

---

# CFMessagePort Reference

Core Foundation



2007-03-20



Apple Inc.  
© 2002, 2007 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

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

Apple and the Apple logo are trademarks of Apple Inc., registered in the United States and other countries.

iPhone is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR**

**CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## CFMessagePort Reference 5

---

Overview	5
Functions by Task	5
Creating a CFMessagePort Object	5
Configuring a CFMessagePort Object	6
Using a Message Port	6
Examining a Message Port	6
Getting the CFMessagePort Type ID	6
Functions	6
CFMessagePortCreateLocal	6
CFMessagePortCreateRemote	7
CFMessagePortCreateRunLoopSource	8
CFMessagePortGetContext	9
CFMessagePortGetInvalidationCallBack	9
CFMessagePortGetName	10
CFMessagePortGetTypeID	10
CFMessagePortInvalidate	10
CFMessagePortIsRemote	11
CFMessagePortIsValid	11
CFMessagePortSendRequest	12
CFMessagePortSetInvalidationCallBack	13
CFMessagePortSetName	13
Callbacks	14
CFMessagePortCallBack	14
CFMessagePortInvalidationCallBack	15
Data Types	15
CFMessagePortContext	15
CFMessagePortRef	16
Constants	16
CFMessagePortSendRequest Error Codes	16

---

## Document Revision History 19

---

## Index 21

---



# CFMessagePort Reference

---

<b>Derived From:</b>	CFType
<b>Framework:</b>	CoreFoundation/CoreFoundation.h
<b>Declared in</b>	CFMessagePort.h

## Overview

CFMessagePort objects provide a communications channel that can transmit arbitrary data between multiple threads or processes on the local machine.

You create a local message port with [CFMessagePortCreateLocal](#) (page 6) and make it available to other processes by giving it a name, either when you create it or later with [CFMessagePortSetName](#) (page 13). Other processes then connect to it using [CFMessagePortCreateRemote](#) (page 7), specifying the name of the port.

To listen for messages, you need to create a run loop source with [CFMessagePortCreateRunLoopSource](#) (page 8) and add it to a run loop with [CFRunLoopAddSource](#).

**Important:** If you want to tear down the connection, you must invalidate the port (using [CFMessagePortInvalidate](#) (page 10)) before releasing the runloop source and the message port object.

Your message port's callback function will be called when a message arrives. To send data, you store the data in a CFData object and call [CFMessagePortSendRequest](#) (page 12). You can optionally have the function wait for a reply and return the reply in another CFData object.

Message ports only support communication on the local machine. For network communication, you have to use a CFSocket object.

## Functions by Task

### Creating a CFMessagePort Object

[CFMessagePortCreateLocal](#) (page 6)  
Returns a local CFMessagePort object.

[CFMessagePortCreateRemote](#) (page 7)  
Returns a CFMessagePort object connected to a remote port.

## Configuring a CFMessagePort Object

[CFMessagePortCreateRunLoopSource](#) (page 8)

Creates a CFRunLoopSource object for a CFMessagePort object.

[CFMessagePortSetInvalidationCallback](#) (page 13)

Sets the callback function invoked when a CFMessagePort object is invalidated.

[CFMessagePortSetName](#) (page 13)

Sets the name of a local CFMessagePort object.

## Using a Message Port

[CFMessagePortInvalidate](#) (page 10)

Invalidates a CFMessagePort object, stopping it from receiving or sending any more messages.

[CFMessagePortSendRequest](#) (page 12)

Sends a message to a remote CFMessagePort object.

## Examining a Message Port

[CFMessagePortGetContext](#) (page 9)

Returns the context information for a CFMessagePort object.

[CFMessagePortGetInvalidationCallback](#) (page 9)

Returns the invalidation callback function for a CFMessagePort object.

[CFMessagePortGetName](#) (page 10)

Returns the name with which a CFMessagePort object is registered.

[CFMessagePortIsRemote](#) (page 11)

Returns a Boolean value that indicates whether a CFMessagePort object represents a remote port.

[CFMessagePortIsValid](#) (page 11)

Returns a Boolean value that indicates whether a CFMessagePort object is valid and able to send or receive messages.

## Getting the CFMessagePort Type ID

[CFMessagePortGetTypeID](#) (page 10)

Returns the type identifier for the CFMessagePort opaque type.

## Functions

### **CFMessagePortCreateLocal**

Returns a local CFMessagePort object.

```
CFMessagePortRef CFMessagePortCreateLocal (
    CFAllocatorRef allocator,
    CFStringRef name,
    CFMessagePortCallBack callout,
    CFMessagePortContext *context,
    Boolean *shouldFreeInfo
);
```

**Parameters***allocator*

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

*name*

The name with which to register the port. *name* can be `NULL`.

*callout*

The callback function invoked when a message is received on the message port.

*context*

A structure holding contextual information for the message port. The function copies the information out of the structure, so the memory pointed to by *context* does not need to persist beyond the function call.

*shouldFreeInfo*

A flag set by the function to indicate whether the *info* member of *context* should be freed. The flag is set to `true` on failure or if a local port named *name* already exists, `false` otherwise. *shouldFreeInfo* can be `NULL`.

**Return Value**

The new `CFMessagePort` object, or `NULL` on failure. If a local port is already named *name*, the function returns that port instead of creating a new object; the *context* and *callout* parameters are ignored in this case. Ownership follows the Create Rule.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

BackgroundExporter  
BasicInputMethod

**Declared In**

CFMessagePort.h

**CFMessagePortCreateRemote**

Returns a `CFMessagePort` object connected to a remote port.

```
CFMessagePortRef CFMessagePortCreateRemote (
    CFAllocatorRef allocator,
    CFStringRef name
);
```

**Parameters***allocator*

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

*name*

The name of the remote message port to which to connect.

**Return Value**

The new `CFMessagePort` object, or `NULL` on failure. If a message port has already been created for the remote port, the pre-existing object is returned. Ownership follows the Create Rule.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

BackgroundExporter

BasicInputMethod

**Declared In**

CFMessagePort.h

**CFMessagePortCreateRunLoopSource**

Creates a `CFRunLoopSource` object for a `CFMessagePort` object.

```
CFRunLoopSourceRef CFMessagePortCreateRunLoopSource (
    CFAllocatorRef allocator,
    CFMessagePortRef local,
    CFIndex order
);
```

**Parameters***allocator*

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

*ms*

The message port for which to create a run loop source.

*order*

A priority index indicating the order in which run loop sources are processed. *order* is currently ignored by `CFMessagePort` object run loop sources. Pass `0` for this value.

**Return Value**

The new `CFRunLoopSource` object for *ms*. Ownership follows the Create Rule.

**Discussion**

The run loop source is not automatically added to a run loop. To add the source to a run loop, use `CFRunLoopAddSource`.



**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

BackgroundExporter

BasicInputMethod

**Declared In**

CFMessagePort.h

**CFMessagePortGetContext**

Returns the context information for a CFMessagePort object.

```
void CFMessagePortGetContext (
    CFMessagePortRef ms,
    CFMessagePortContext *context
);
```

**Parameters**

*ms*

The message port to examine.

*context*

A pointer to the structure into which the context information for *ms* is to be copied. The information being returned is usually the same information you passed to [CFMessagePortCreateLocal](#) (page 6) when creating *ms*. However, if [CFMessagePortCreateLocal](#) (page 6) returned a cached object instead of creating a new object, *context* is filled with information from the original message port instead of the information you passed to the function.

**Discussion**

The context version number for message ports is currently 0. Before calling this function, you need to initialize the *version* member of *context* to 0.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortGetInvalidationCallback**

Returns the invalidation callback function for a CFMessagePort object.

```
CFMessagePortInvalidationCallback CFMessagePortGetInvalidationCallback (
    CFMessagePortRef ms
);
```

**Parameters**

*ms*

The message port to examine.

**Return Value**

The callback function invoked when *ms* is invalidated. NULL if no callback has been set with [CFMessagePortSetInvalidationCallBack](#) (page 13).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortGetName**

Returns the name with which a CFMessagePort object is registered.

```
CFStringRef CFMessagePortGetName (
    CFMessagePortRef ms
);
```

**Parameters**

*ms*

The message port to examine.

**Return Value**

The registered name of *ms*, NULL if unnamed. Ownership follows the Get Rule.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortGetTypeID**

Returns the type identifier for the CFMessagePort opaque type.

```
CFTypeID CFMessagePortGetTypeID (
    void
);
```

**Return Value**

The type identifier for the CFMessagePort opaque type.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortInvalidate**

Invalidates a CFMessagePort object, stopping it from receiving or sending any more messages.

```
void CFMessagePortInvalidate (
    CFMessagePortRef ms
);
```

**Parameters***ms*

The message port to invalidate.

**Discussion**

Invalidating a message port prevents the port from ever sending or receiving any more messages; the message port is not deallocated, though. If the port has not already been invalidated, the port's invalidation callback function is invoked, if one has been set with [CFMessagePortSetInvalidationCallback](#) (page 13). The [CFMessagePortContext](#) (page 15) info information for *ms* is also released, if a release callback was specified in the port's context structure. Finally, if a run loop source was created for *ms*, the run loop source is also invalidated.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

BackgroundExporter

**Declared In**

CFMessagePort.h

**CFMessagePortIsRemote**

Returns a Boolean value that indicates whether a CFMessagePort object represents a remote port.

```
Boolean CFMessagePortIsRemote (
    CFMessagePortRef ms
);
```

**Parameters***ms*

The message port to examine.

**Return Value**

true if *ms* is a remote port, otherwise false.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortIsValid**

Returns a Boolean value that indicates whether a CFMessagePort object is valid and able to send or receive messages.

```
Boolean CFMessagePortIsValid (
    CFMessagePortRef ms
);
```

**Parameters**

*ms*  
The message port to examine.

**Return Value**

true if *ms* can be used for communication, otherwise false.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortSendRequest**

Sends a message to a remote CFMessagePort object.

```
SInt32 CFMessagePortSendRequest (
    CFMessagePortRef remote,
    SInt32 msgid,
    CFDataRef data,
    CFTimeInterval sendTimeout,
    CFTimeInterval rcvTimeout,
    CFStringRef replyMode,
    CFDataRef *returnData
);
```

**Parameters**

*remote*  
The message port to which *data* should be sent.

*msgid*  
An arbitrary integer value that you can send with the message.

*data*  
The data to send to *remote*.

*sendTimeout*  
The time to wait for *data* to be sent.

*rcvTimeout*  
The time to wait for a reply to be returned.

*replyMode*  
The run loop mode in which the function should wait for a reply. If the message is a oneway (so no response is expected), then *replyMode* should be NULL. If *replyMode* is non-NULL, the function runs the run loop waiting for a reply, in that mode. *replyMode* can be any string name of a run loop mode, but it should be one with input sources installed. You should use the `kCFRunLoopDefaultMode` constant unless you have a specific reason to use a different mode.

*returnData*  
Upon return, contains a CFData object containing the reply data. Ownership follows the Create Rule.

**Return Value**

Error code indicating success or failure. See “[CFMessagePortSendRequest Error Codes](#)” (page 16) for the possible return values.

**Availability**

Available in Mac OS X v10.0 and later.

**Related Sample Code**

BackgroundExporter

BasicInputMethod

**Declared In**

CFMessagePort.h

**CFMessagePortSetInvalidationCallback**

Sets the callback function invoked when a CFMessagePort object is invalidated.

```
void CFMessagePortSetInvalidationCallback (
    CFMessagePortRef ms,
    CFMessagePortInvalidationCallback callout
);
```

**Parameters**

*ms*

The message port to examine.

*callout*

The callback function to invoke when *ms* is invalidated. Pass NULL to remove a callback.

**Discussion**

If *ms* is already invalid, *callout* is invoked immediately.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortSetName**

Sets the name of a local CFMessagePort object.

```
Boolean CFMessagePortSetName (
    CFMessagePortRef ms,
    CFStringRef newName
);
```

**Parameters**

*ms*

The local message port to examine.

*newName*

The new name for *ms*.

**Return Value**

true if the name change succeeds, otherwise false.

**Discussion**

Other threads and processes can connect to a named message port with [CFMessagePortCreateRemote](#) (page 7).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

## Callbacks

**CFMessagePortCallback**

Callback invoked to process a message received on a CFMessagePort object.

```
typedef CFDataRef (*CFMessagePortCallback) (
    CFMessagePortRef local,
    SInt32 msgid,
    CFDataRef data,
    void *info
);
```

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

```
CFDataRef MyCallback (
    CFMessagePortRef local,
    SInt32 msgid,
    CFDataRef data,
    void *info
);
```

**Parameters**

*local*

The local message port that received the message.

*msgid*

An arbitrary integer value assigned to the message by the sender.

*data*

The message data.

*info*

The `info` member of the [CFMessagePortContext](#) (page 15) structure that was used when creating *local*.

**Return Value**

Data to send back to the sender of the message. The system releases the returned CFData object. Return NULL if you want an empty reply returned to the sender.

**Discussion**

If you want the message data to persist beyond this callback, you must explicitly create a copy of *data* rather than merely retain it; the contents of *data* will be deallocated after the callback exits.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortInvalidationCallback**

Callback invoked when a CFMessagePort object is invalidated.

```
typedef void (*CFMessagePortInvalidationCallback) (
    CFMessagePortRef ms,
    void *info
);
```

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

```
void MyCallback (
    CFMessagePortRef ms,
    void *info
);
```

**Parameters**

*ms*

The message port that has been invalidated.

*info*

The `info` member of the [CFMessagePortContext](#) (page 15) structure that was used when creating *ms*, if *ms* is a local port; NULL if *ms* is a remote port.

**Discussion**

Your callback should free any resources allocated for *ms*.

You specify this callback with [CFMessagePortSetInvalidationCallback](#) (page 13).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

## Data Types

**CFMessagePortContext**

A structure that contains program-defined data and callbacks with which you can configure a CFMessagePort object's behavior.

```

struct CFMessagePortContext {
    CFIndex version;
    void *info;
    CFAllocatorRetainCallback retain;
    CFAllocatorReleaseCallback release;
    CFAllocatorCopyDescriptionCallback copyDescription;
};
typedef struct CFMessagePortContext CFMessagePortContext;

```

**Fields**

version

Version number of the structure. Must be 0.

info

An arbitrary pointer to program-defined data, which can be associated with the message port at creation time. This pointer is passed to all the callbacks defined in the context.

retain

A retain callback for your program-defined `info` pointer. Can be `NULL`.

release

A release callback for your program-defined `info` pointer. Can be `NULL`.

copyDescription

A copy description callback for your program-defined `info` pointer. Can be `NULL`.**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

**CFMessagePortRef**

A reference to a message port object.

```
typedef struct __CFMessagePort *CFMessagePortRef;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CFMessagePort.h

## Constants

**CFMessagePortSendRequest Error Codes**Error codes for `CFMessagePortSendRequest`.



```
enum {  
    kCFMessagePortSuccess = 0,  
    kCFMessagePortSendTimeout = -1,  
    kCFMessagePortReceiveTimeout = -2,  
    kCFMessagePortIsInvalid = -3,  
    kCFMessagePortTransportError = -4  
};
```

### Constants

`kCFMessagePortSuccess`

The message was successfully sent and, if a reply was expected, a reply was received.

Available in Mac OS X v10.0 and later.

Declared in `CFMessagePort.h`.

`kCFMessagePortSendTimeout`

The message could not be sent before the send timeout.

Available in Mac OS X v10.0 and later.

Declared in `CFMessagePort.h`.

`kCFMessagePortReceiveTimeout`

No reply was received before the receive timeout.

Available in Mac OS X v10.0 and later.

Declared in `CFMessagePort.h`.

`kCFMessagePortIsInvalid`

The message could not be sent because the message port is invalid.

Available in Mac OS X v10.0 and later.

Declared in `CFMessagePort.h`.

`kCFMessagePortTransportError`

An error occurred trying to send the message.

Available in Mac OS X v10.0 and later.

Declared in `CFMessagePort.h`.



# Document Revision History

---

This table describes the changes to *CFMessagePort Reference*.

Date	Notes
2007-03-20	Added note regarding invalidation of CFMessagePort object.
2006-02-07	Made formatting changes.
2005-11-09	Removed reference to retired document.
2005-08-11	Clarified use of reply mode in CFMessagePortSendRequest.
2003-01-01	First version of this document.

## REVISION HISTORY

### Document Revision History

# Index

---

## C

---

CFMessagePortCallback **callback** [14](#)  
CFMessagePortContext **structure** [15](#)  
CFMessagePortCreateLocal **function** [6](#)  
CFMessagePortCreateRemote **function** [7](#)  
CFMessagePortCreateRunLoopSource **function** [8](#)  
CFMessagePortGetContext **function** [9](#)  
CFMessagePortGetInvalidationCallback **function** [9](#)  
CFMessagePortGetName **function** [10](#)  
CFMessagePortGetTypeID **function** [10](#)  
CFMessagePortInvalidate **function** [10](#)  
CFMessagePortInvalidationCallback **callback** [15](#)  
CFMessagePortIsRemote **function** [11](#)  
CFMessagePortIsValid **function** [11](#)  
CFMessagePortRef **data type** [16](#)  
**CFMessagePortSendRequest Error Codes** [16](#)  
CFMessagePortSendRequest **function** [12](#)  
CFMessagePortSetInvalidationCallback **function** [13](#)  
CFMessagePortSetName **function** [13](#)

## K

---

kCFMessagePortIsValid **constant** [17](#)  
kCFMessagePortReceiveTimeout **constant** [17](#)  
kCFMessagePortSendTimeout **constant** [17](#)  
kCFMessagePortSuccess **constant** [17](#)  
kCFMessagePortTransportError **constant** [17](#)