available in Mac OS X 10.0 and later.

Declared In

`MacLocales.h`

Data Types

LocaleAndVariant

Identifies a specific locale and/or sub-class of locale-sensitive operations.

```
struct LocaleAndVariant {
    LocaleRef locale;
    LocaleOperationVariant opVariant;
};
typedef struct LocaleAndVariant LocaleAndVariant;
```

Fields

`locale`

A `LocaleRef` value, encapsulating information regarding language, script, and/or region.

`opVariant`

A `LocaleOperationVariant` value, identifying an individual sub-class of locale-sensitive operations. The `opVariant` field of a `LocaleAndVariant` value may be 0 for entries that do not have a specific operation variant.

Discussion

You use the function `LocaleOperationGetLocales` (page 12) to obtain a list of all the locale-and-variant combinations available for a given class of operations. The `LocaleOperationGetLocales` function specifies these combinations using the `LocaleAndVariant` type.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`MacLocales.h`

LocaleNameMask

Specifies named parts of locale data.

```
typedef UInt32 LocaleNameMask;
```

Discussion

The bits set in a `LocaleNameMask` value determine the kind of localized names that are produced by the functions `LocaleGetName` (page 8) and `LocaleGetIndName` (page 7). For a description of the `LocaleNameMask` values, see "Locale Name Masks" (page 21).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`MacLocales.h`

LocaleOperationClass

Identifies individual classes of locale-sensitive operations.

```
typedef FourCharCode LocaleOperationClass;
```

Discussion

Locale-sensitive Unicode Utilities operations fall into several classes, such as collation, text break determination, and date and time formatting. You use the `LocaleOperationClass` type to specify these classes.

Specific `LocaleOperationClass` values depend on the Locale Utilities client. Values from Unicode Utilities include `kUnicodeCollationClass ('ucol')` and `kUnicodeTextBreakClass ('ubrk')`. For example, the locales and collation variants available for collation operations can be determined by calling the functions `LocaleOperationCountLocales` (page 10) and `LocaleOperationGetLocales` (page 12) with the `opClass` parameter set to the `kUnicodeCollationClass` constant.

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

Declared In
`MacLocales.h`

LocaleOperationVariant

Identifies individual sub-classes of locale-sensitive operations.

```
typedef FourCharCode LocaleOperationVariant;
```

Discussion
Locale-sensitive Unicode Utilities operations fall into several classes, such as collation, text break determination, and date and time formatting. You use the type `LocaleOperationClass` (page 19) to specify these classes. Some of these classes also have sub-classes of their operations. For example, sub-classes of collation operations include dictionary vs. bibliographic collation, and radical-stroke vs. pinyin collation.

For classes such as collation, the `LocaleOperationVariant` type provides an operation-specific variant field. This field is analogous to the `@variant` part of a POSIX locale string or the final `_VARIANT` part of a Java locale string.

You can use `LocaleOperationVariant` values in the functions `LocaleGetName` (page 8) , `LocaleGetIndName` (page 7) , and `LocaleCountNames` (page 6) to obtain localized names for operation variants.

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

Declared In
`MacLocales.h`

LocalePartMask

Specifies kinds of locale data.

```
typedef UInt32 LocalePartMask;
```

Discussion
The bits set in a `LocalePartMask` value determine the kinds of locale information that are produced by the function `LocaleRefGetPartString` (page 16). For a description of the `LocalePartMask` values, see "Locale Part Masks" (page 22).

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

Declared In
`MacLocales.h`

LocaleRef

An opaque type that refers to locale information.

```
typedef struct OpaqueLocaleRef * LocaleRef;
```

Discussion
A `LocaleRef` is an opaque type that references static locale data. You typically provide a `LocaleRef` value to the locale-sensitive Unicode Utilities functions and to some Locale Utilities functions such as `LocaleGetName` (page 8) and `LocaleOperationGetName` (page 13).

To obtain a `LocaleRef`, you can use the function `LocaleRefFromLangOrRegionCode` (page 14) to convert a Mac OS language and/or region code to a `LocaleRef` value. Additionally, you can use the function `LocaleRefFromLocaleString` (page 16) to convert a locale part string—or an Internet language tag or POSIX/Java locale string—to a `LocaleRef` value.

You should not save `LocaleRef` values in a file or other persistent storage, since a given `LocaleRef` is not guaranteed to be valid across multiple launches of your application. For persistent storage, you can either save the original value or string used to construct the `LocaleRef`, or you can use the function `LocaleRefGetPartString` (page 16) to convert a `LocaleRef` to a locale part string and save that.

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

Declared In
`MacLocales.h`

Constants

Locale Name Masks

Specify named parts of locale data.

```
enum {
    kLocaleNameMask = 1L << 0,
    kLocaleOperationVariantNameMask = 1L << 1,
    kLocaleAndVariantNameMask = 0x00000003
};
```

Constants
`kLocaleNameMask`
   If the bit specified by this mask is set, then the name for a locale is specified.

   Available in Mac OS X v10.0 and later.

   Declared in `MacLocales.h`.

`kLocaleOperationVariantNameMask`
: If the bit specified by this mask is set, then the name for an operation variant is specified.

  Available in Mac OS X v10.0 and later.

  Declared in `MacLocales.h`.

`kLocaleAndVariantNameMask`
: If the bits specified by this mask are set, then the name for a locale-and-variant combination is specified.

  Available in Mac OS X v10.0 and later.

  Declared in `MacLocales.h`.

Discussion

The bits set in a `LocaleNameMask` value determine the kind of localized names that are produced by the functions `LocaleGetName` (page 8) and `LocaleGetIndName` (page 7). With a `LocaleNameMask` value, you can specify the name of a locale alone, the name of an operation variant alone, or the name for the combination of the two.

Locale Part Masks

Specify kinds of locale data.

```
enum {
    kLocaleLanguageMask = 1L << 0,
    kLocaleLanguageVariantMask = 1L << 1,
    kLocaleScriptMask = 1L << 2,
    kLocaleScriptVariantMask = 1L << 3,
    kLocaleRegionMask = 1L << 4,
    kLocaleRegionVariantMask = 1L << 5,
    kLocaleAllPartsMask = 0x0000003F
};
```

Constants

`kLocaleLanguageMask`
: If the bit specified by this mask is set, then a ISO 639-1 or -2 language code is specified.

  Available in Mac OS X v10.0 and later.

  Declared in `MacLocales.h`.

`kLocaleLanguageVariantMask`
: If the bit specified by this mask is set, then a custom string for a language variant is specified.

  Available in Mac OS X v10.0 and later.

  Declared in `MacLocales.h`.

`kLocaleScriptMask`
: If the bit specified by this mask is set, then a ISO 15924 script code is specified.

  Available in Mac OS X v10.0 and later.

  Declared in `MacLocales.h`.

`kLocaleScriptVariantMask`
: If the bit specified by this mask is set, then a custom string for a script variant is specified.

  Available in Mac OS X v10.0 and later.

  Declared in `MacLocales.h`.

`kLocaleRegionMask`

If the bit specified by this mask is set, then a ISO 3166 country/region code is specified.

Available in Mac OS X v10.0 and later.

Declared in `MacLocales.h`.

`kLocaleRegionVariantMask`

If the bit specified by this mask is set, then a custom string for a region variant is specified.

Available in Mac OS X v10.0 and later.

Declared in `MacLocales.h`.

`kLocaleAllPartsMask`

If the bits specified by this mask are set, then all types (language and variant, script and variant, and region and variant) of locale information are specified.

Available in Mac OS X v10.0 and later.

Declared in `MacLocales.h`.

Discussion

The bits set in a `LocalePartMask` value determine the kinds of locale information that are produced by the function `LocaleRefGetPartString` (page 16).

Deprecated Locale Utilities Functions

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

Deprecated in Mac OS X v10.5

LocaleGetRegionLanguageName

Obtains the localized language name for a region. (Deprecated in Mac OS X v10.5.)

```
OSStatus LocaleGetRegionLanguageName (
   RegionCode region,
   Str255 languageName
);
```

Parameters

region

A Mac OS `RegionCode` value, specifying the region for which you are requesting a localized language name.

languageName

A Pascal string. On return, the string contains the name of the language corresponding to the region specified in the `region` parameter. The language name is produced in its own language and in the appropriate Mac OS encoding for that region.

Return Value

A result code. The `LocaleGetRegionLanguageName` function returns `paramErr` if `languageName` is `NULL` or if `region` is invalid (

Discussion

For a particular Mac OS region code, the `LocaleGetRegionLanguageName` function returns the name of the corresponding language in that language and in the non-Unicode Mac OS text encoding used for that region.

Note that, in some languages, the name for a language or locale may have several different grammatical forms, where the correct form depends on the usage or context. For example, Swedish uses different forms of a language name depending on whether the name is applied to a collation order, to text break rules, or to keyboard layouts. However, the Locale Utilities functions only return one form of a language or locale name—typically that name which would be used for the language name alone. Therefore, this name may not be the correct form for some usages in some languages.

Special Considerations

This function can move memory in Carbon and Mac OS 9.

Availability

Available in CarbonLib 1.0 and later when running Mac OS 9 or later.

Available in Mac OS X 10.0 and later.

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

Declared In
`MacLocales.h`

Document Revision History

This table describes the changes to *Locale Utilities Reference*.

Date	Notes
2003-01-24	Updated formatting.

Index

K

`kLocaleAllPartsMask` constant 23
`kLocaleAndVariantNameMask` constant 22
`kLocaleLanguageMask` constant 22
`kLocaleLanguageVariantMask` constant 22
`kLocaleNameMask` constant 21
`kLocaleOperationVariantNameMask` constant 22
`kLocaleRegionMask` constant 23
`kLocaleRegionVariantMask` constant 23
`kLocaleScriptMask` constant 22
`kLocaleScriptVariantMask` constant 22

L

Locale Name Masks 21
Locale Part Masks 22
`LocaleAndVariant` structure 18
`LocaleCountNames` function 6
`LocaleGetIndName` function 7
`LocaleGetName` function 8
`LocaleGetRegionLanguageName` function (Deprecated in Mac OS X v10.5) 25
`LocaleNameMask` data type 19
`LocaleOperationClass` data type 19
`LocaleOperationCountLocales` function 10
`LocaleOperationCountNames` function 10
`LocaleOperationGetIndName` function 11
`LocaleOperationGetLocales` function 12
`LocaleOperationGetName` function 13
`LocaleOperationVariant` data type 20
`LocalePartMask` data type 20
`LocaleRef` data type 21
`LocaleRefFromLangOrRegionCode` function 14
`LocaleRefFromLocaleString` function 16
`LocaleRefGetPartString` function 16
`LocaleStringToLangAndRegionCodes` function 17