CFMutableSet Reference

Core Foundation

2005-12-06

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

Simultaneously published in the United States and Canada.

Contents

CFMutableSet Reference

Derived From:	CFSet : CFType
Framework:	CoreFoundation/CoreFoundation.h
Companion guide	Collections Programming Topics for Core Foundation
Declared in	CFSet.h

Overview

CFMutableSet manages dynamic sets. The basic interface for managing sets is provided by CFSet. CFMutableSet adds functions to modify the contents of a set.

You create a mutable set object using either the `CFSetCreateMutable` (page 6) or `CFSetCreateMutableCopy` (page 7) function.

CFMutableSet provides several functions for adding and removing values from a set. The `CFSetAddValue` (page 5) function adds a value to a set and `CFSetRemoveValue` (page 8) removes a value from a set.

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

Functions

CFSetAddValue

Adds a value to a CFMutableSet object.

```
void CFSetAddValue (
   CFMutableSetRef theSet,
   const void *value
);
```

Parameters

theSet

The set to modify.

value

A CFType object or a pointer value to add to *theSet* (or the value itself, if it fits into the size of a pointer).

value is retained by *theSet* using the retain callback provided when *theSet* was created. If *value* is not of the type expected by the retain callback, the behavior is undefined. If *value* already exists in the collection, this function returns without doing anything.

Availability
Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Related Sample Code
CFLocalServer

Declared In
CFSet.h

CFSetCreateMutable

Creates an empty CFMutableSet object.

```
CFMutableSetRef CFSetCreateMutable (
   CFAllocatorRef allocator,
   CFIndex capacity,
   const CFSetCallBacks *callBacks
);
```

Parameters

allocator

The allocator to use to allocate memory for the new set and its storage for values. Pass NULL or kCFAllocatorDefault to use the current default allocator.

capacity

The maximum number of values that can be contained by new set. The set starts empty and can grow to this number of values (and it can have less). If this parameter is 0, the set's maximum capacity is not limited. The value must not be negative.

callBacks

A pointer to a CFSetCallBacks structure initialized with the callbacks to use to retain, release, describe, and compare values in the set. A copy of the contents of the callbacks structure is made, so that a pointer to a structure on the stack can be passed in or can be reused for multiple collection creations. This parameter may be NULL, which is treated as if a valid structure of version 0 with all fields NULL had been passed in.

If any of the fields are not valid pointers to functions of the correct type, or this parameter is not a valid pointer to a CFSetCallBacks structure, the behavior is undefined. If any value put into the collection is not one understood by one of the callback functions, the behavior when that callback function is used is undefined.

If the collection contains CFType objects only, then pass kCFTypeSetCallBacks as this parameter to use the default callback functions.

Return Value
A new mutable set, or NULL if there was a problem creating the object. Ownership follows the Create Rule.

Availability
Available in CarbonLib v1.0 and later.
Available in Mac OS X v10.0 and later.

Related Sample Code
CFLocalServer

Declared In
`CFSet.h`

CFSetCreateMutableCopy

Creates a new mutable set with the values from another set.

```
CFMutableSetRef CFSetCreateMutableCopy (
   CFAllocatorRef allocator,
   CFIndex capacity,
   CFSetRef theSet
);
```

Parameters

`allocator`

The allocator to use to allocate memory for the new set and its storage for values. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

`capacity`

The maximum number of values that can be contained by the new set. The set starts with the same count as *`theSet`*, and can grow to this number of values (and it can have less). If this parameter is `0`, the set's maximum capacity is not limited. This parameter must be greater than or equal to the count of *`theSet`*, or the behavior is undefined. If this parameter is negative, the behavior is undefined.

`theSet`

The set to copy. The pointer values from *`theSet`* are copied into the new set. The values are also retained by the new set. The count of the new set is the same as the count of *`theSet`*. The new set uses the same callbacks as *`theSet`*.

Return Value
A new mutable set that contains the same values as *`theSet`*. Ownership follows the Create Rule.

Availability
Available in CarbonLib v1.0 and later.
Available in Mac OS X v10.0 and later.

Declared In
`CFSet.h`

CFSetRemoveAllValues

Removes all values from a CFMutableSet object.

```
void CFSetRemoveAllValues (
   CFMutableSetRef theSet
);
```

Parameters

theSet

The set to modify.

Availability
Available in CarbonLib v1.0 and later.
Available in Mac OS X v10.0 and later.

Declared In
CFSet.h

CFSetRemoveValue

Removes a value from a CFMutableSet object.

```
void CFSetRemoveValue (
   CFMutableSetRef theSet,
   const void *value
);
```

Parameters

theSet

The set to modify.

value

The value to remove from *theSet*.

Availability
Available in CarbonLib v1.0 and later.
Available in Mac OS X v10.0 and later.

Related Sample Code
CFLocalServer

Declared In
CFSet.h

CFSetReplaceValue

Replaces a value in a CFMutableSet object.

```
void CFSetReplaceValue (
   CFMutableSetRef theSet,
   const void *value
);
```

Parameters

theSet

The set to modify.

value

The value to replace in *theSet*. If this value does not already exist in *theSet*, the function does nothing. You may pass the value itself instead of a pointer if it is pointer-size or less. The equal callback provided when *theSet* was created is used to compare. If the equal callback was NULL, pointer equality (in C, ==) is used. If *value*, or any other value in *theSet*, is not understood by the equal callback, the behavior is undefined.

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFSet.h

CFSetSetValue

Sets a value in a CFMutableSet object.

```
void CFSetSetValue (
   CFMutableSetRef theSet,
   const void *value
);
```

Parameters

theSet

The set to modify.

value

The value to be set in *theSet*. If this value already exists in *theSet*, it is replaced. You may pass the value itself instead of a pointer to it if the value is pointer-size or less. If *theSet* is fixed-size and setting the value would increase its size beyond its capacity, the behavior is undefined.

Discussion

Depending on the implementation of the equal callback specified when creating *theSet*, the value that is replaced by *value* may not have the same pointer equality.

Availability

Available in CarbonLib v1.0 and later.

Available in Mac OS X v10.0 and later.

Declared In

CFSet.h

Data Types

CFMutableSetRef

A reference to a mutable set object.

```
typedef struct __CFSet *CFMutableSetRef;
```

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

Declared In
`CFSet.h`

Document Revision History

This table describes the changes to *CFMutableSet Reference*.

Date	Notes
2005-12-06	Made minor changes to text to conform to reference consistency guidelines.
2005-08-11	Cosmetic changes to conform to documentation guidelines.
2003-08-01	Enhanced description of all the `kCFType*Callbacks` and added link to Carbon-Cocoa integration document.
2003-01-01	First version of this document.

Document Revision History

Index

C