
Debugging Programming Topics for Core Foundation

[Core Foundation](#) > [Performance](#)



2003-01-17



Apple Inc.
© 2003 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, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

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

Introduction to Debugging 7

Organization of This Document 7

Core Foundation Libraries 9

Debuggers 11

Handling Errors 13

Debugging Utilities 15

Document Revision History 17

Listings

Debugging Utilities 15

Listing 1 Calling the inspection functions in a debugger 15

Introduction to Debugging

This topic presents information that you might find useful when you begin to work with the Core Foundation programming interfaces. It offers tips and suggestions for coding and debugging.

Organization of This Document

The concepts and tasks covered in this topic are:

- [“Core Foundation Libraries”](#) (page 9)
- [“Debuggers”](#) (page 11)
- [“Handling Errors”](#) (page 13)
- [“Debugging Utilities”](#) (page 15)

Core Foundation Libraries

When Core Foundation is installed on a Mac OS 9 systems, you have two libraries at your disposal: a production library (`CoreFoundationLib`) and a debug library (`CoreFoundationLib Debug`). When you use the debug version of the library you automatically get assertion checking on common programming errors such as

- accessing a collection out of bounds
- attempting to change an immutable object
- passing an argument of a wrong type

Currently assertions print a log message and cause a break in the debugger.

The production version of the Core Foundation library is optimized and performs no error checking; invalid parameters, even references to the wrong Core Foundation type, will cause indeterminate and probably incorrect behavior. Obviously, you should use the debug version of the Core Foundation library during development to catch errors in programming.

On Mac OS X, Core Foundation is a framework: a package consisting of a dynamic shared library and associated resources, including public header files. `CoreFoundation.framework` also includes a production version and a debug version of the library. To use the debug version, set the environment variable `DYLD_IMAGE_SUFFIX` to `_debug` before running the application; the dynamic linker will then check for and link against the debug versions of any dynamic libraries, such as Core Foundation, that your application uses.

Debuggers

You can use the debugger to reveal the contents of Core Foundation objects, but how you use it depends on your development platform.

- **Mac OS X:** Because Core Foundation uses opaque types, `CFShow` is a convenient tool for discovering the contents of objects. Call `CFShow` within the debugger or programmatically on an object to print a description of that object. The description is appropriate to the type of object. For example, calling `CFShow` on `CFNumber` prints a number; for a `CFArray`, `CFShow` prints information general to the array as well as information on each element of the array.
- **Mac OS 9:** Because you cannot use the debugger to call functions (such as `CFShow`) at runtime, you must use other techniques to examine memory. Consult the Core Foundation release notes for the latest debugging tips and techniques.

Handling Errors

Mostly in the interest of code simplification, Core Foundation functions do not follow the Mac OS `OSErr` convention for error reporting. However, if you rigorously test your program with the debug version of the Core Foundation library (or framework), then the possibilities for error are much reduced.

Yet there can be occasions when you want to test whether a function succeeds in its intended purpose and proceed conditionally from there. By a common-sense interpretation of returned values (and sometimes by a quick check of the reference documentation), you can usually determine the reason for failure, as in these examples:

- If functions that are supposed to create or copy objects return `NULL`, the probable reason is an error in the allocation of memory.
- If functions that are supposed to return an element from an index-based collection (that is, a `CFArray`) return an indeterminate value, the probable cause is an index parameter that is out of bounds.
- If a function is supposed to return a `CFArray` of values, each element of which satisfies some condition specified by the function, and an empty `CFArray` is returned, then no object has satisfied that condition.

Some Core Foundation functions return a value of type `Boolean` or an `enum` constant or some other explicitly typed indication of the reason for an outcome. Consult the Core Foundation reference documentation for an explanation of these return values.

Debugging Utilities

Because Core Foundation types are opaque, it is difficult to inspect the Core Foundation objects created by your code using traditional means. However, Core Foundation implements a couple of functions that print descriptions of Core Foundation objects, either from your code or in any debugger that supports functions calls.

The first of these functions is `CFShow`, which takes a reference to any Core Foundation object (that is, its sole parameter is typed as `CFTypeRef`). This function calls the `CFDescription` function on the object, causing it to return a reference to a `CFString` object containing the description. The `CFShow` function then prints this description to the output device.

The second “inspection” function is `CFShowStr`, which takes a reference to a `CFString` object. This function displays the attributes of a `CFString` object but not its contents.

[Listing 1](#) (page 15) shows the information printed by both functions in a debugger.

Listing 1 Calling the inspection functions in a debugger

```
(gdb) s
stringGettingContentsAsCStringExample () at StringExample.c:105
105         str = CFStringCreateWithCString(NULL, "Hello World",
CFStringGetSystemEncoding());
(gdb) n
111         bytes = CFStringGetCStringPtr(str, CFStringGetSystemEncoding());
(gdb) call CFShow(str)
Hello World
$1 = 0
(gdb) call CFShowStr(str)

Length 11
IsEightBit 1
HasLengthByte 1
HasNullByte 1
InlineContents 1
Allocator SystemDefault
Mutable 0
Contents 0x4e7c0
$2 = 17
(gdb)
```


Document Revision History

This table describes the changes to *Debugging Programming Topics for Core Foundation*.

Date	Notes
2003-01-17	Converted existing Core Foundation documentation into topic format. Added revision history.

