
Debugger Services Reference

[Carbon](#) > [Performance](#)



2003-01-01



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, Carbon, 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

Debugger Services Reference 5

Overview	5
Functions by Task	5
Using Debugger Services	5
Managing Callback UPPs	6
Functions	6
DebugAssert	6
DisposeDebugAssertOutputHandlerUPP	7
DisposeDebugComponent	7
DisposeDebugComponentCallbackUPP	8
GetDebugComponentInfo	8
GetDebugOptionInfo	8
InstallDebugAssertOutputHandler	9
InvokeDebugAssertOutputHandlerUPP	10
InvokeDebugComponentCallbackUPP	10
NewDebugAssertOutputHandlerUPP	10
NewDebugComponent	11
NewDebugComponentCallbackUPP	11
NewDebugOption	11
SetDebugOptionValue	12
TaskLevel	13
Callbacks	13
DebugAssertOutputHandlerProcPtr	13
DebugComponentCallbackProcPtr	14
Data Types	15
DebugAssertOutputHandlerUPP	15
DebugComponentCallbackUPP	15
Constants	16
Interrupt Level Masks	16
Unmapped Addresses	17
Debug Option Types	17
Commands for Debug Option Callbacks	17
Result Codes	18

Document Revision History 19

Index 21

Debugger Services Reference

Framework:	CoreServices/CoreServices.h
Declared in	Debugging.h

Overview

Debugger Services is a Carbon API that provides standard exception handling and assertion functions to assist you in debugging Mac OS applications.

Functions by Task

Using Debugger Services

[NewDebugComponent](#) (page 11)

Registers a component with Debugger Services.

[NewDebugOption](#) (page 11)

Registers a new debug option with Debugger Services.

[GetDebugComponentInfo](#) (page 8)

Returns the signature and name of a registered component.

[GetDebugOptionInfo](#) (page 8)

Returns information about the debug option of a registered component.

[SetDebugOptionValue](#) (page 12)

Modifies the setting of a registered debug option.

[DisposeDebugComponent](#) (page 7)

Removes a component registration and all related debug options.

[DebugAssert](#) (page 6)

Displays an assertion message using the current output handler.

[InstallDebugAssertOutputHandler](#) (page 9)

Installs an output handler for `DebugAssert` to call in place of `DebugStr`, the default handler.

[TaskLevel](#) (page 13)

Provides information about the task interrupt level, if the task is running at interrupt-time.

Managing Callback UPPs

[NewDebugAssertOutputHandlerUPP](#) (page 10)

[InvokeDebugAssertOutputHandlerUPP](#) (page 10)

[DisposeDebugAssertOutputHandlerUPP](#) (page 7)

[NewDebugComponentCallbackUPP](#) (page 11)

[InvokeDebugComponentCallbackUPP](#) (page 10)

[DisposeDebugComponentCallbackUPP](#) (page 8)

Functions

DebugAssert

Displays an assertion message using the current output handler.

```
void DebugAssert (
    OSType componentSignature,
    UInt32 options,
    const char *assertionString,
    const char *exceptionLabelString,
    const char *errorString,
    const char *fileName,
    long lineNumber,
    void *value
);
```

Parameters

componentSignature

The unique signature of the component causing the assertion.

options

Reserved for use by Apple.

assertionString

A pointer to a string containing the assertion, or NULL.

exceptionLabelString

A pointer to a string containing the exceptionLabel, or NULL.

errorString

A pointer to the error string, or NULL.

fileName

A pointer to the file name or path name generated by the preprocessor `__FILE__` identifier, or NULL.

lineNumber

The line number in the file (generated by the preprocessor `__LINE__` identifier), or 0 (zero).

value

A value associated with the assertion, or NULL.

Discussion

The `DEBUGASSERTMSG` macro calls this function (by default) to display assertion messages. To redirect the output from this function, use [InstallDebugAssertOutputHandler](#) (page 9) to install a custom output handler.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

DisposeDebugAssertOutputHandlerUPP

```
void DisposeDebugAssertOutputHandlerUPP (
    DebugAssertOutputHandlerUPP userUPP
);
```

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

DisposeDebugComponent

Removes a component registration and all related debug options.

```
OSStatus DisposeDebugComponent (
    OSType componentSignature
);
```

Parameters

componentSignature

The unique signature of a component.

Return Value

A result code. If the result is non-zero, the Notification Manager cannot remove the debug options.

Availability

Available in Mac OS X v10.0 and later.

Declared In

Debugging.h

DisposeDebugComponentCallbackUPP

```
void DisposeDebugComponentCallbackUPP (
    DebugComponentCallbackUPP userUPP
);
```

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

GetDebugComponentInfo

Returns the signature and name of a registered component.

```
OSStatus GetDebugComponentInfo (
    UInt32 index,
    OSType *componentSignature,
    Str255 componentName
);
```

Parameters

index

An index into a list of registered components (one-based).

componentSignature

A pointer to an OSType, provided by the caller to receive the unique signature of the specified component.

componentName

A string buffer, provided by the caller to receive the component name.

Return Value

A result code. If *index* is not valid, the result code is `debuggingNoMatchErr`.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

GetDebugOptionInfo

Returns information about the debug option of a registered component.


```
OSStatus GetDebugOptionInfo (
    UInt32 index,
    OSType componentSignature,
    SInt32 *optionSelectorNum,
    Str255 optionName,
    Boolean *optionSetting
);
```

Parameters*index*

An index into a list of registered debug options (zero-based). You should use the constant [kComponentDebugOption](#) (page 17).

componentSignature

The unique signature of your registered component.

optionSelectorNum

A pointer to an integer, provided by the caller to receive the option selector number.

optionName

A string buffer, provided by the caller to receive the option name.

optionSetting

A pointer to a Boolean, provided by the caller to receive the current option setting.

Return Value

A result code. Debugger Services returns `debuggingNoMatchErr` if the index is not valid, `debuggingInvalidSignatureErr` if the component is not registered, or `noErr`.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

InstallDebugAssertOutputHandler

Installs an output handler for `DebugAssert` to call in place of `DebugStr`, the default handler.

```
void InstallDebugAssertOutputHandler (
    DebugAssertOutputHandlerUPP handler
);
```

Parameters*handler*

The custom output handler to install, or NULL to switch back to `DebugStr`.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

InvokeDebugAssertOutputHandlerUPP

```
void InvokeDebugAssertOutputHandlerUPP (
    OSType componentSignature,
    UInt32 options,
    const char *assertionString,
    const char *exceptionLabelString,
    const char *errorString,
    const char *fileName,
    long lineNumber,
    void *value,
    ConstStr255Param outputMsg,
    DebugAssertOutputHandlerUPP userUPP
);
```

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

InvokeDebugComponentCallbackUPP

```
void InvokeDebugComponentCallbackUPP (
    SInt32 optionSelectorNum,
    UInt32 command,
    Boolean *optionSetting,
    DebugComponentCallbackUPP userUPP
);
```

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

NewDebugAssertOutputHandlerUPP

```
DebugAssertOutputHandlerUPP NewDebugAssertOutputHandlerUPP (
    DebugAssertOutputHandlerProcPtr userRoutine
);
```

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

NewDebugComponent

Registers a component with Debugger Services.

```
OSStatus NewDebugComponent (
    OSType componentSignature,
    ConstStr255Param componentName,
    DebugComponentCallbackUPP componentCallback
);
```

Parameters

componentSignature

The unique signature of a new component.

componentName

A displayable string that names the new component.

componentCallback

A universal procedure pointer (UPP) to a debug component callback function, provided by the caller for working with options.

Return Value

A result code. See [Debugger Services Result Codes](#) (page 18).

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

NewDebugComponentCallbackUPP

```
DebugComponentCallbackUPP NewDebugComponentCallbackUPP (
    DebugComponentCallbackProcPtr userRoutine
);
```

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

NewDebugOption

Registers a new debug option with Debugger Services.

```
OSStatus NewDebugOption (
    OSType componentSignature,
    SInt32 optionSelectorNum,
    ConstStr255Param optionName
);
```

Parameters*componentSignature*

The unique signature of a registered component.

optionSelectorNum

The selector number of the new debug option.

optionName

A displayable string that names this debug option.

Return ValueA result code. See [Debugger Services Result Codes](#) (page 18).**Availability**

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

SetDebugOptionValue

Modifies the setting of a registered debug option.

```
OSStatus SetDebugOptionValue (
    OSType componentSignature,
    SInt32 optionSelectorNum,
    Boolean newOptionSetting
);
```

Parameters*componentSignature*

The unique signature of a registered component.

optionSelectorNum

The selector number of a registered debug option.

newOptionSetting

The new setting for the option.

Return ValueA result code. Debugger Services returns `debuggingInvalidOptionErr` if the selector number is not valid, `debuggingInvalidSignatureErr` if the component is not registered, or `noErr`.**Availability**

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

TaskLevel

Provides information about the task interrupt level, if the task is running at interrupt-time.

```
UInt32 TaskLevel (
    void
);
```

Return Value

The current task interrupt level. If the return value is 0, the task is (probably) running at non-interrupt time. Otherwise, one of the `TaskLevel` masks can be used to learn more.

Availability

Available in CarbonLib 1.0 and later.

Available in Mac OS X version 10.0 and later.

Declared In

Debugging.h

Callbacks

DebugAssertOutputHandlerProcPtr

Defines a pointer to a function that handles the output from [DebugAssert](#) (page 6).

```
typedef void (*DebugAssertOutputHandlerProcPtr)
(
    OSType componentSignature,
    UInt32 options,
    const char * assertionString,
    const char * exceptionLabelString,
    const char * errorString,
    const char * fileName,
    long lineNumber,
    void * value,
    ConstStr255Param outputMsg
);
```

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

```
void MyDebugAssertOutputHandler (
    OSType componentSignature,
    UInt32 options,
    const char * assertionString,
    const char * exceptionLabelString,
    const char * errorString,
    const char * fileName,
    long lineNumber,
    void * value,
    ConstStr255Param outputMsg
);
```

Parameters*componentSignature*

The unique signature of the component causing the assertion.

options

Reserved for use by Apple.

assertionString

The name of the assertion, or NULL.

exceptionLabelString

The exception label, or NULL.

errorString

The description of an error condition, or NULL.

*fileName*The file or path name (generated by the preprocessor `__FILE__` identifier), or NULL.*fileName*The file or path name (generated by the preprocessor `__FILE__` identifier), or NULL.*lineNumber*The line number in the file (generated by the preprocessor `__LINE__` identifier), or 0 (zero).*value*

A value associated with the assertion, or NULL.

*outputMsg*The string that the caller (`DebugAssert`) normally passes to `DebugStr` when a custom output handler isn't installed.**Discussion**

The parameters (excluding `outputMsg`) are the raw values passed to `DebugAssert` when an exception occurs. A custom output handler can safely ignore these parameters and simply redirect the output message (for example, to a log file).

Availability

Available in Mac OS X v10.0 and later.

Declared In`Debugging.h`**DebugComponentCallbackProcPtr**

Defines a pointer to a function that Debugger Services calls to read or modify the debug option settings defined by a component.

```
typedef void (*DebugComponentCallbackProcPtr)
(
    SInt32 optionSelectorNum,
    UInt32 command,
    Boolean * optionSetting
);
```

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

```
void MyDebugComponentCallback (
    SInt32 optionSelectorNum,
```

```

    UInt32 command,
    Boolean * optionSetting
);

```

Parameters*optionSelectorNum*

A component debug option, previously defined by calling [NewDebugOption](#) (page 11).

command

Specifies the operation to be performed—`kGetDebugOption` to get current setting, or `kSetDebugOption` to modify the setting.

optionSetting

A pointer to a Boolean that Debugger Services uses to

- pass in the new setting, if `command` is `kSetDebugOption`
- receive the result of the operation, if `command` is `kGetDebugOption`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Debugging.h`

Data Types

DebugAssertOutputHandlerUPP

Defines a universal procedure pointer (UPP) type for a custom assertion output handler.

```
typedef DebugAssertOutputHandlerProcPtr DebugAssertOutputHandlerUPP;
```

Discussion

For information about custom assertion output handlers, see [DebugAssertOutputHandlerProcPtr](#) (page 13).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`Debugging.h`

DebugComponentCallbackUPP

Defines a universal procedure pointer (UPP) type for a custom component debug option callback.

```
typedef DebugComponentCallbackProcPtr DebugComponentCallbackUPP;
```

Discussion

For information about custom component debug option callbacks, see [DebugComponentCallbackProcPtr](#) (page 14).

Availability

Available in Mac OS X v10.0 and later.

Declared In

Debugging.h

Constants

Interrupt Level Masks

Masks to determine what kind of tasks are executing at interrupt time.

```
enum {
    k68kInterruptLevelMask = 0x00000007,
    kInVBLTaskMask = 0x00000010,
    kInDeferredTaskMask = 0x00000020,
    kInSecondaryIntHandlerMask = 0x00000040,
    kInNestedInterruptMask = 0x00000080
};
```

Constants

k68kInterruptLevelMask

68K interrupt levels 0 through 7.

Available in Mac OS X v10.0 and later.

Declared in Debugging.h.

kInVBLTaskMask

VBLs are executing.

Available in Mac OS X v10.0 and later.

Declared in Debugging.h.

kInDeferredTaskMask

Deferred tasks are executing.

Available in Mac OS X v10.0 and later.

Declared in Debugging.h.

kInSecondaryIntHandlerMask

Secondary interrupt handlers are executing.

Available in Mac OS X v10.0 and later.

Declared in Debugging.h.

kInNestedInterruptMask

The operating system is handling an interrupt.

Available in Mac OS X v10.0 and later.

Declared in Debugging.h.

DiscussionFor more information, see [TaskLevel](#) (page 13).

Unmapped Addresses

Addresses not mapped in Mac OS 8 or 9.

```
enum {
    kBlessedBusErrorBait = 0x68F168F1
};
```

Constants

`kBlessedBusErrorBait`

An address that will never be mapped in Mac OS 8 or 9.

Available in Mac OS X v10.0 and later.

Declared in `Debugging.h`.

Discussion

An exception occurs when an application tries to access the address `kBlessedBusErrorBait` in Mac OS 8 or 9, which makes it a good value to use when initializing pointers.

In Mac OS X, you should use `0x00000000` for this purpose.

Debug Option Types

Defines the debug option types supported by Debugger Services.

```
enum {
    kComponentDebugOption = 0
};
```

Constants

`kComponentDebugOption`

Specifies the component debug option type.

Available in Mac OS X v10.0 and later.

Declared in `Debugging.h`.

Discussion

For information about how this constant is used, see [GetDebugOptionInfo](#) (page 8).

Commands for Debug Option Callbacks

Defines the commands (or operations) that a debug option callback needs to implement.

```
enum {
    kGetDebugOption = 1,
    kSetDebugOption = 2
};
```

Constants

`kGetDebugOption`

The callback should return the current `Boolean` value of the specified debug option.

Available in Mac OS X v10.0 and later.

Declared in `Debugging.h`.

`kSetDebugOption`

The callback should modify the `Boolean` value of the specified debug option.

Available in Mac OS X v10.0 and later.

Declared in `Debugging.h`.

Result Codes

The most common result codes returned by Debugger Services are listed in the table below.

Result Code	Value	Description
<code>debuggingExecutionContextErr</code>	-13880	routine cannot be called at this time Available in Mac OS X v10.0 and later.
<code>debuggingDuplicateSignatureErr</code>	-13881	<code>componentSignature</code> already registered Available in Mac OS X v10.0 and later.
<code>debuggingDuplicateOptionErr</code>	-13882	<code>optionSelectorNum</code> already registered Available in Mac OS X v10.0 and later.
<code>debuggingInvalidSignatureErr</code>	-13883	<code>componentSignature</code> not registered Available in Mac OS X v10.0 and later.
<code>debuggingInvalidOptionErr</code>	-13884	<code>optionSelectorNum</code> is not registered Available in Mac OS X v10.0 and later.
<code>debuggingInvalidNameErr</code>	-13885	<code>componentName</code> or <code>optionName</code> is invalid (NULL) Available in Mac OS X v10.0 and later.
<code>debuggingNoCallbackErr</code>	-13886	debugging component has no callback Available in Mac OS X v10.0 and later.
<code>debuggingNoMatchErr</code>	-13887	debugging component or option not found at this index Available in Mac OS X v10.0 and later.

Document Revision History

This table describes the changes to *Debugger Services Reference*.

Date	Notes
2003-01-01	Incorporated some of the information found in the Debugging.h comments. Published a PDF version of this document.

REVISION HISTORY

Document Revision History

Index

C

Commands for Debug Option Callbacks [17](#)

D

Debug Option Types [17](#)

DebugAssert [function 6](#)

DebugAssertOutputHandlerProcPtr [callback 13](#)

DebugAssertOutputHandlerUPP [data type 15](#)

DebugComponentCallbackProcPtr [callback 14](#)

DebugComponentCallbackUPP [data type 15](#)

debuggingDuplicateOptionErr [constant 18](#)

debuggingDuplicateSignatureErr [constant 18](#)

debuggingExecutionContextErr [constant 18](#)

debuggingInvalidNameErr [constant 18](#)

debuggingInvalidOptionErr [constant 18](#)

debuggingInvalidSignatureErr [constant 18](#)

debuggingNoCallbackErr [constant 18](#)

debuggingNoMatchErr [constant 18](#)

DisposeDebugAssertOutputHandlerUPP [function 7](#)

DisposeDebugComponent [function 7](#)

DisposeDebugComponentCallbackUPP [function 8](#)

G

GetDebugComponentInfo [function 8](#)

GetDebugOptionInfo [function 8](#)

I

InstallDebugAssertOutputHandler [function 9](#)

Interrupt Level Masks [16](#)

InvokeDebugAssertOutputHandlerUPP [function 10](#)

InvokeDebugComponentCallbackUPP [function 10](#)

K

k68kInterruptLevelMask [constant 16](#)

kBlessedBusErrorBait [constant 17](#)

kComponentDebugOption [constant 17](#)

kGetDebugOption [constant 17](#)

kInDeferredTaskMask [constant 16](#)

kInNestedInterruptMask [constant 16](#)

kInSecondaryIntHandlerMask [constant 16](#)

kInVBLTaskMask [constant 16](#)

kSetDebugOption [constant 18](#)

N

NewDebugAssertOutputHandlerUPP [function 10](#)

NewDebugComponent [function 11](#)

NewDebugComponentCallbackUPP [function 11](#)

NewDebugOption [function 11](#)

S

SetDebugOptionValue [function 12](#)

T

TaskLevel [function 13](#)

U

Unmapped Addresses [17](#)