CFMutableString Reference

Core Foundation

2008-10-15

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

CFMutableString Reference

Derived From:	CFString : CFPropertyList : CFType
Framework:	CoreFoundation/CoreFoundation.h
Declared in	CFBase.h CFString.h
Companion guides	Property List Programming Topics for Core Foundation Strings Programming Guide for Core Foundation

Overview

CFMutableString manages dynamic strings. The basic interface for managing strings is provided by CFString. CFMutableString adds functions to modify the contents of a string.

CFMutableString is "toll-free bridged" with its Cocoa Foundation counterpart, `NSMutableString`. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSMutableString *` parameter, you can pass in a `CFMutableStringRef`, and in a function where you see a `CFMutableStringRef` parameter, you can pass in an NSMutableString instance. This also applies to concrete subclasses of NSMutableString. See Interchangeable Data Types for more information on toll-free bridging.

Functions

CFStringAppend

Appends the characters of a string to those of a CFMutableString object.

```
void CFStringAppend (
   CFMutableStringRef theString,
   CFStringRef appendedString
);
```

Parameters

theString

The string to which *appendedString* is appended. If *theString* is not a CFMutableString object, an assertion is raised.

appendedString

The string to append.

Discussion
This function reallocates the backing store of *theString* to accommodate the new length.

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

Related Sample Code
KillEveryOneButMe
MovieVideoChart
ProfileSystem
QTMetaData
TypeServicesForUnicode

Declared In
CFString.h

CFStringAppendCharacters

Appends a buffer of Unicode characters to the character contents of a CFMutableString object.

```
void CFStringAppendCharacters (
   CFMutableStringRef theString,
   const UniChar *chars,
   CFIndex numChars
);
```

Parameters

theString
The string to which the characters in *chars* are appended.

chars
A pointer to a buffer of Unicode characters.

numChars
The number of Unicode characters in *chars*.

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

Declared In
CFString.h

CFStringAppendCString

Appends a C string to the character contents of a CFMutableString object.

```
void CFStringAppendCString (
   CFMutableStringRef theString,
   const char *cStr,
   CFStringEncoding encoding
);
```

Parameters

theString

The string to which the characters from *cStr* are appended. If this value is not a CFMutableString object, an assertion is raised.

cStr

A pointer to a C string buffer.

encoding

The encoding of the characters in *cStr*.

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

Related Sample Code
audioburntest

simpleJavaLauncher

Declared In
`CFString.h`

CFStringAppendFormat

Appends a formatted string to the character contents of a CFMutableString object.

```
void CFStringAppendFormat (
   CFMutableStringRef theString,
   CFDictionaryRef formatOptions,
   CFStringRef format,
   ...
);
```

Parameters

theString

The string to which the formatted characters from *format* are appended. If this value is not a CFMutableString object, an assertion is raised.

formatOptions

A dictionary containing formatting options for the string (such as the thousand-separator character, which is dependent on locale). Currently, these options are an unimplemented feature.

format

A formatted string with `printf`-style specifiers.

...

Variable list of the values to be inserted in *format*.

Discussion
A formatted string is one with `printf`-style format specifiers embedded in the text such as `%d` (decimal), `%f` (double), and `%@` (Core Foundation object). The subsequent arguments, in order, are substituted for the specifiers in the character data appended to *`theString`*. You can also reorder the arguments in the string by using modifiers of the form "n$" with the format specifiers (for example, `%2$d`).

For more information on supported specifiers, see the relevant section in *Strings Programming Guide for Core Foundation*.

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

Related Sample Code
Custom_HIView_Tutorial
DockBrowser
GetHWEthernetAddr
MoreIsBetter
MovieVideoChart

Declared In
`CFString.h`

CFStringAppendFormatAndArguments

Appends a formatted string to the character contents of a CFMutableString object.

```
void CFStringAppendFormatAndArguments (
   CFMutableStringRef theString,
   CFDictionaryRef formatOptions,
   CFStringRef format,
   va_list arguments
);
```

Parameters

`theString`
The string to which the formatted characters from *`format`* are appended. If this value is not a CFMutableString object, an assertion is raised.

`formatOptions`
A dictionary containing formatting options for the string (such as the thousand-separator character, which is dependent on locale). Currently, these options are an unimplemented feature.

`format`
A formatted string with `printf`-style specifiers.

`arguments`
List of values to be inserted in *`format`*.

Discussion
A formatted string is one with `printf`-style format specifiers embedded in the text such as `%d` (decimal), `%f` (double), and `%@` (Core Foundation object). The subsequent arguments, in order, are substituted for the specifiers in the character data appended to *`theString`*. You can also reorder the arguments in the string by using modifiers of the form "n$" with the format specifiers (for example, `%2$d`).

For more information on supported specifiers, see the relevant section in *Strings Programming Guide for Core Foundation*.

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

Declared In
`CFString.h`

CFStringAppendPascalString

Appends a Pascal string to the character contents of a CFMutableString object.

```
void CFStringAppendPascalString (
   CFMutableStringRef theString,
   ConstStr255Param pStr,
   CFStringEncoding encoding
);
```

Parameters

theString
The string to which the characters in *pStr* are appended. If this value is not a CFMutableString object, an assertion is raised.

pStr
A Pascal string buffer.

encoding
The string encoding of the characters in *pStr*.

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

Declared In
`CFString.h`

CFStringCapitalize

Changes the first character in each word of a string to uppercase (if it is a lowercase alphabetical character).

```
void CFStringCapitalize (
   CFMutableStringRef theString,
   CFLocaleRef locale
);
```

Parameters

theString
The string to be capitalized. If this value is not a CFMutableString object, an assertion is raised.

locale
A locale that specifies a particular language or region. Prior to Mac OS X v10.3, this parameter was an untyped pointer and not used.

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

Declared In
CFString.h

CFStringCreateMutable

Creates an empty CFMutableString object.

```
CFMutableStringRef CFStringCreateMutable (
   CFAllocatorRef alloc,
   CFIndex maxLength
);
```

Parameters

alloc

The allocator to use to allocate memory for the new string. Pass NULL or kCFAllocatorDefault to use the current default allocator.

maxLength

The maximum number of Unicode characters that can be stored by the returned string. Pass 0 if there should be no character limit. Note that initially the string still has a length of 0; this parameter simply specifies what the maximum size is. CFMutableString might try to optimize its internal storage by paying attention to this value.

Return Value
A new empty CFMutableString object or NULL if there was a problem creating the object. Ownership follows the Create Rule.

Discussion
This function creates an empty (that is, content-less) CFMutableString object. You can add character data to this object with any of the CFStringAppend... functions, and thereafter you can insert, delete, replace, pad, and trim characters with the appropriate CFString functions. If the *maxLength* parameter is greater than 0, any attempt to add characters beyond this limit results in a run-time error.

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

Related Sample Code
iTunesController
KillEveryOneButMe
MovieVideoChart
QTMetaData
simpleJavaLauncher

Declared In
CFString.h

CFStringCreateMutableCopy

Creates a mutable copy of a string.

```
CFMutableStringRef CFStringCreateMutableCopy (
   CFAllocatorRef alloc,
   CFIndex maxLength,
   CFStringRef theString
);
```

Parameters

alloc

The allocator to use to allocate memory for the new string. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

maxLength

The maximum number of Unicode characters that can be stored by the returned object. Pass `0` if there should be no character limit. Note that initially the returned object still has the same length as the string argument; this parameter simply specifies what the maximum size is. CFString might try to optimize its internal storage by paying attention to this value.

theString

A string to copy.

Return Value

A string that has the same contents as *theString*. Returns `NULL` if there was a problem copying the object. Ownership follows the Create Rule.

Discussion

The returned mutable string is identical to the original string except for (perhaps) the mutability attribute. You can add character data to the returned string with any of the `CFStringAppend...` functions, and you can insert, delete, replace, pad, and trim characters with the appropriate CFString functions. If the *maxLength* parameter is greater than `0`, any attempt to add characters beyond this limit results in a run-time error.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Custom_HIView_Tutorial

DTSCarbonShell

iTunesController

MFSLives

MoreSCF

Declared In

`CFString.h`

CFStringCreateMutableWithExternalCharactersNoCopy

Creates a CFMutableString object whose Unicode character buffer is controlled externally.

```
CFMutableStringRef CFStringCreateMutableWithExternalCharactersNoCopy (
   CFAllocatorRef alloc,
   UniChar *chars,
   CFIndex numChars,
   CFIndex capacity,
   CFAllocatorRef externalCharactersAllocator
);
```

Parameters

alloc

The allocator to use to allocate memory for the string. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

chars

The Unicode character buffer for the new CFMutableString. Before calling, create this buffer on the stack or heap and optionally initialize it with Unicode character data. Upon return, the created CFString object keeps its own copy of the pointer to this buffer. You may pass in `NULL` if there is no initial buffer being provided.

numChars

The number of characters initially in the Unicode buffer pointed to by *chars*.

capacity

The capacity of the external buffer (*chars*); that is, the maximum number of Unicode characters that can be stored. This value should be `0` if no initial buffer is provided.

externalCharactersAllocator

The allocator to use to reallocate the external buffer when editing takes place and for deallocating the buffer when string is deallocated. If the default allocator is suitable for these purposes, pass `NULL`. If you do not want the new string to reallocate or deallocate memory for the buffer (that is, you assume responsibility for these things yourself), pass `kCFAllocatorNull`.

Return Value

A new mutable string, or `NULL` if there was a problem creating the object. Ownership follows the Create Rule.

Discussion

This function permits you to create a CFMutableString object whose backing store is an external Unicode character buffer—that is, a buffer that you control (or can control) entirely. This function allows you to take advantage of the features of CFString, particularly the CFMutableString functions that add and modify character data. But at the same time you can directly add, delete, modify, and examine the characters in the buffer. You can even replace the buffer entirely. If, however, you directly modify or replace the character buffer, you should inform the CFString object of this change with the `CFStringSetExternalCharactersNoCopy` (page 19) function.

If you mutate the character contents with the CFString functions, and the buffer needs to be enlarged, the CFString object calls the allocation callbacks specified for the allocator *externalCharactersAllocator*.

This function should be used in special circumstances where you want to create a CFString wrapper around an existing, potentially large `UniChar` buffer you own. Using this function causes the CFString object to forgo some of its internal optimizations, so it should be avoided in general use. That is, if you want to create a CFString object from a small `UniChar` buffer, and you don't need to continue owning the buffer, use one of the other creation functions (for instance `CFStringCreateWithCharacters`) instead.

Availability

Available in Mac OS X v10.0 and later.

Declared In
`CFString.h`

CFStringDelete

Deletes a range of characters in a string.

```
void CFStringDelete (
   CFMutableStringRef theString,
   CFRange range
);
```

Parameters

theString

A string from which characters are to be deleted.

range

The range of characters in *theString* to delete.

Discussion
The characters after the deleted range are adjusted to "fill in" the gap.

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

Related Sample Code
DockBrowser
HID Calibrator
HID Config Save
HID Explorer

Declared In
`CFString.h`

CFStringFindAndReplace

Replaces all occurrences of a substring within a given range.

```
CFIndex CFStringFindAndReplace (
   CFMutableStringRef theString,
   CFStringRef stringToFind,
   CFStringRef replacementString,
   CFRange rangeToSearch,
   CFOptionFlags compareOptions
);
```

Parameters

theString

The string to modify.

stringToFind

The substring to search for in `theString`.

replacementString

The replacement string for *stringToFind*.

rangeToSearch

The range within which to search in *theString*.

compareOptions

Flags that select different types of comparisons, such as localized comparison, case-insensitive comparison, and non-literal comparison. If you want the default comparison behavior, pass `0`. See `CFStringCompareFlags` for the available flags.

Return Value
The number of instances of *stringToFind* that were replaced.

Discussion
The possible values of *compareOptions* are combinations of the `kCFCompareCaseInsensitive`, `kCFCompareBackwards`, `kCFCompareNonliteral`, and `kCFCompareAnchored` constants.

The `kCFCompareBackwards` option can be used to replace a substring starting from the end, which could produce different results. For example, if the parameter *theString* is "AAAAA", *stringToFind* is "AA", and *replacementString* is "B", then the result is normally "BBA". However, if the `kCFCompareBackwards` constant is used, the result is "ABB."

The `kCFCompareAnchored` option assures that only anchored but multiple instances are found (the instances must be consecutive at start or end). For example, if the parameter *theString* is "AAXAA", *stringToFind* is "A", and *replacementString* is "B", then the result is normally "BBXBB." However, if the `kCFCompareAnchored` constant is used, the result is "BBXAA."

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
simpleJavaLauncher

Declared In
`CFString.h`

CFStringFold

Folds a given string into the form specified by optional flags.

```
void CFStringFold (
    CFMutableStringRef theString,
    CFOptionFlags theFlags,
    CFLocaleRef theLocale
);
```

Parameters

theString

The string which is to be folded. If this parameter is not a valid mutable CFString, the behavior is undefined.

theFlags

The equivalency flags which describes the character folding form. See “String Comparison Flags” in *CFString Reference* for possible values. Only those flags containing the word "insensitive" are recognized; other flags are ignored.

Folding with `kCFCompareCaseInsensitive` removes case distinctions in accordance with the mapping specified by ftp://ftp.unicode.org/Public/UNIDATA/CaseFolding.txt. Folding with `kCFCompareDiacriticInsensitive` removes distinctions of accents and other diacritics. Folding with `kCFCompareWidthInsensitive` removes character width distinctions by mapping characters in the range `U+FF00-U+FFEF` to their ordinary equivalents.

theLocale

The locale to use for the operation. `NULL` specifies the canonical locale (the return value from `CFLocaleGetSystem`).

The locale argument affects the case mapping algorithm. For example, for the Turkish locale, case-insensitive compare matches “I” to “ı” (Unicode code point U+0131, Latin Small Dotless I), not the normal “i” character.

Discussion
Character foldings are operations that convert any of a set of characters sharing similar semantics into a single representative from that set.

You can use this function to preprocess strings that are to be compared, searched, or indexed. Note that folding does not include normalization, so you must use `CFStringNormalize` (page 16) in addition to CFStringFold in order to obtain the effect of `kCFCompareNonliteral`.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
DerivedProperty

Declared In
`CFString.h`

CFStringInsert

Inserts a string at a specified location in the character buffer of a CFMutableString object.

```
void CFStringInsert (
   CFMutableStringRef str,
   CFIndex idx,
   CFStringRef insertedStr
);
```

Parameters

str

The string to be modified. If this value is not a CFMutableString object, an assertion is raised.

index

The index of the character in *str* after which the new characters are to be inserted. If the index is out of bounds, an assertion is raised.

insertedStr

The string to insert into *str*.

Discussion
To accommodate the new characters, this function moves any existing characters to the right of the inserted characters the appropriate number of positions.

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

Related Sample Code
HID Explorer
iTunesController

Declared In
`CFString.h`

CFStringLowercase

Changes all uppercase alphabetical characters in a CFMutableString to lowercase.

```
void CFStringLowercase (
   CFMutableStringRef theString,
   CFLocaleRef locale
);
```

Parameters

`theString`

The string to be lowercased. If this value is not a CFMutableString object, an assertion is raised.

`locale`

The locale to use when the lowercasing operation is performed. Prior to Mac OS X v10.3 this parameter was an untyped pointer and not used.

The locale argument affects the case mapping algorithm. For example, for the Turkish locale, case-insensitive compare matches "I" to "ı" (Unicode code point U+0131, Latin Small Dotless I), not the normal "i" character.

Special Considerations

The *`locale`* parameter type changed from `void *` to `CFLocaleRef` in Mac OS X v10.3.

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

Related Sample Code
MoreIsBetter
MoreSCF
NSLMiniBrowser
QISA

Declared In
`CFString.h`

CFStringNormalize

Normalizes the string into the specified form as described in Unicode Technical Report #15.

```
void CFStringNormalize (
   CFMutableStringRef theString,
   CFStringNormalizationForm theForm
);
```

Parameters

theString

The string to be normalized.

theForm

The form to normalize *theString*.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
DerivedProperty

MFSLives

MoreSCF

Declared In
CFString.h

CFStringPad

Enlarges a string, padding it with specified characters, or truncates the string.

```
void CFStringPad (
   CFMutableStringRef theString,
   CFStringRef padString,
   CFIndex length,
   CFIndex indexIntoPad
);
```

Parameters

theString

The string to modify.

padString

A string containing the characters with which to fill the extended character buffer. Pass `NULL` to truncate the string.

length

The new length of *theString*. If this length is greater than the current length, padding takes place; if it is less, truncation takes place.

indexIntoPad

The index of the character in *padString* with which to begin padding. If you are truncating the string represented by the object, this parameter is ignored.

Discussion
This function has two purposes. It either enlarges the character buffer of a CFMutableString object to a given length, padding the added length with a given character or characters, or it truncates the character buffer to a smaller size. The key parameter for this behavior is *length*; if it is greater than the current length of the represented string, padding takes place, and if it less than the current length, truncation occurs.

For example, say you have a string, `aMutStr`, containing the characters "abcdef". The call

```
CFStringPad(aMutStr, CFSTR("123"), 9, 1);
```

results in `aMutStr` containing "abcdef231". However, the following call

```
CFStringPad(aMutStr, NULL, 3, 0);
```

results in `aMutStr` containing "abc".

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

Declared In
`CFString.h`

CFStringReplace

Replaces part of the character contents of a CFMutableString object with another string.

```
void CFStringReplace (
   CFMutableStringRef theString,
   CFRange range,
   CFStringRef replacement
);
```

Parameters

`theString`

The string to modify. The characters are adjusted left or right (depending on the length of the substituted string) and the character buffer of the object is resized accordingly. If this value is not a CFMutableString object, an assertion is raised.

`range`

The range of characters in *`theString`* to replace.

`replacement`

The replacement string to put into *`theString`*.

Discussion
Although you can use this function to replace all characters in a CFMutableString object (by specifying a range of `(0, CFStringGetLength(theString))`), it is more convenient to use the `CFStringReplaceAll` (page 19) function for that purpose.

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

Related Sample Code
CIVideoDemoGL
MoreIsBetter
MoreSCF
QISA
SpellingChecker CarbonCocoa Bundled

Declared In
`CFString.h`

CFStringReplaceAll

Replaces all characters of a CFMutableString object with other characters.

```
void CFStringReplaceAll (
   CFMutableStringRef theString,
   CFStringRef replacement
);
```

Parameters

theString

The string to modify. If this value is not a CFMutableString object, an assertion is raised.

replacement

The replacement string to put into *theString*.

Discussion

The character buffer of *theString* is resized according to the length of the new characters.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ImageBrowserView

iTunesController

Declared In

CFString.h

CFStringSetExternalCharactersNoCopy

Notifies a CFMutableString object that its external backing store of Unicode characters has changed.

```
void CFStringSetExternalCharactersNoCopy (
   CFMutableStringRef theString,
   UniChar *chars,
   CFIndex length,
   CFIndex capacity
);
```

Parameters

theString

The string to act as a "wrapper" for the external backing store (*chars*). If this value is not a CFMutableString object, an assertion is raised.

chars

The external (client-owned) Unicode buffer acting as the backing store for *theString*.

length

The current length of the contents of *chars* (in Unicode characters).

capacity

The capacity of the Unicode buffer—that is, the total number of Unicode characters that can be stored in it before the buffer has to be grown.

Discussion
You use this function to reallocate memory for a string, if necessary, and change its references to the data in the buffer. The object must have been created with the `CFStringCreateMutableWithExternalCharactersNoCopy` (page 11) function; see the discussion of this function for more information.

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

Declared In
`CFString.h`

CFStringTransform

Perform in-place transliteration on a mutable string.

```
Boolean CFStringTransform (
   CFMutableStringRef string,
   CFRange *range,
   CFStringRef transform,
   Boolean reverse
);
```

Parameters

string
The string to transform.

range
A pointer to the range over which the transformation is applied. `NULL` causes the whole string to be transformed. On return, *range* is modified to reflect the new range corresponding to the original range.

transform
A CFString object that identifies the transformation to apply. For a list of valid values, see "Transform Identifiers for CFStringTransform" (page 23). On Mac OS X v10.4 and later, you can also use any valid ICU transform ID defined in the ICU User Guide for Transforms.

reverse
A Boolean that, if `true`, specifies that the inverse transform should be used (if it exists).

Return Value
`true` if the transform is successful; otherwise `false`.

Discussion
The transformation represented by *transform* is applied to the given range of *string*, modifying it in place. Only the specified range is modified, but the transform may look at portions of the string outside that range for context. Reasons that the transform may be unsuccessful include an invalid transform identifier, and attempting to reverse an irreversible transform.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`CFString.h`

CFStringTrim

Trims a specified substring from the beginning and end of a CFMutableString object.

```
void CFStringTrim (
   CFMutableStringRef theString,
   CFStringRef trimString
);
```

Parameters

theString

The string to trim. If this value is not a CFMutableString object, an assertion is raised.

trimString

The string to trim from *theString*. The characters of the trim string are treated as a substring and not individually; for example, if the mutable characters are "abc X" and the trim string is "XY", the mutable characters are not affected.

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

Related Sample Code
MoreSCF

Declared In
CFString.h

CFStringTrimWhitespace

Trims whitespace from the beginning and end of a CFMutableString object.

```
void CFStringTrimWhitespace (
   CFMutableStringRef theString
);
```

Parameters

theString

The string to trim. If this value is not a CFMutableString object, an assertion is raised.

Discussion
Whitespace for this function includes space characters, tabs, newlines, carriage returns, and any similar characters that do not have a visible representation.

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

Declared In
CFString.h

CFStringUppercase

Changes all lowercase alphabetical characters in a CFMutableString object to uppercase.

```
void CFStringUppercase (
   CFMutableStringRef theString,
   CFLocaleRef locale
);
```

Parameters

theString

The string to uppercase. If this value is not a CFMutableString object, an assertion is raised.

locale

A CFLocale object that specifies a particular language or region. Prior to Mac OS X v10.3, this parameter was an untyped pointer and not used.

The locale argument affects the case mapping algorithm. For example, for the Turkish locale, case-insensitive compare matches “I” to “ı” (Unicode code point U+0131, Latin Small Dotless I), not the normal “i” character.

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

Declared In
CFString.h

Data Types

CFMutableStringRef

A reference to a CFMutableString object.

```
typedef CFStringRef CFMutableStringRef;
```

Discussion
The type refers to a CFMutableString object, which “encapsulates” a Unicode string along with its length; the object has the attribute of being mutable, which means that its character contents can be modified. CFString is an opaque type that defines the characteristics and behavior of CFString objects, both immutable and mutable.

CFMutableString derives from CFString. Therefore, you can pass CFMutableString objects into functions accepting CFString objects.

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

Declared In
CFBase.h

Constants

String Normalization Forms

Unicode normalization forms as described in Unicode Technical Report #15.

```
enum CFStringNormalizationForm {
    kCFStringNormalizationFormD = 0,
    kCFStringNormalizationFormKD = 1,
    kCFStringNormalizationFormC = 2,
    kCFStringNormalizationFormKC = 3
};
typedef enum CFStringNormalizationForm CFStringNormalizationForm;
```

Constants

`kCFStringNormalizationFormD`
Canonical decomposition.
Available in Mac OS X v10.2 and later.
Declared in `CFString.h`.

`kCFStringNormalizationFormKD`
Compatibility decomposition.
Available in Mac OS X v10.2 and later.
Declared in `CFString.h`.

`kCFStringNormalizationFormC`
Canonical decomposition followed by canonical composition.
Available in Mac OS X v10.2 and later.
Declared in `CFString.h`.

`kCFStringNormalizationFormKC`
Compatibility decomposition followed by canonical composition.
Available in Mac OS X v10.2 and later.
Declared in `CFString.h`.

Transform Identifiers for CFStringTransform

Constants that identify transforms used with `CFStringTransform` (page 20).

```
const CFStringRef kCFStringTransformStripCombiningMarks;
const CFStringRef kCFStringTransformToLatin;
const CFStringRef kCFStringTransformFullwidthHalfwidth;
const CFStringRef kCFStringTransformLatinKatakana;
const CFStringRef kCFStringTransformLatinHiragana;
const CFStringRef kCFStringTransformHiraganaKatakana;
const CFStringRef kCFStringTransformMandarinLatin;
const CFStringRef kCFStringTransformLatinHangul;
const CFStringRef kCFStringTransformLatinArabic;
const CFStringRef kCFStringTransformLatinHebrew;
const CFStringRef kCFStringTransformLatinThai;
const CFStringRef kCFStringTransformLatinCyrillic;
const CFStringRef kCFStringTransformLatinGreek;
const CFStringRef kCFStringTransformToXMLHex;
const CFStringRef kCFStringTransformToUnicodeName;
const CFStringRef kCFStringTransformStripDiacritics;
```

Constants

`kCFStringTransformStripCombiningMarks`

The identifier of a transform to strip combining marks (accents or diacritics).

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformToLatin`

The identifier of a transform to transliterate all text possible to Latin script. Ideographs are transliterated as Mandarin Chinese.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformFullwidthHalfwidth`

The identifier of a reversible transform to convert full-width characters to their half-width equivalents.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformLatinKatakana`

The identifier of a reversible transform to transliterate text to Katakana from Latin.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformLatinHiragana`

The identifier of a reversible transform to transliterate text to Hiragana from Latin.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformHiraganaKatakana`

The identifier of a reversible transform to transliterate text to Katakana from Hiragana.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformMandarinLatin`

The identifier of a reversible transform to transliterate text to Latin from ideographs interpreted as Mandarin Chinese.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformLatinHangul`
The identifier of a reversible transform to transliterate text to Hangul from Latin.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformLatinArabic`
The identifier of a reversible transform to transliterate text to Arabic from Latin.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformLatinHebrew`
The identifier of a reversible transform to transliterate text to Hebrew from Latin.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformLatinThai`
The identifier of a reversible transform to transliterate text to Thai from Latin.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformLatinCyrillic`
The identifier of a reversible transform to transliterate text to Cyrillic from Latin.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformLatinGreek`
The identifier of a reversible transform to transliterate text to Greek from Latin.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformToXMLHex`
The identifier of a reversible transform to transliterate characters other than printable ASCII to XML/HTML numeric entities.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformToUnicodeName`
The identifier of a reversible transform to transliterate characters other than printable ASCII (minus braces) to their Unicode character name in braces.

Examples include {AIRPLANE} and {GREEK CAPITAL LETTER PSI}.

Available in Mac OS X v10.4 and later.

Declared in `CFString.h`.

`kCFStringTransformStripDiacritics`
The identifier of a transform to remove diacritic markings.

Available in Mac OS X v10.5 and later.

Declared in `CFString.h`.

Discussion
On Mac OS X v10.4 and later, with `CFStringTransform` (page 20) you can also use any valid ICU transform ID defined in the ICU User Guide for Transforms.

Document Revision History

This table describes the changes to *CFMutableString Reference*.

Date	Notes
2008-10-15	Added explanation of the effect of the locale argument on CFStringLowercase, CFStringUppercase, and CFStringFold functions.
2007-10-31	Included new API introduced in Mac OS X v10.5.
2007-01-08	Updated definition of CFStringTransform and related constants.
2005-12-06	Made minor changes to text to conform to reference consistency guidelines.
2005-08-11	Corrected minor typographical errors.
2005-04-29	Updated to include new API for Mac OS X version 10.4.
2003-01-01	First version of this document.

Document Revision History

Index