# CFBinaryHeap Reference

**Core Foundation**

**2006-01-10**

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



Simultaneously published in the United States and Canada.

# Contents

# CFBinaryHeap Reference

| | |
|---|---|
| **Derived From:** | CFType |
| **Framework:** | CoreFoundation/CoreFoundation.h |
| **Companion guide** | Collections Programming Topics for Cocoa |
| **Declared in** | CFBinaryHeap.h |

## Overview

`CFBinaryHeap` implements a container that stores values sorted using a binary search algorithm. All binary heaps are mutable; there is not a separate immutable variety. Binary heaps can be useful as priority queues.

## Functions

### CFBinaryHeapAddValue

Adds a value to a binary heap.

```
void CFBinaryHeapAddValue (
   CFBinaryHeapRef heap,
   const void *value
);
```

**Parameters**

*heap*

The binary heap to use.

*value*

The value to add to the binary heap. The value is retained by the binary heap using the retain callback provided in the `CFBinaryHeapCallBacks` (page 14) structure when the binary heap was created.

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

**Declared In**
`CFBinaryHeap.h`

### CFBinaryHeapApplyFunction

Iteratively applies a function to all the values in a binary heap.

```
void CFBinaryHeapApplyFunction (
   CFBinaryHeapRef heap,
   CFBinaryHeapApplierFunction applier,
   void *context
);
```

**Parameters**

*heap*

The binary heap to use.

*applier*

The callback function to call once for each value in *heap*.

*context*

A program-defined value that is passed to the *applier* callback function, but is otherwise unused by this function.

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

**Declared In**
CFBinaryHeap.h

## CFBinaryHeapContainsValue

Returns whether a given value is in a binary heap.

```
Boolean CFBinaryHeapContainsValue (
   CFBinaryHeapRef heap,
   const void *value
);
```

**Parameters**

*heap*

The binary heap to search.

*value*

The value for which to find matches in the binary heap. The compare callback provided in the CFBinaryHeapCallBacks (page 14) structure when the binary heap was created is used to compare values. If *value*, or any of the values in the binary heap, are not understood by the compare callback, the behavior is undefined.

**Return Value**
true if *value* is a member of *heap*, false otherwise.

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

**Declared In**
CFBinaryHeap.h

## CFBinaryHeapCreate

Creates a new mutable or fixed-mutable binary heap.

```
CFBinaryHeapRef CFBinaryHeapCreate (
   CFAllocatorRef allocator,
   CFIndex capacity,
   const CFBinaryHeapCallBacks *callBacks,
   const CFBinaryHeapCompareContext *compareContext
);
```

**Parameters**

*allocator*

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

*capacity*

The maximum number of values that can be contained by the binary heap. The binary heap starts empty and can grow to this number of values. If this parameter is `0`, the binary heap's maximum capacity is limited only by memory.

*callBacks*

A pointer to a `CFBinaryHeapCallBacks` (page 14) structure initialized with the callbacks that operate on the values placed into the binary heap. If the binary heap will be holding `CFString` objects, pass the `kCFStringBinaryHeapCallBacks` (page 15) constant. This functions makes a copy of the contents of the callbacks structure, so that a pointer to a structure on the stack can be passed in, or can be reused for multiple binary heap creations. This callbacks parameter may not be `NULL`.

*compareContext*

Not used. Pass `NULL`.

**Return Value**

A new `CFBinaryHeap` object. Ownership follows the Create Rule.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`CFBinaryHeap.h`

## CFBinaryHeapCreateCopy

Creates a new mutable or fixed-mutable binary heap with the values from a pre-existing binary heap.

```
CFBinaryHeapRef CFBinaryHeapCreateCopy (
   CFAllocatorRef allocator,
   CFIndex capacity,
   CFBinaryHeapRef heap
);
```

**Parameters**

*allocator*

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

*capacity*

The maximum number of values that can be contained by the binary heap. The binary heap starts with the same number of values as *heap* and can grow to this number of values. If this parameter is 0, the binary heap's maximum capacity is limited only by memory. If nonzero, *capacity* must be large enough to hold all the values in *heap*.

*heap*

The binary heap which is to be copied. The values from the binary heap are copied as pointers into the new binary heap (that is, the values themselves are copied, not that to which the values point, if anything). However, the values are also retained by the new binary heap.

**Return Value**
A new CFBinaryHeap object holding the same values as *heap*. The new binary heap uses the same callbacks as *heap*. Ownership follows the Create Rule.

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

**Declared In**
CFBinaryHeap.h

## CFBinaryHeapGetCount

Returns the number of values currently in a binary heap.

```
CFIndex CFBinaryHeapGetCount (
   CFBinaryHeapRef heap
);
```

**Parameters**

*heap*

The binary heap to use.

**Return Value**
The number of values in *heap*.

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

**Declared In**
CFBinaryHeap.h

## CFBinaryHeapGetCountOfValue

Counts the number of times a given value occurs in a binary heap.

```
CFIndex CFBinaryHeapGetCountOfValue (
   CFBinaryHeapRef heap,
   const void *value
);
```

**Parameters**

*heap*

The binary heap to search.

*value*

The value for which to find matches in the binary heap. The compare callback provided in the `CFBinaryHeapCallBacks` (page 14) structure when the binary heap was created is used to compare. If *value*, or any of the values in the binary heap, are not understood by the compare callback, the behavior is undefined.

**Return Value**
The number of times *value* occurs in *heap*.

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

**Declared In**
`CFBinaryHeap.h`

## CFBinaryHeapGetMinimum

Returns the minimum value in a binary heap.

```
const void * CFBinaryHeapGetMinimum (
   CFBinaryHeapRef heap
);
```

**Parameters**

*heap*

The binary heap to use.

**Return Value**
The minimum value in *heap* as determined by the binary heap's compare callback. If *heap* contains several equal minimum values, any one may be returned. If the value is a Core Foundation object, ownership follows the Get Rule.

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

**Declared In**
`CFBinaryHeap.h`

## CFBinaryHeapGetMinimumIfPresent

Returns the minimum value in a binary heap, if present.

```
Boolean CFBinaryHeapGetMinimumIfPresent (
   CFBinaryHeapRef heap,
   const void **value
);
```

**Parameters**

*heap*

The binary heap to use.

*value*

On return, the minimum value in *heap* as determined by the binary heap's compare callback. If *heap* contains several equal minimum values, any one may be returned. If the value is a Core Foundation object, ownership follows the Get Rule.

**Return Value**
`true` if a minimum value exists in *heap*, `false` otherwise. `false` is returned only if *heap* is empty.

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

**Declared In**
`CFBinaryHeap.h`

## CFBinaryHeapGetTypeID

Returns the type identifier of the `CFBinaryHeap` opaque type.

```
CFTypeID CFBinaryHeapGetTypeID (
   void
);
```

**Return Value**
The type identifier of the `CFBinaryHeap` opaque type.

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

**Declared In**
`CFBinaryHeap.h`

## CFBinaryHeapGetValues

Copies all the values from a binary heap into a sorted C array.

```
void CFBinaryHeapGetValues (
   CFBinaryHeapRef heap,
   const void **values
);
```

**Parameters**

*heap*
The binary heap to use.

*values*
On return, the memory pointed to by this argument holds a C array of all the values in heap, sorted from minimum to maximum values. You must allocate sufficient memory to hold all the values in *heap* before calling this function. If the values are Core Foundation objects, ownership follows the Get Rule.

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

**Declared In**
`CFBinaryHeap.h`

### CFBinaryHeapRemoveAllValues

Removes all values from a binary heap, making it empty.

```
void CFBinaryHeapRemoveAllValues (
   CFBinaryHeapRef heap
);
```

**Parameters**

*heap*

The binary heap to use.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`CFBinaryHeap.h`

### CFBinaryHeapRemoveMinimumValue

Removes the minimum value from a binary heap.

```
void CFBinaryHeapRemoveMinimumValue (
   CFBinaryHeapRef heap
);
```

**Parameters**

*heap*

The binary heap to use.

**Discussion**

If *heap* contains several equal minimum values, only one of them is removed. If *heap* is empty, this function does nothing.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`CFBinaryHeap.h`

## Callbacks

### CFBinaryHeapApplierFunction

Callback function used to apply a function to all members of a binary heap.

```
typedef void (*CFBinaryHeapApplierFunction) (
    const void *val,
    void *context
);
```

If you name your function `MyCallBack`, you would declare it like this:

```
void MyCallBack (
    const void *val,
    void *context
);
```

**Parameters**

*val*

The current value from the binary heap.

*context*

The program-defined context parameter given to the `CFBinaryHeapApplyFunction` (page 5) function.

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

**Declared In**
`CFBinaryHeap.h`

## CFBinaryHeapCompareCallBack

Callback function used to compare two members of a binary heap.

```
typedef CFComparisonResult (*CFBinaryHeapCompareCallBack) (
    const void *ptr1,
    const void *ptr2,
    void *info
);
```

If you name your function `MyCallBack`, you would declare it like this:

```
CFComparisonResult MyCallBack (
    const void *ptr1,
    const void *ptr2,
    void *info
);
```

**Parameters**

*ptr1*

First value to compare.

*ptr2*

Second value to compare.

*info*

Not used. Should always be `NULL`.

**Return Value**
`kCFCompareLessThan` if *ptr1* is less than *ptr2*, `kCFCompareEqualTo` if *ptr1* and *ptr2* are equal, or `kCFCompareGreaterThan` if *ptr1* is greater than *ptr2*.

## CFBinaryHeapCopyDescriptionCallBack

Callback function used to get a description of a value in a binary heap.

```
typedef CFStringRef (*CFBinaryHeapCopyDescriptionCallBack) (
    const void *ptr
);
```

If you name your function `MyCallBack`, you would declare it like this:

```
CFStringRef MyCallBack (
    const void *ptr
);
```

#### Parameters

*ptr*

The value to be described.

### CFBinaryHeapReleaseCallBack

Callback function used to release a value before it is removed from a binary heap.

```
typedef void (*CFBinaryHeapReleaseCallBack) (
    CFAllocatorRef allocator,
    const void *ptr
);
```

If you name your function `MyCallBack`, you would declare it like this:

```
void MyCallBack (
    CFAllocatorRef allocator,
    const void *ptr
);
```

#### Parameters

*allocator*

The binary heap's allocator.

*ptr*

The value to release.

### CFBinaryHeapRetainCallBack

Callback function used to retain a value being added to a binary heap.

```
typedef const void *(*CFBinaryHeapRetainCallBack) (
    CFAllocatorRef allocator,
    const void *ptr
);
```

If you name your function `MyCallBack`, you would declare it like this:

```
const void *MyCallBack (
    CFAllocatorRef allocator,
    const void *ptr
);
```

**Parameters**

*allocator*

The binary heap's allocator.

*ptr*

The value to retain.

**Return Value**

The value to store in the binary heap, which is usually the *ptr* parameter passed to this callback, but may be a different value if a different value should be stored in the binary heap.

# Data Types

## CFBinaryHeapCallBacks

Structure containing the callbacks for values for a `CFBinaryHeap` object.

```
struct CFBinaryHeapCallBacks {
    CFIndex version;
    CFBinaryHeapRetainCallBack retain;
    CFBinaryHeapReleaseCallBack release;
    CFAllocatorCopyDescriptionCallBack copyDescription;
    CFBinaryHeapCompareCallBack compare;
};
typedef struct CFBinaryHeapCallBacks CFBinaryHeapCallBacks;
```

**Fields**

`version`

The version number of the structure type being passed in as a parameter to the `CFBinaryHeap` creation functions. This structure is version `0`.

`retain`

The callback used to add a retain for the binary heap on values as they are put into the binary heap. This callback returns the value to use as the value in the binary heap, which is usually the value parameter passed to this callback, but may be a different value if a different value should be added to the binary heap. If this field is `NULL`, the binary heap does nothing to retain a value being added.

`release`

The callback used to remove a retain previously added for the binary heap from values as they are removed from the binary heap. If this field is `NULL`, the binary heap does nothing to release a value being removed.

`copyDescription`

The callback used to create a descriptive string representation of each value in the binary heap. This is used by the `CFCopyDescription` function. If this field is `NULL`, the binary heap constructs a `CFString` object describing the value based on its pointer value.

`compare`

The callback used to compare values in the binary heap in some operations. This field cannot be `NULL`.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

`CFBinaryHeap.h`

### CFBinaryHeapCompareContext

Not used.

```
struct CFBinaryHeapCompareContext {
    CFIndex version;
    void *info;
    CFAllocatorRetainCallBack retain;
    CFAllocatorReleaseCallBack release;
    CFAllocatorCopyDescriptionCallBack copyDescription;
};
typedef struct CFBinaryHeapCompareContext CFBinaryHeapCompareContext;
```

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

**Declared In**
`CFBinaryHeap.h`

### CFBinaryHeapRef

A reference to a binary heap object.

```
typedef struct __CFBinaryHeap *CFBinaryHeapRef;
```

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

**Declared In**
`CFBinaryHeap.h`

# Constants

## Predefined Callback Structures

`CFBinaryHeap` provides some predefined callbacks for your convenience.

```
const CFBinaryHeapCallBacks kCFStringBinaryHeapCallBacks;
```

**Constants**

`kCFStringBinaryHeapCallBacks`

Predefined `CFBinaryHeapCallBacks` (page 14) structure containing a set of callbacks appropriate for use when the values in a binary heap are all `CFString` objects.

Available in Mac OS X v10.0 and later.

Declared in `CFBinaryHeap.h`.

# Document Revision History

This table describes the changes to *CFBinaryHeap Reference*.

| Date | Notes |
| --- | --- |
| 2006-01-10 | Made minor changes to text. |
| 2005-08-11 | Cosmetic changes to conform to documentation guidelines. |
| 2003-01-01 | First version of this document. |

Document Revision History

# Index