

---

# Code Fragment Manager Reference

[Carbon](#) > [Runtime Architecture](#)



2005-07-07



Apple Inc.  
© 2003, 2005 Apple Computer, 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, the Apple logo, Carbon, Logic, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

PowerPC and the PowerPC logo are trademarks of International Business Machines Corporation, used under license therefrom.

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

## Code Fragment Manager Reference 5

---

|                                |    |
|--------------------------------|----|
| Overview                       | 5  |
| Functions by Task              | 5  |
| Finding Symbols                | 5  |
| Loading Fragments              | 6  |
| Unloading Fragments            | 6  |
| Converting a Bundle Prelocator | 6  |
| Callbacks                      | 6  |
| CFragInitFunction              | 6  |
| CFragTermProcedure             | 7  |
| Data Types                     | 8  |
| CFragCFBundleLocator           | 8  |
| CFragClosureID                 | 8  |
| CFragConnectionID              | 8  |
| CFragContainerID               | 9  |
| CFragContextID                 | 9  |
| CFragHFSDiskFlatLocator        | 9  |
| CFragHFSLocator                | 9  |
| CFragHFSLocatorPtr             | 9  |
| CFragHFSMemoryLocator          | 10 |
| CFragHFSSegmentedLocator       | 10 |
| CFragInitBlock                 | 10 |
| CFragInitBlockPtr              | 10 |
| CFragResource                  | 11 |
| CFragResourceExtensionHeader   | 12 |
| CFragResourceMember            | 12 |
| CFragResourceSearchExtension   | 13 |
| CFragSystem7DiskFlatLocator    | 13 |
| CFragSystem7InitBlock          | 14 |
| CFragSystem7Locator            | 15 |
| CFragSystem7MemoryLocator      | 16 |
| CFragSystem7SegmentedLocator   | 17 |
| CFragUsage1Union               | 17 |
| CFragUsage2Union               | 18 |
| CFragWhere1Union               | 18 |
| CFragWhere2Union               | 18 |
| ConnectionID                   | 18 |
| DiskFragment                   | 19 |
| FragmentLocator                | 19 |
| FragmentLocatorPtr             | 19 |
| InitBlock                      | 19 |

- InitBlockPtr 20
- LoadFlags 20
- MemFragment 20
- SegmentedFragment 20
- SymClass 21
- Constants 21
  - Architecture Constants 21
  - Code Fragment Kind 22
  - Current Resource Version 23
  - Default Name Length 23
  - File Location 23
  - kCFragGoesToEOF 23
  - kCFragLibUsageMapPrivatelyMask 23
  - kCFragResourceSearchExtensionKind 24
  - kCFragResourceType 24
  - kCompiledCFragArch 24
  - kLoadCFrag 25
  - kPowerPC 25
  - Load Flag, Symbol Class, and Fragment Locator Constants 26
  - Load Options 29
  - Locator Kind 30
  - Symbol Class Constants 31
  - Unresolved Symbol Address 31
  - Usage Constants 32
  - Version Number 33
- Result Codes 33

---

**Appendix A      Deprecated Code Fragment Manager Functions 37**

---

- Deprecated in Mac OS X v10.5 37
  - CloseConnection 37
  - ConvertBundlePreLocator 38
  - CountSymbols 38
  - FindSymbol 39
  - GetDiskFragment 40
  - GetIndSymbol 41
  - GetMemFragment 42
  - GetSharedLibrary 43

---

**Document Revision History 47**

---

---

**Index 49**

---

# Code Fragment Manager Reference

---

|                    |                             |
|--------------------|-----------------------------|
| <b>Framework:</b>  | CoreServices/CoreServices.h |
| <b>Declared in</b> | CodeFragments.h             |

## Overview

This chapter describes the Code Fragment Manager, the part of the Mac OS that loads fragments into memory and prepares them for execution. A fragment can be an application, an import library, a system extension, or any other block of executable code and its associated data.

The Code Fragment Manager is intended to operate transparently to most applications and other software. You need to use the Code Fragment Manager explicitly only if

- you need to load code modules dynamically during the execution of your application or other software
- you want to unload code modules before the termination of your application
- you want to obtain information about the symbols exported by a fragment

For example, if your application supports dynamic loading of tools, filters, or other software modules contained in fragments, you'll need to use the Code Fragment Manager to load and prepare them for execution.

Carbon supports the Code Fragment Manager.

## Functions by Task

### Finding Symbols

[CountSymbols](#) (page 38) **Deprecated in Mac OS X v10.5**

Determines how many symbols are exported from a specified fragment.

[FindSymbol](#) (page 39) **Deprecated in Mac OS X v10.5**

Searches for a specific exported symbol.

[GetIndSymbol](#) (page 41) **Deprecated in Mac OS X v10.5**

Gets information about the exported symbols in a fragment.

## Loading Fragments

[GetDiskFragment](#) (page 40) **Deprecated in Mac OS X v10.5**

Locates and possibly also loads a fragment contained in a file's data fork into your application's context.

[GetMemFragment](#) (page 42) **Deprecated in Mac OS X v10.5**

Prepares a memory-based fragment for subsequent execution.

[GetSharedLibrary](#) (page 43) **Deprecated in Mac OS X v10.5**

Locates and possibly also loads an import library into your application's context.

## Unloading Fragments

[CloseConnection](#) (page 37) **Deprecated in Mac OS X v10.5**

Closes a connection to a fragment.

## Converting a Bundle Prelocator

[ConvertBundlePreLocator](#) (page 38) **Deprecated in Mac OS X v10.5**

Converts a bundle prelocator to a Core Foundation bundle locator.

# Callbacks

## CFragInitFunction

Defines a fragment initialization function that is executed by the Code Fragment Manager when the fragment is first loaded into memory and prepared for execution.

```
typedef OSErr (*CFragInitFunction) (
    const CFragInitBlock * initBlock
);
```

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

```
OSErr MyCFragInitFunction (
    const CFragInitBlock * initBlock
);
```

### Parameters

*initBlock*

A pointer to a fragment initialization block specifying information about the fragment.

### Return Value

A result code. See [“Code Fragment Manager Result Codes”](#) (page 33). Your initialization function should return `noErr` if it executes successfully, and some other result code if it does not. If your initialization function returns any result code other than `noErr`, the entire load fails and the error `fragUserInitProcErr` is returned to the code that requested the root load.

**Discussion**

A fragment's initialization function is executed immediately before the fragment's main function (if it has one) is executed. The initialization function is passed a pointer to an initialization block, which contains information about the fragment, such as its location and connection ID. See [InitBlock](#) (page 19) for a description of the fields of the initialization block.

You can use the initialization function to perform any tasks that need to be performed before any of the code or data in the fragment is accessed. For example, you might want to open the fragment's resource fork (if it has one). You can determine the location of the fragment's container from the `FragmentLocator` field of the fragment initialization block whose address is passed to your initialization function.

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**

`CodeFragments.h`

**CFragTermProcedure**

Defines a pointer to a fragment termination function that is executed by the Code Fragment Manager when the fragment is unloaded from memory.

```
typedef void (*CFragTermProcedure) (
);
```

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

```
void MyCFragTermProcedure ();
```

**Discussion**

A fragment's termination function is executed immediately before the fragment is unloaded from memory. You can use the termination function to perform any necessary clean-up tasks, such as closing open resource files or disposing of any memory allocated by the fragment.

Note that a termination function is not passed any parameters and does not return any result. You are expected to maintain any information about the fragment (such as file reference numbers of any open files) in its static data area.

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**

`CodeFragments.h`

## Data Types

### CFragCFBundleLocator

```
struct CFragCFBundleLocator {
    CFBundleRef fragmentBundle;
    UInt32 offset;
    UInt32 length;
};
typedef struct CFragCFBundleLocator CFragCFBundleLocator;
```

#### Fields

fragmentBundle  
offset  
length

#### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

#### Declared In

CodeFragments.h

### CFragClosureID

```
typedef struct OpaqueCFragClosureID * CFragClosureID;
```

#### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

#### Declared In

CodeFragments.h

### CFragConnectionID

```
typedef struct OpaqueCFragConnectionID * CFragConnectionID;
```

#### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

#### Declared In

CodeFragments.h



## CFragContainerID

```
typedef struct OpaqueCFragContainerID * CFragContainerID;
```

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

CodeFragments.h

## CFragContextID

```
typedef MPPProcessID CFragContextID;
```

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

CodeFragments.h

## CFragHFSDiskFlatLocator

```
typedef CFragSystem7DiskFlatLocator CFragHFSDiskFlatLocator;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

CodeFragments.h

## CFragHFSLocator

```
typedef CFragSystem7Locator CFragHFSLocator;
```

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

CodeFragments.h

## CFragHFSLocatorPtr

```
typedef CFragSystem7LocatorPtr CFragHFSLocatorPtr;
```

### Availability

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**CFragHFSMemoryLocator**

```
typedef CFragSystem7MemoryLocator CFragHFSMemoryLocator;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**CFragHFSSegmentedLocator**

```
typedef CFragSystem7SegmentedLocator CFragHFSSegmentedLocator;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**CFragInitBlock**

```
typedef CFragSystem7InitBlock CFragInitBlock;
```

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**

CodeFragments.h

**CFragInitBlockPtr**

```
typedef CFragSystem7InitBlockPtr CFragInitBlockPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**

CodeFragments.h

**CFragResource**

```

struct CFragResource {
    UInt32 reservedA;
    UInt32 reservedB;
    UInt16 reservedC;
    UInt16 version;
    UInt32 reservedD;
    UInt32 reservedE;
    UInt32 reservedF;
    UInt32 reservedG;
    UInt16 reservedH;
    UInt16 memberCount;
    CFragResourceMember firstMember;
};
typedef struct CFragResource CFragResource;
typedef CFragResource * CFragResourcePtr;
typedef CFragResourcePtr * CFragResourceHandle;

```

**Fields**

reservedA

This field is reserved for future use. Set this field to 0.

reservedB

This field is reserved for future use. Set this field to 0.

reservedC

This field is reserved for future use. Set this field to 0.

version

reservedD

This field is reserved for future use. Set this field to 0.

reservedE

This field is reserved for future use. Set this field to 0.

reservedF

This field is reserved for future use. Set this field to 0.

reservedG

This field is reserved for future use. Set this field to 0.

reservedH

This field is reserved for future use. Set this field to 0.

memberCount

firstMember

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**

CodeFragments.h

## CFragResourceExtensionHeader

```

struct CFragResourceExtensionHeader {
    UInt16 extensionKind;
    UInt16 extensionSize;
};
typedef struct CFragResourceExtensionHeader CFragResourceExtensionHeader;
typedef CFragResourceExtensionHeader * CFragResourceExtensionHeaderPtr;

```

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

CodeFragments.h

## CFragResourceMember

```

struct CFragResourceMember {
    CFragArchitecture architecture;
    UInt16 reservedA;
    UInt8 reservedB;
    UInt8 updateLevel;
    CFragVersionNumber currentVersion;
    CFragVersionNumber oldDefVersion;
    CFragUsage1Union uUsage1;
    CFragUsage2Union uUsage2;
    CFragUsage usage;
    CFragLocatorKind where;
    UInt32 offset;
    UInt32 length;
    CFragWhere1Union uWhere1;
    CFragWhere2Union uWhere2;
    UInt16 extensionCount;
    UInt16 memberSize;
    unsigned char name[16];
};
typedef struct CFragResourceMember CFragResourceMember;
typedef CFragResourceMember * CFragResourceMemberPtr;

```

### Fields

architecture

reservedA

This field is reserved. Set to 0.

reservedB

This field is reserved. Set to 0.

updateLevel  
currentVersion  
oldDefVersion  
uUsage1  
uUsage2  
usage  
where  
offset  
length  
uWhere1  
uWhere2  
extensionCount

Specifies the number of extensions beyond the name.

memberSize

Specifies the size in bytes, including all extensions.

name

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**

CodeFragments.h

## CFragResourceSearchExtension

```
struct CFragResourceSearchExtension {
    CFragResourceExtensionHeader header;
    OSType libKind;
    unsigned char qualifiers[1];
};
typedef struct CFragResourceSearchExtension CFragResourceSearchExtension;
typedef CFragResourceSearchExtension * CFragResourceSearchExtensionPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**

CodeFragments.h

## CFragSystem7DiskFlatLocator

Defines a disk location structure.

```

struct CFragSystem7DiskFlatLocator {
    FSSpec * fileSpec;
    UInt32 offset;
    UInt32 length;
};
typedef struct CFragSystem7DiskFlatLocator CFragSystem7DiskFlatLocator;
typedef CFragSystem7DiskFlatLocator DiskFragment;

```

**Fields**

fileSpec

A pointer to a file specification structure (a data structure of type `FSSpec`) for the data fork of a file. This pointer is valid only while the initialization function is executing. If you need to access the information in the file specification structure at any later time, you must make a copy of that structure.

offset

The offset, in bytes, from the beginning of the file's data fork to the beginning of the fragment.

length

The length, in bytes, of the fragment. If this field contains the value 0, the fragment extends to the end-of-file.

**Discussion**

For fragments located in the data fork of a file on disk, the `onDisk` field of a fragment location structure contains a disk location structure, which specifies the location of the fragment.

The fields of a fragment initialization block are aligned in memory in accordance with 680x0 alignment conventions.

**Availability**

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

**Declared In**

CodeFragments.h

**CFragSystem7InitBlock**

```

struct CFragSystem7InitBlock {
    CFragContextID contextID;
    CFragClosureID closureID;
    CFragConnectionID connectionID;
    CFragSystem7Locator fragLocator;
    StringPtr libName;
    UInt32 reservedA;
};
typedef struct CFragSystem7InitBlock CFragSystem7InitBlock;
typedef CFragSystem7InitBlock * CFragSystem7InitBlock;
typedef CFragSystem7InitBlock CFragInitBlock;

```

**Fields**

contextID

A context ID.

closureID

A closure ID.

connectionID

A connection ID.

fragLocator

A fragment location structure, [CFragSystem7Locator](#) (page 15) that specifies the location of the fragment.

libName

A pointer to the name of the fragment being initialized. The name is a Pascal string (a length byte followed by the name itself).

reservedA

Reserved for use by Apple Computer.

### Discussion

The Code Fragment Manager passes to your fragment's initialization function a pointer to a fragment initialization block, which contains information about the fragment. A fragment initialization block is defined by the `InitBlock` data type.

The fields of a fragment initialization block are aligned in memory in accordance with 680x0 alignment conventions.

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

`CodeFragments.h`

## CFragSystem7Locator

Defines a fragment location structure.

```
struct CFragSystem7Locator {
    SInt32 where
    union {
        CFragSystem7DiskFlatLocator onDisk;
        CFragSystem7MemoryLocator inMem;
        CFragSystem7SegmentedLocator inSegs;
        CFragCFBundleLocator inBundle;
    } u;
};
typedef struct CFragSystem7Locator CFragSystem7Locator;
typedef CFragSystem7Locator * CFragSystem7LocatorPtr;
typedef CFragSystem7Locator FragmentLocator;
```

### Fields

where

A selector that determines which member of the following union is relevant. This field can contain one of the constants described in [“Load Flag, Symbol Class, and Fragment Locator Constants”](#) (page 26).

u

If the `where` field has the value `kOnDiskFlat`, a disk location structure.

### Discussion

The `fragLocator` field of an initialization block contains a fragment location structure that provides information about the location of a fragment.

The fields of a fragment initialization block are aligned in memory in accordance with 680x0 alignment conventions.

#### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

#### Declared In

CodeFragments.h

### CFragSystem7MemoryLocator

Defines a memory location structure.

```
struct CFragSystem7MemoryLocator {
    LogicalAddress address;
    UInt32 length;
    Boolean inPlace;
    UInt8 reservedA;
    UInt16 reservedB;
};
typedef struct CFragSystem7MemoryLocator CFragSystem7MemoryLocator;
typedef CFragSystem7MemoryLocator MemFragment;
```

#### Fields

address

A pointer to the beginning of the fragment in memory.

length

The length, in bytes, of the fragment.

inPlace

A Boolean value that specifies whether the container's data section is instantiated in place (`true`) or elsewhere (`false`).

reservedA

This field is reserved for future use. Set to 0.

reservedB

This field is reserved for future use. Set to 0.

#### Discussion

For fragments located in memory, the `inMem` field of a fragment location structure contains a memory location structure, which specifies the location of the fragment in memory.

The fields of a fragment initialization block are aligned in memory in accordance with 680x0 alignment conventions.

#### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

#### Declared In

CodeFragments.h



## CFragSystem7SegmentedLocator

Defines a segment location structure.

```

struct CFragSystem7SegmentedLocator {
    FSSpec * fileSpec;
    OSType rsrcType;
    SInt16 rsrcID;
    UInt16 reservedA;
};
typedef struct CFragSystem7SegmentedLocator CFragSystem7SegmentedLocator;
typedef CFragSystem7SegmentedLocator SegmentedFragment;

```

### Fields

fileSpec

A pointer to a file specification structure (a data structure of type `FSSpec`) for the resource fork of a file. This pointer is valid only while the initialization function is executing. If you need to access the information in the file specification structure at any later time, you must make a copy of that structure.

rsrcType

The resource type of the resource containing the fragment.

rsrcID

The resource ID of the resource containing the fragment.

reservedA

This field is reserved for future use.

### Discussion

For fragments located in the resource fork of a file on disk, the `inSegs` field of a fragment location structure contains a segment location structure, which specifies the location of the fragment.

The fields of a fragment initialization block are aligned in memory in accordance with 680x0 alignment conventions.

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

CodeFragments.h

## CFragUsage1Union

```

union CFragUsage1Union {
    UInt32 appStackSize;
};
typedef union CFragUsage1Union CFragUsage1Union;

```

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

CodeFragments.h

## CFragUsage2Union

```
union CFragUsage2Union {
    SInt16 appSubdirID;
    UInt16 libFlags;
};
typedef union CFragUsage2Union CFragUsage2Union;
```

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

CodeFragments.h

## CFragWhere1Union

```
union CFragWhere1Union {
    UInt32 spaceID;
};
typedef union CFragWhere1Union CFragWhere1Union;
```

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

CodeFragments.h

## CFragWhere2Union

```
union CFragWhere2Union {
    UInt16 reserved;
};
typedef union CFragWhere2Union CFragWhere2Union;
```

### Availability

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

### Declared In

CodeFragments.h

## ConnectionID

```
typedef CFragConnectionID ConnectionID;
```

### Availability

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**DiskFragment**

A CFragSystem7DiskFlatLocator structure.

```
typedef CFragSystem7DiskFlatLocator DiskFragment;
```

**Discussion**

See [CFragSystem7DiskFlatLocator](#) (page 13).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**FragmentLocator**

A CFragSystem7Locator structure.

```
typedef CFragSystem7Locator FragmentLocator;
```

**Discussion**

See [CFragSystem7Locator](#) (page 15).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**FragmentLocatorPtr**

```
typedef CFragSystem7LocatorPtr FragmentLocatorPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**InitBlock**

```
typedef CFragInitBlock InitBlock;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**InitBlockPtr**

```
typedef CFragInitBlockPtr InitBlockPtr;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**LoadFlags**

```
typedef CFragLoadOptions LoadFlags;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**MemFragment**

A `CFragSystem7MemoryLocator` structure.

```
typedef CFragSystem7MemoryLocator MemFragment;
```

**Discussion**

See [CFragSystem7MemoryLocator](#) (page 16).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**SegmentedFragment**

A `CFragSystem7SegmentedLocator` structure.

```
typedef CFragSystem7SegmentedLocator SegmentedFragment;
```

**Discussion**

See [CFragSystem7SegmentedLocator](#) (page 17).

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

**SymClass**

```
typedef CFragSymbolClass SymClass;
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

CodeFragments.h

## Constants

### Architecture Constants

```
typedef OSType CFragArchitecture;  
enum {  
    kPowerPCCFragArch = 'pwpc',  
    kMotorola68KCFragArch = 'm68k',  
    kAnyCFragArch = 0x3F3F3F3F  
};
```

**Constants**

kPowerPCCFragArch

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in CodeFragments.h.

kMotorola68KCFragArch

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in CodeFragments.h.

kAnyCFragArch

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in CodeFragments.h.

## Code Fragment Kind

```
enum {  
    kIsCompleteCFrag = 0,  
    kFirstCFragUpdate = 1  
};
```

### Constants

`kIsCompleteCFrag`

Indicates a base fragment rather than an update.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kFirstCFragUpdate`

Indicates the first update, others are numbered starting with 2.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

## Current Resource Version

```
enum {  
    kCurrCFragResourceVersion = 1  
};
```

## Default Name Length

```
enum {  
    kDefaultCFragNameLen = 16  
};
```

## File Location

```
#define IsFileLocation CFragHasFileLocation;
```

## kCFragGoesToEOF

```
enum {  
    kCFragGoesToEOF = 0  
};
```

## kCFragLibUsageMapPrivatelyMask

```
enum {  
    kCFragLibUsageMapPrivatelyMask = 0x0001  
};
```

### Constants

**kCFragLibUsageMapPrivatelyMask**  
Available in Mac OS X v10.0 and later.  
Not available to 64-bit applications.  
Declared in `CodeFragments.h`.

## kCFragResourceSearchExtensionKind

```
enum {
    kCFragResourceSearchExtensionKind = 0x30EE
};
```

## kCFragResourceType

```
enum {
    kCFragResourceType = 'cfrg',
    kCFragResourceID = 0,
    kCFragLibraryFileType = 'shlb',
    kCFragAllFileTypes = 0xFFFFFFFF
};
```

### Constants

**kCFragResourceType**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

**kCFragResourceID**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

**kCFragLibraryFileType**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

**kCFragAllFileTypes**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

## kCompiledCFragArch

```
enum {
    kCompiledCFragArch = 'kPowerPCCFragArch'
};
```

### Constants

**kCompiledCFragArch**  
 The value for this constant is 'kPowerPCCFragArch' if you have defined `TARGET_CPU_PPC`. If you define `TARGET_CPU_X86`, then the value of this constant is 'none'.  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.



## kLoadCFrag

```
enum {  
    kLoadCFrag = kReferenceCFrag  
};
```

### Constants

**kLoadCFrag**  
Available in Mac OS X v10.0 and later.  
Not available to 64-bit applications.  
Declared in `CodeFragments.h`.

## kPowerPC

```
enum {  
    kPowerPC = kPowerPCCFragArch,  
    kMotorola68K = kMotorola68KCFragArch  
};
```

### Constants

**kPowerPC**  
Available in Mac OS X v10.0 and later.  
Declared in `CodeFragments.h`.

**kMotorola68K**  
Available in Mac OS X v10.0 and later.  
Declared in `CodeFragments.h`.

## Load Flag, Symbol Class, and Fragment Locator Constants

```
enum {
    kPowerPCArch = kPowerPCCFragArch,
    kMotorola68KArch = kMotorola68KCFragArch,
    kAnyArchType = kAnyCFragArch,
    kNoLibName = 0,
    kNoConnectionID = 0,
    kLoadLib = kLoadCFrag,
    kFindLib = kFindCFrag,
    kNewCFragCopy = kPrivateCFragCopy,
    kLoadNewCopy = kPrivateCFragCopy,
    kUseInPlace = 0x80,
    kCodeSym = kCodeCFragSymbol,
    kDataSym = kDataCFragSymbol,
    kTVectSym = kTVectorCFragSymbol,
    kTOCSym = kTOCCFragSymbol,
    kGlueSym = kGlueCFragSymbol,
    kInMem = kMemoryCFragLocator,
    kOnDiskFlat = kDataForkCFragLocator,
    kOnDiskSegmented = kResourceCFragLocator,
    kIsLib = kImportLibraryCFrag,
    kIsApp = kApplicationCFrag,
    kIsDropIn = kDropInAdditionCFrag,
    kFullLib = kIsCompleteCFrag,
    kUpdateLib = kFirstCFragUpdate,
    kWholeFork = kCFragGoesToEOF,
    kCFMRsrcType = kCFragResourceType,
    kCFMRsrcID = kCFragResourceID,
    kSHLBFileType = kCFragLibraryFileType,
    kUnresolvedSymbolAddress = kUnresolvedCFragSymbolAddress
};
```

### Constants

kPowerPCArch

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

**Declared in** CodeFragments.h.

kMotorola68KArch

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

**Declared in** CodeFragments.h.

kAnyArchType

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

**Declared in** CodeFragments.h.

kNoLibName

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

**Declared in** CodeFragments.h.

kNoConnectionID

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

**Declared in** CodeFragments.h.

`kLoadLib`

Specifies that the Code Fragment Manager search for the specified fragment.

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kFindLib`

Specifies that the Code Fragment Manager search for the specified fragment and, if it finds it, load it into memory. If the fragment has already been loaded, it is not loaded again. The Code Fragment Manager uses the data-instantiation method specified in the fragment's container (which is either global or per-connection instantiation).

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kNewCFragCopy`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kLoadNewCopy`

Specifies that the Code Fragment Manager load the specified fragment, creating a new copy of any writable data maintained by the fragment. You specify `kLoadNewCopy` to obtain one instance per load of the fragment's data and to override the data-instantiation method specified in the container itself. This is most useful for application extensions (for example, drop-in tools).

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kUseInPlace`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kCodeSym`

Specifies a code symbol.

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kDataSym`

Specifies a data symbol.

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kTVectSym`

Specifies a transition vector symbol.

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kTOCSym`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kGlueSym`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kInMem`

Specifies that the container is in memory. If used in the `where` parameter of a `FragmentLocator` structure, the relevant member of the union is a [CFragSystem7SegmentedLocator](#) (page 17) structure.

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kOnDiskFlat`

Specifies that the container is in a data fork. If used in the `where` parameter of a `FragmentLocator` structure, the relevant member of the union is a [CFragSystem7SegmentedLocator](#) (page 17) structure.

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kOnDiskSegmented`

Specifies that the container is in a resource. If used in the `where` parameter of a `FragmentLocator` structure, the relevant member of the union is a [CFragSystem7SegmentedLocator](#) (page 17) structure.

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kIsLib`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kIsApp`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kIsDropIn`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kFullLib`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kUpdateLib`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kWholeFork`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kCFMRsrcType`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kCFMRsrcID`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kSHLBFilEType`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

`kUnresolvedSymbolAddress`

Available in Mac OS X v10.0 and later.

Declared in `CodeFragments.h`.

### Discussion

The load flag constants (`kLoadLib`, `kFindLib`, and `kLoadNewCopy`) are used in the `loadFlags` parameter of the [GetDiskFragment](#) (page 40), [GetMemFragment](#) (page 42), and [GetSharedLibrary](#) (page 43) functions to specify the action taken by those functions.

The symbol class constants (`kCodeSym`, `kDataSym`, and `kTVectSym`) are returned in the `symClass` parameter of the [FindSymbol](#) (page 39) function to specify the class of the specified symbol.

The fragment locator constants (`kInMem`, `kOnDiskFlat`, and `kOnDiskSegmented`) are used in the `where` field of the [FragmentLocator](#) (page 19) structure to indicate which member of the union `u` is relevant.

## Load Options

```
typedef OptionBits CFragLoadOptions;
enum {
    kReferenceCFrag = 0x0001,
    kFindCFrag = 0x0002,
    kPrivateCFragCopy = 0x0005
};
```

### Constants

`kReferenceCFrag`

Try to use existing copy, increment reference counts.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kFindCFrag`

Try find an existing copy, do not increment reference counts.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kPrivateCFragCopy`

Prepare a new private copy.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

## Locator Kind

```
typedef UInt8 CFragLocatorKind;
enum {
    kMemoryCFragLocator = 0,
    kDataForkCFragLocator = 1,
    kResourceCFragLocator = 2,
    kNamedFragmentCFragLocator = 4,
    kCFBundleCFragLocator = 5,
    kCFBundlePreCFragLocator = 6
};
```

### Constants

`kMemoryCFragLocator`

Indicates the container is in memory.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kDataForkCFragLocator`

Indicates the container is in a file's data fork.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kResourceCFragLocator`

Indicates the container is in a file's resource fork.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kNamedFragmentCFragLocator`

This constant is reserved for future use.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kCFBundleCFragLocator`

Indicates the container is in the executable of a `CFBundle`.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kCFBundlePreCFragLocator`

Indicates it was passed to the initialization routines in lieu of `kCFBundleCFragLocator`

Available in Mac OS X v10.1 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

## Symbol Class Constants

```
typedef UInt8 CFragSymbolClass;
enum {
    kCodeCFragSymbol = 0,
    kDataCFragSymbol = 1,
    kTVectorCFragSymbol = 2,
    kTOCCFragSymbol = 3,
    kGlueCFragSymbol = 4
};
```

### Constants

**kCodeCFragSymbol**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

**kDataCFragSymbol**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

**kTVectorCFragSymbol**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

**kTOCCFragSymbol**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

**kGlueCFragSymbol**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

## Unresolved Symbol Address

```
enum {
    kUnresolvedCFragSymbolAddress = 0
};
```

### Constants

**kUnresolvedCFragSymbolAddress**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

## Usage Constants

```
typedef UInt8 CFragUsage;
enum {
    kImportLibraryCFrag = 0,
    kApplicationCFrag = 1,
    kDropInAdditionCFrag = 2,
    kStubLibraryCFrag = 3,
    kWeakStubLibraryCFrag = 4
};
```

### Constants

`kImportLibraryCFrag`

Indicates a standard CFM import library.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kApplicationCFrag`

Indicates a MacOS application.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kDropInAdditionCFrag`

Indicates an application or library private extension/plug-in.

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kStubLibraryCFrag`

Indicates an import library used for linking only

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.

`kWeakStubLibraryCFrag`

Indicates an import library used for linking only and will be automatically weak linked

Available in Mac OS X v10.0 and later.

Not available to 64-bit applications.

Declared in `CodeFragments.h`.



## Version Number

```
typedef UInt32 CFragVersionNumber;
enum {
    kNullCFragVersion = 0,
    kWildcardCFragVersion = 0xFFFFFFFF
};
```

### Constants

**kNullCFragVersion**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

**kWildcardCFragVersion**  
 Available in Mac OS X v10.0 and later.  
 Not available to 64-bit applications.  
 Declared in `CodeFragments.h`.

## Result Codes

The most common result codes returned by Code Fragment Manager are listed in the table below. The Code Fragment Manager may also return `paramErr` (-50).

| Result Code                          | Value | Description   |
|--------------------------------------|-------|---|
| <code>cfragContextIDErr</code>       | -2800 | The context ID was not valid.<br>Available in Mac OS X v10.0 and later.               |
| <code>cfragFirstErrCode</code>       | -2800 | The first value in the range of CFM errors.<br>Available in Mac OS X v10.0 and later. |
| <code>cfragConnectionIDErr</code>    | -2801 | The connection ID was not valid.<br>Available in Mac OS X v10.0 and later.            |
| <code>cfragNoSymbolErr</code>        | -2802 | The specified symbol was not found.<br>Available in Mac OS X v10.0 and later.         |
| <code>cfragNoSectionErr</code>       | -2803 | The specified section was not found.<br>Available in Mac OS X v10.0 and later.        |
| <code>cfragNoLibraryErr</code>       | -2804 | The named library was not found.<br>Available in Mac OS X v10.0 and later.            |
| <code>cfragDupRegistrationErr</code> | -2805 | The registration name was already in use.<br>Available in Mac OS X v10.0 and later.   |

| Result Code             | Value | Description  |
|-------------------------|-------|--|
| cfragFragmentFormatErr  | -2806 | A fragment's container format is unknown.<br>Available in Mac OS X v10.0 and later.                      |
| cfragUnresolvedErr      | -2807 | A fragment had "hard" unresolved imports.<br>Available in Mac OS X v10.0 and later.                      |
| cfragNoPositionErr      | -2808 | The registration insertion point was not found.<br>Available in Mac OS X v10.0 and later.                |
| cfragNoPrivateMemErr    | -2809 | Out of memory for internal bookkeeping.<br>Available in Mac OS X v10.0 and later.                        |
| cfragNoClientMemErr     | -2810 | Out of memory for fragment mapping or section instances.<br>Available in Mac OS X v10.0 and later.       |
| cfragNoIDsErr           | -2811 | No more CFM IDs for contexts, connections, etc.<br>Available in Mac OS X v10.0 and later.                |
| cfragInitOrderErr       | -2812 | Available in Mac OS X v10.0 and later.   |
| cfragImportTooOldErr    | -2813 | An import library was too old for a client.<br>Available in Mac OS X v10.0 and later.                    |
| cfragImportTooNewErr    | -2814 | An import library was too new for a client.<br>Available in Mac OS X v10.0 and later.                    |
| cfragInitLoopErr        | -2815 | Circularity in required initialization order.<br>Available in Mac OS X v10.0 and later.                  |
| cfragInitAtBootErr      | -2816 | A boot library has an initialization function. (System 7 only)<br>Available in Mac OS X v10.0 and later. |
| cfragCFMStartupErr      | -2818 | Internal error during CFM initialization.<br>Available in Mac OS X v10.0 and later.                      |
| cfragCFMInternalErr     | -2819 | An internal inconsistency has been detected.<br>Available in Mac OS X v10.0 and later.                   |
| cfragFragmentCorruptErr | -2820 | A fragment's container was corrupt (known format).<br>Available in Mac OS X v10.0 and later.             |
| cfragInitFunctionErr    | -2821 | A fragment's initialization routine returned an error.<br>Available in Mac OS X v10.0 and later.         |
| cfragNoApplicationErr   | -2822 | No application member found in the cfrg resource.<br>Available in Mac OS X v10.0 and later.              |

| Result Code            | Value | Description   |
|------------------------|-------|---|
| cfragArchitectureErr   | -2823 | A fragment has an unacceptable architecture.<br>Available in Mac OS X v10.0 and later.        |
| cfragFragmentUsageErr  | -2824 | A semantic error in usage of the fragment.<br>Available in Mac OS X v10.0 and later.          |
| cfragFileSizeErr       | -2825 | A file was too large to be mapped.<br>Available in Mac OS X v10.0 and later.                  |
| cfragNotClosureErr     | -2826 | The closure ID was actually a connection ID.<br>Available in Mac OS X v10.0 and later.        |
| cfragNoRegistrationErr | -2827 | The registration name was not found.<br>Available in Mac OS X v10.0 and later.                |
| cfragContainerIDErr    | -2828 | The fragment container ID was not valid.<br>Available in Mac OS X v10.0 and later.            |
| cfragClosureIDErr      | -2829 | The closure ID was not valid.<br>Available in Mac OS X v10.0 and later.                       |
| cfragAbortClosureErr   | -2830 | Used by notification handlers to abort a closure.<br>Available in Mac OS X v10.0 and later.   |
| cfragOutputLengthErr   | -2831 | An output parameter is too small to hold the value.<br>Available in Mac OS X v10.0 and later. |
| cfragMapFileErr        | -2851 | A file could not be mapped.<br>Available in Mac OS X v10.4 and later.                         |
| cfragExecFileRefErr    | -2854 | Bundle does not have valid executable file.<br>Available in Mac OS X v10.4 and later.         |
| cfragStdFolderErr      | -2855 | Could not find standard CFM folder.<br>Available in Mac OS X v10.4 and later.                 |
| cfragRsrcForkErr       | -2856 | Resource fork could not be opened.<br>Available in Mac OS X v10.4 and later.                  |
| cfragCFragRsrcErr      | -2857 | 'cfrg' resource could not be loaded.<br>Available in Mac OS X v10.4 and later.                |
| cfragFirstReservedCode | -2897 | Reserved value for internal warnings.<br>Available in Mac OS X v10.0 and later.               |

| Result Code         | Value | Description  |
|---------------------|-------|--|
| cfragReservedCode_3 | -2897 | Reserved value for internal warnings.<br>Available in Mac OS X v10.0 and later.      |
| cfragReservedCode_2 | -2898 | Reserved value for internal warnings.<br>Available in Mac OS X v10.0 and later.      |
| cfragLastErrCode    | -2899 | The last value in the range of CFM errors.<br>Available in Mac OS X v10.0 and later. |
| cfragReservedCode_1 | -2899 | Available in Mac OS X v10.0 and later.   |

# Deprecated Code Fragment Manager Functions

---

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

## Deprecated in Mac OS X v10.5

### CloseConnection

Closes a connection to a fragment. (Deprecated in Mac OS X v10.5.)

```
OSErr CloseConnection (
    CFragConnectionID *connID
);
```

#### Parameters

*connID*

A pointer to a connection ID.

#### Return Value

A result code. See “Code Fragment Manager Result Codes” (page 33).

#### Discussion

The `CloseConnection` function closes the connection to a fragment indicated by the `connID` parameter. `CloseConnection` decrements the count of existing connections to the specified fragment and, if the resulting count is 0, calls the fragment’s termination function and releases the memory occupied by the code and data sections of the fragment. If the resulting count is not 0, any per-connection data is released but the code section remains in memory.

When a fragment is unloaded as a result of its final connection having been closed, all libraries that depend on that fragment are also released, provided that their usage counts are also 0.

The Code Fragment Manager automatically closes any connections that remain open at the time `ExitToShell` is called for your application, so you need to call `CloseConnection` only for fragments you wish to unload before your application terminates.

#### Special Considerations

You can close a connection only to the root of a loading sequence (that is, the fragment whose loading triggered the entire load chain).

#### Availability

Available in CarbonLib 1.0 and later when Code Fragment Manager 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

## Deprecated Code Fragment Manager Functions

**Declared In**

CodeFragments.h

**ConvertBundlePreLocator**

Converts a bundle prelocator to a Core Foundation bundle locator. (Deprecated in Mac OS X v10.5.)

```
OSErr ConvertBundlePreLocator (
    CFragSystem7LocatorPtr initBlockLocator
);
```

**Parameters***initBlockLocator*

A pointer to a fragment locator structure. On input, the structure contains a System 7 locator. On output, the structure contains a CFragCFBundleLocator.

**Return Value**

A result code. See “Code Fragment Manager Result Codes” (page 33).

**Discussion**

This function can be used by initialization routines.

**Availability**

Available in Mac OS X 10.1 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

CodeFragments.h

**CountSymbols**

Determines how many symbols are exported from a specified fragment. (Deprecated in Mac OS X v10.5.)

```
OSErr CountSymbols (
    CFragConnectionID connID,
    long *symCount
);
```

**Parameters***connID*

A connection ID.

*symCount*

On return, a pointer to the number of exported symbols in the fragment whose connection ID is *connID*. You can use the value returned in *symCount* to index through all the exported symbols in a particular fragment (using the `GetIndSymbol` function).

**Return Value**

A result code. See “Code Fragment Manager Result Codes” (page 33).

**Availability**

Available in CarbonLib 1.0 and later when Code Fragment Manager 1.0 or later is present.

Available in Mac OS X 10.0 and later.

## Deprecated Code Fragment Manager Functions

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

**Declared In**

CodeFragments.h

**FindSymbol**

Searches for a specific exported symbol. (Deprecated in Mac OS X v10.5.)

```
OSErr FindSymbol (
    CFragConnectionID connID,
    ConstStr255Param symName,
    Ptr *symAddr,
    CFragSymbolClass *symClass
);
```

**Parameters**

*connID*

A connection ID.

*symName*

A symbol name.

*symAddr*

On return, a pointer to the address of the symbol whose name is *symName*.

*symClass*

On return, a pointer to the class of the symbol whose name is *symName*. The currently recognized symbol classes are defined by the “[Load Flag, Symbol Class, and Fragment Locator Constants](#)” (page 26).

**Return Value**

A result code. See “[Code Fragment Manager Result Codes](#)” (page 33).

**Discussion**

The `FindSymbol` function searches the code fragment identified by the `connID` parameter for the symbol whose name is specified by the `symName` parameter. If that symbol is found, `FindSymbol` returns the address of the symbol in the `symAddr` parameter and the class of the symbol in the `symClass` parameter.

Because a fragment’s code is normally exported through transition vectors to that code, the value `kCodeSymbol` is not returned in the PowerPC environment. You can use the other two constants to distinguish exports that represent code (of class `kTVectSymbol`) from those that represent general data (of class `kDataSymbol`).

**Availability**

Available in CarbonLib 1.0 and later when Code Fragment Manager 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

CodeFragments.h

## GetDiskFragment

Locates and possibly also loads a fragment contained in a file's data fork into your application's context. (Deprecated in Mac OS X v10.5.)

```
OSErr GetDiskFragment (
    const FSSpec *fileSpec,
    UInt32 offset,
    UInt32 length,
    ConstStr63Param fragName,
    CFragLoadOptions options,
    CFragConnectionID *connID,
    Ptr *mainAddr,
    Str255 errMessage
);
```

### Parameters

*fileSpec*

A pointer to a file system specification that identifies the disk-based fragment to load.

*offset*

The number of bytes from the beginning of the file's data fork at which the beginning of the fragment is located.

*length*

The length (in bytes) of the fragment. Specify the constant `kWholeFork` for this parameter if the fragment extends to the end-of-file of the data fork. Specify a nonzero value for the exact length of the fragment.

*fragName*

An optional name of the fragment. (This information is used primarily to allow you to identify the fragment during debugging.)

*loadFlags*

A flag that specifies the operation to perform on the fragment. The Code Fragment Manager recognizes the constants described in [“Load Flag, Symbol Class, and Fragment Locator Constants”](#) (page 26).

*connID*

On return, a pointer to the connection ID that identifies the connection to the fragment. You can pass this ID to other Code Fragment Manager functions.

*mainAddr*

On return, a pointer to the main address of the fragment. The value returned is specific to the fragment itself. Your application can use this parameter for its own purposes.

*errMessage*

On return, the name of the fragment that could not successfully be loaded. This parameter is meaningful only if the call to `GetDiskFragment` fails.

### Return Value

A result code. See [“Code Fragment Manager Result Codes”](#) (page 33). The `kFindLib` constant in the `loadFlags` parameter specifies that the Code Fragment Manager search for the specified fragment. If the fragment is already prepared and connected to your application, `GetDiskFragment` returns `fragNoErr`. If the specified fragment is not found, `GetDiskFragment` returns the result code `fragLibNotFound`. If the specified fragment is found but could not be connected to your application, the function returns `fragLibConnErr`.



## Deprecated Code Fragment Manager Functions

**Discussion**

Loading involves finding the specified fragment, reading it into memory (if it is not already in memory), and preparing it for execution. The Code Fragment Manager attempts to resolve all symbols imported by the fragment; to do so may involve loading import libraries.

If the fragment loading fails, the Code Fragment Manager returns an error code. Note, however, that the error encountered is not always in the fragment you asked to load. Rather, the error might have occurred while attempting to load an import library that the fragment you want to load depends on. For this reason, the Code Fragment Manager also returns, in the `errMessage` parameter, the name of the fragment that caused the load to fail. Although fragment names are restricted to 63 characters, the `errMessage` parameter is declared as type `Str255`; doing this allows future versions of the Code Fragment Manager to return a more informative message in the `errMessage` parameter.

**Availability**

Modified in Carbon. Available in CarbonLib 1.0 and later when Code Fragment Manager 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Carbon Porting Notes**

On Mac OS X, `GetDiskFragment` does not include the folder containing the root fragment (assuming that it is different from the application fragment) in its search path for import libraries. For example, say your application used a special folder to store plugins. If that folder also contained special libraries for those plugins, then calling `GetDiskFragment` to load a plugin would not find those libraries.

The workaround is to make sure that any import libraries you require are in the Code Fragment Manager's search path (such as by designating an application library subfolder in the code fragment resource, or placing the libraries in the application's container). For more details of how the Code Fragment Manager searches for import libraries, see Mac OS Runtime Architectures.

**Declared In**

`CodeFragments.h`

**GetIndSymbol**

Gets information about the exported symbols in a fragment. (Deprecated in Mac OS X v10.5.)

```
OSErr GetIndSymbol (
    CFragConnectionID connID,
    long symIndex,
    Str255 symName,
    Ptr *symAddr,
    CFragSymbolClass *symClass
);
```

**Parameters**

*connID*

A connection ID.

*symIndex*

A symbol index. This index is zero-based. That is, the value of this parameter should be between zero and the number of symbols - 1 (where the number of symbols is determined by calling the [CountSymbols](#) (page 38) function).

## Deprecated Code Fragment Manager Functions

*symName*

On return, the name of the indicated symbol.

*symAddr*

On return, a pointer to the address of the indicated symbol.

*symClass*

On return, a pointer to the class of the indicated symbol. See [“Load Flag, Symbol Class, and Fragment Locator Constants”](#) (page 26).

**Return Value**

A result code. See [“Code Fragment Manager Result Codes”](#) (page 33).

**Discussion**

If `GetIndSymbol` executes successfully, it returns the symbol’s name, starting address, and class in the `symName`, `symAddr`, and `symClass` parameters, respectively. A fragment’s exported symbols are retrieved in no predetermined order.

**Availability**

Available in CarbonLib 1.0 and later when Code Fragment Manager 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`CodeFragments.h`

**GetMemFragment**

Prepares a memory-based fragment for subsequent execution. (Deprecated in Mac OS X v10.5.)

```
OSErr GetMemFragment (
    void *memAddr,
    UInt32 length,
    ConstStr63Param fragName,
    CFragLoadOptions options,
    CFragConnectionID *connID,
    Ptr *mainAddr,
    Str255 errMessage
);
```

**Parameters***memAddr*

The address of the fragment.

*length*

The size, in bytes, of the fragment.

*fragName*

The name of the fragment. (This information is used primarily to allow you to identify the fragment during debugging.)

*loadFlags*

A flag that specifies the operation to perform on the fragment. The Code Fragment Manager recognizes the constants described in [“Load Flag, Symbol Class, and Fragment Locator Constants”](#) (page 26).

## Deprecated Code Fragment Manager Functions

*connID*

On return, a pointer to the connection ID that identifies the connection to the fragment. You can pass this ID to other Code Fragment Manager functions (for example, `CloseConnection`).

*mainAddr*

On return, a pointer to the main address of the fragment. The value returned is specific to the fragment itself.

*errorMessage*

On return, the name of the fragment that could not successfully be loaded. This parameter is meaningful only if the call to `GetMemFragment` fails.

**Return Value**

A result code. See “Code Fragment Manager Result Codes” (page 33).

**Discussion**

The `GetMemFragment` is most useful for handling code that is contained in a resource. You can read the resource data into memory using normal Resource Manager functions (for example, `Get1Resource`) and then call `GetMemFragment` to complete the processing required to prepare it for use (for example, to resolve any imports and execute the fragment’s initialization function).

You must lock the resource-based fragment into memory (for example, by calling `HLock`) before calling `GetMemFragment`. You must not unlock the memory until you have closed the connection to the fragment (by calling `CloseConnection`).

Loading involves finding the specified fragment, reading it into memory (if it is not already in memory), and preparing it for execution. The Code Fragment Manager attempts to resolve all symbols imported by the fragment; to do so may involve loading import libraries.

If the fragment loading fails, the Code Fragment Manager returns an error code. Note, however, that the error encountered is not always in the fragment you asked to load. Rather, the error might have occurred while attempting to load an import library that the fragment you want to load depends on. For this reason, the Code Fragment Manager also returns, in the `errorMessage` parameter, the name of the fragment that caused the load to fail. Although fragment names are restricted to 63 characters, the `errorMessage` parameter is declared as type `Str255`; doing this allows future versions of the Code Fragment Manager to return a more informative message in the `errorMessage` parameter.

**Availability**

Available in CarbonLib 1.0 and later when Code Fragment Manager 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

**Declared In**

`CodeFragments.h`

**GetSharedLibrary**

Locates and possibly also loads an import library into your application’s context. (Deprecated in Mac OS X v10.5.)

## Deprecated Code Fragment Manager Functions

```
OSErr GetSharedLibrary (
    ConstStr63Param libName,
    CFragArchitecture archType,
    CFragLoadOptions options,
    CFragConnectionID *connID,
    Ptr *mainAddr,
    Str255 errMessage
);
```

**Parameters***libName*

The name of an import library.

*archType*

The instruction set architecture of the import library. For the PowerPC architecture, use the constant `kPowerPCArch`. For the 680x0 architecture, use the constant `kMotorola68KArch`.

*loadFlags*

A flag that specifies the operation to perform on the import library. The Code Fragment Manager recognizes the constants described in [“Load Flag, Symbol Class, and Fragment Locator Constants”](#) (page 26).

*connID*

On return, a pointer to the connection ID that identifies the connection to the import library. You can pass this ID to other Code Fragment Manager functions.

*mainAddr*

On return, a pointer to the main address of the import library. The value returned is specific to the import library itself and is not used by the Code Fragment Manager.

*errMessage*

On return, the name of the fragment that could not successfully be loaded. This parameter is meaningful only if the call to `GetSharedLibrary` fails.

**Return Value**

A result code. See [“Code Fragment Manager Result Codes”](#) (page 33).

**Discussion**

The `GetSharedLibrary` function locates the import library named by the `libName` parameter and possibly also loads that import library into your application’s context. The actions of `GetSharedLibrary` depend on the action flag you pass in the `loadFlags` parameter; pass `kFindLib` to get the connection ID of an existing connection to the specified fragment, `kLoadLib` to load the specified fragment, or `kLoadNewCopy` to load the fragment with a new copy of the fragment’s data section.

The `GetSharedLibrary` function does not resolve any unresolved imports in your application. In particular, you cannot use it to resolve any weak imports in your code fragment.

Loading involves finding the specified fragment, reading it into memory (if it is not already in memory), and preparing it for execution. The Code Fragment Manager attempts to resolve all symbols imported by the fragment; to do so may involve loading import libraries.

If the fragment loading fails, the Code Fragment Manager returns an error code. Note, however, that the error encountered is not always in the fragment you asked to load. Rather, the error might have occurred while attempting to load an import library that the fragment you want to load depends on. For this reason, the Code Fragment Manager also returns, in the `errMessage` parameter, the name of the fragment that caused the load to fail. Although fragment names are restricted to 63 characters, the `errMessage` parameter is declared as type `Str255`; doing this allows future versions of the Code Fragment Manager to return a more informative message in the `errMessage` parameter.

## APPENDIX A

### Deprecated Code Fragment Manager Functions

#### **Availability**

Available in CarbonLib 1.0 and later when Code Fragment Manager 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Deprecated in Mac OS X v10.5.

Not available to 64-bit applications.

#### **Declared In**

CodeFragments.h



# Document Revision History

---

This table describes the changes to *Code Fragment Manager Reference*.

| Date       | Notes   |
|------------|---|
| 2005-07-07 | Added information for five result codes.  |
| 2003-06-17 | Added documentation for the function <a href="#">ConvertBundlePreLocator</a> (page 38). |
| 2003-02-01 | Updated formatting.   |
| 2002-02-01 | Last version of this document.  |

## REVISION HISTORY

### Document Revision History



# Index

---

## A

---

Architecture Constants 21

## C

---

- cfragAbortClosureErr **constant** 35
- cfragArchitectureErr **constant** 35
- CFragCFBundleLocator **structure** 8
- cfragCFMInternalErr **constant** 34
- cfragCFMStartupErr **constant** 34
- cfragCFragRsrcErr **constant** 35
- CFragClosureID **data type** 8
- cfragClosureIDErr **constant** 35
- CFragConnectionID **data type** 8
- cfragConnectionIDErr **constant** 33
- CFragContainerID **data type** 9
- cfragContainerIDErr **constant** 35
- CFragContextID **data type** 9
- cfragContextIDErr **constant** 33
- cfragDupRegistrationErr **constant** 33
- cfragExecFileRefErr **constant** 35
- cfragFileSizeErr **constant** 35
- cfragFirstErrCode **constant** 33
- cfragFirstReservedCode **constant** 35
- cfragFragmentCorruptErr **constant** 34
- cfragFragmentFormatErr **constant** 34
- cfragFragmentUsageErr **constant** 35
- CFragHFSDiskFlatLocator **data type** 9
- CFragHFSLocator **data type** 9
- CFragHFSLocatorPtr **data type** 9
- CFragHFSSMemoryLocator **data type** 10
- CFragHFSSegmentedLocator **data type** 10
- cfragImportTooNewErr **constant** 34
- cfragImportTooOldErr **constant** 34
- cfragInitAtBootErr **constant** 34
- CFragInitBlock **data type** 10
- CFragInitBlockPtr **data type** 10
- CFragInitFunction **callback** 6
- cfragInitFunctionErr **constant** 34
- cfragInitLoopErr **constant** 34
- cfragInitOrderErr **constant** 34
- cfragLastErrCode **constant** 36
- cfragMapFileErr **constant** 35
- cfragNoApplicationErr **constant** 34
- cfragNoClientMemErr **constant** 34
- cfragNoIDsErr **constant** 34
- cfragNoLibraryErr **constant** 33
- cfragNoPositionErr **constant** 34
- cfragNoPrivateMemErr **constant** 34
- cfragNoRegistrationErr **constant** 35
- cfragNoSectionErr **constant** 33
- cfragNoSymbolErr **constant** 33
- cfragNotClosureErr **constant** 35
- cfragOutputLengthErr **constant** 35
- cfragReservedCode\_1 **constant** 36
- cfragReservedCode\_2 **constant** 36
- cfragReservedCode\_3 **constant** 36
- CFragResource **structure** 11
- CFragResourceExtensionHeader **structure** 12
- CFragResourceMember **structure** 12
- CFragResourceSearchExtension **structure** 13
- cfragRsrcForkErr **constant** 35
- cfragStdFolderErr **constant** 35
- CFragSystem7DiskFlatLocator **structure** 13
- CFragSystem7InitBlock **structure** 14
- CFragSystem7Locator **structure** 15
- CFragSystem7MemoryLocator **structure** 16
- CFragSystem7SegmentedLocator **structure** 17
- CFragTermProcedure **callback** 7
- cfragUnresolvedErr **constant** 34
- CFragUsage1Union **structure** 17
- CFragUsage2Union **structure** 18
- CFragWhere1Union **structure** 18
- CFragWhere2Union **structure** 18
- CloseConnection **function** (Deprecated in Mac OS X v10.5) 37
- Code Fragment Kind 22
- ConnectionID **data type** 18
- ConvertBundlePreLocator **function** (Deprecated in Mac OS X v10.5) 38

CountSymbols function (Deprecated in Mac OS X v10.5) 38

Current Resource Version 23

## D

---

Default Name Length 23

DiskFragment data type 19

## F

---

File Location 23

FindSymbol function (Deprecated in Mac OS X v10.5) 39

FragmentLocator data type 19

FragmentLocatorPtr data type 19

## G

---

GetDiskFragment function (Deprecated in Mac OS X v10.5) 40

GetIndSymbol function (Deprecated in Mac OS X v10.5) 41

GetMemFragment function (Deprecated in Mac OS X v10.5) 42

GetSharedLibrary function (Deprecated in Mac OS X v10.5) 43

## I

---

InitBlock data type 19

InitBlockPtr data type 20

## K

---

kAnyArchType constant 26

kAnyCFragArch constant 21

kApplicationCFrag constant 32

kCFBundleCFragLocator constant 30

kCFBundlePreCFragLocator constant 30

kCFMRsrcID constant 28

kCFMRsrcType constant 28

kCFragAllFileTypes constant 24

kCFragGoesToEOF 23

kCFragLibraryFileType constant 24

kCFragLibUsageMapPrivatelyMask 23

kCFragLibUsageMapPrivatelyMask constant 23

kCFragResourceID constant 24

kCFragResourceSearchExtensionKind 24

kCFragResourceType 24

kCFragResourceType constant 24

kCodeCFragSymbol constant 31

kCodeSym constant 27

kCompiledCFragArch 24

kCompiledCFragArch constant 24

kDataCFragSymbol constant 31

kDataForkCFragLocator constant 30

kDataSym constant 27

kDropInAdditionCFrag constant 32

kFindCFrag constant 29

kFindLib constant 27

kFirstCFragUpdate constant 22

kFullLib constant 28

kGlueCFragSymbol constant 31

kGlueSym constant 27

kImportLibraryCFrag constant 32

kInMem constant 28

kIsApp constant 28

kIsCompleteCFrag constant 22

kIsDropIn constant 28

kIsLib constant 28

kLoadCFrag 25

kLoadCFrag constant 25

kLoadLib constant 27

kLoadNewCopy constant 27

kMemoryCFragLocator constant 30

kMotorola68K constant 25

kMotorola68KArch constant 26

kMotorola68KCFragArch constant 21

kNamedFragmentCFragLocator constant 30

kNewCFragCopy constant 27

kNoConnectionID constant 26

kNoLibName constant 26

kNullCFragVersion constant 33

kOnDiskFlat constant 28

kOnDiskSegmented constant 28

kPowerPC 25

kPowerPC constant 25

kPowerPCArch constant 26

kPowerPCCFragArch constant 21

kPrivateCFragCopy constant 29

kReferenceCFrag constant 29

kResourceCFragLocator constant 30

kSHLBFFileType constant 29

kStubLibraryCFrag constant 32

kTOCCFragSymbol constant 31

kTOCSym constant 27

kTVectorCFragSymbol constant 31

kTVectSym constant 27

kUnresolvedCFragSymbolAddress **constant** [31](#)  
kUnresolvedSymbolAddress **constant** [29](#)  
kUpdateLib **constant** [28](#)  
kUseInPlace **constant** [27](#)  
kWeakStubLibraryCFrag **constant** [32](#)  
kWholeFork **constant** [28](#)  
kWildcardCFragVersion **constant** [33](#)

## L

---

Load Flag, Symbol Class, and Fragment Locator Constants [26](#)  
Load Options [29](#)  
LoadFlags **data type** [20](#)  
Locator Kind [30](#)

## M

---

MemFragment **data type** [20](#)

## S

---

SegmentedFragment **data type** [20](#)  
Symbol Class Constants [31](#)  
SymClass **data type** [21](#)

## U

---

Unresolved Symbol Address [31](#)  
Usage Constants [32](#)

## V

---

Version Number [33](#)