

---

# HI Archive Reference

[Carbon](#) > [Human Interface Toolbox](#)



2005-08-11



Apple Inc.  
© 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, Cocoa, 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

---

## HIArchive Reference 5

Overview	5
Functions by Task	5
Storing Objects in an Archive	5
Retrieving Objects from an Archive	6
Miscellaneous Function	6
Functions	6
HIArchiveCopyDecodedCType	6
HIArchiveCopyEncodedData	7
HIArchiveCreateForDecoding	7
HIArchiveCreateForEncoding	8
HIArchiveDecodeBoolean	8
HIArchiveDecodeNumber	9
HIArchiveEncodeBoolean	10
HIArchiveEncodeCType	10
HIArchiveEncodeNumber	11
HIArchiveGetTypeID	12
Data Types	12
HIArchiveRef	12
Constants	12
Archive Decoding Option Constant	12
Result Codes	13

---

## Document Revision History 15

---

## Index 17

---



# HIArchive Reference

---

<b>Framework:</b>	Carbon/Carbon.h
<b>Declared in</b>	HIArchive.h

## Overview

HIArchive provides a convenient and standardized mechanism for flattening data objects so they can be stored in memory or on disk. Applications can use these archives whenever they need to package complex data. For example, you can use archives to:

- Store document data
- Transfer data using pasteboards, drag-and-drop, streams, or Apple events
- Store localization strings and user interface elements in the same package

HIArchive encodes archives in the binary property list format. You can convert archives to a text XML format using the `plutil` property list tool accessible from Terminal. HIArchive is comparable to (and uses the same underlying mechanism as) the Cocoa `NSKeyedArchiver/Unarchiver` classes.

For details about using HIArchive, see *HIArchive Programming Guide*.

HIArchive is available in Mac OS X version 10.4 and later.

## Functions by Task

### Storing Objects in an Archive

- [HIArchiveCreateForEncoding](#) (page 8)  
Creates an HIArchive object to store objects.
- [HIArchiveEncodeBoolean](#) (page 10)  
Stores a Boolean value in an archive.
- [HIArchiveEncodeNumber](#) (page 11)  
Stores a number in an archive.
- [HIArchiveEncodeCftype](#) (page 10)  
Stores a Cftype object in an archive.
- [HIArchiveCopyEncodedData](#) (page 7)  
Compresses an archive for storage.

## Retrieving Objects from an Archive

- [HIArchiveCreateForDecoding](#) (page 7)  
Creates an HIArchive object to retrieve objects.
- [HIArchiveDecodeBoolean](#) (page 8)  
Retrieves a Boolean value from an archive.
- [HIArchiveDecodeNumber](#) (page 9)  
Retrieves a number from an archive.
- [HIArchiveCopyDecodedCType](#) (page 6)  
Retrieves a CType object from an archive.

## Miscellaneous Function

- [HIArchiveGetTypeID](#) (page 12)  
Obtains the CType ID for HIArchive objects.

# Functions

### HIArchiveCopyDecodedCType

Retrieves a CType object from an archive.

```
OSStatus HIArchiveCopyDecodedCType (
    HIArchiveRef inDecoder,
    CFStringRef inKey,
    CTypeRef *outCType
);
```

#### Parameters

*inDecoder*

The archive holding the CType object to retrieve.

*inKey*

A Core Foundation string key identifying the CType object to retrieve.

*outCType*

On return, *outCType* points to the retrieved CType object.

#### Return Value

A result code.

#### Discussion

You also use this function for retrieving HIObjects and objects subclassed from HIObject.

#### Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

#### Declared In

HIArchive.h

## HIArchiveCopyEncodedData

Compresses an archive for storage.

```
OSStatus HIArchiveCopyEncodedData (
    HIArchiveRef inEncoder,
    CFDataRef *outData
);
```

### Parameters

*inEncoder*

The archive to compress.

*outData*

On return, *outData* points to the compressed archive.

### Return Value

A result code.

### Discussion

When you have finished adding data to an archive, calling `HIArchiveCopyEncodedData` compresses the data and returns it to you as a `CFData` object. You can use the returned data reference to store or transfer the data as you choose, for example writing it to a file or copying it to a pasteboard.

After compression, you can release the original `HIArchive` reference by calling `CFRelease`.

### Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

### Declared In

`HIArchive.h`

## HIArchiveCreateForDecoding

Creates an `HIArchive` object to retrieve objects.

```
OSStatus HIArchiveCreateForDecoding (
    CFDataRef inData,
    OptionBits inOptions,
    HIArchiveRef *outDecoder
);
```

### Parameters

*inData*

A `CFData` reference pointing to archived data. This archive was originally written to a data stream using `HIArchiveCopyEncodedData` (page 7). This data reference does not have to be the one originally returned by `HIArchiveCopyEncodedData` (page 7), but it must contain a copy of the same data.

*inOptions*

Any decoding options. Currently the only option is `kHIArchiveDecodeSuperclassForUnregisteredObjects`.

*outDecoder*

On return, *outDecoder* points to the newly created `HIArchive` object.

**Return Value**

A result code.

**Discussion**

You use this function when you want to retrieve data from an existing HIArchive.

**Availability**

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

**Declared In**

HIArchive.h

**HIArchiveCreateForEncoding**

Creates an HIArchive object to store objects.

```
OSStatus HIArchiveCreateForEncoding (
    HIArchiveRef *outEncoder
);
```

**Parameters**

*outEncoder*

On return, *outEncoder* points to the newly created HIArchive object.

**Return Value**

A result code.

**Discussion**

Before you can archive any objects, you must create an HIArchive object in which to store them.

**Availability**

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

**Declared In**

HIArchive.h

**HIArchiveDecodeBoolean**

Retrieves a Boolean value from an archive.

```
OSStatus HIArchiveDecodeBoolean (
    HIArchiveRef inDecoder,
    CFStringRef inKey,
    Boolean *outBoolean
);
```

**Parameters**

*inDecoder*

The archive holding the Boolean value.

*inKey*

A Core Foundation string key identifying the Boolean to retrieve.



*outBoolean*

On return, *outBoolean* points to the retrieved Boolean value.

**Return Value**

A result code.

**Discussion**

This function is a convenience wrapper that calls [HIArchiveCopyDecodedCFTYPE](#) (page 6) to obtain a CFBoolean value.

**Availability**

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

**Declared In**

HIArchive.h

**HIArchiveDecodeNumber**

Retrieves a number from an archive.

```
OSStatus HIArchiveDecodeNumber (
    HIArchiveRef inDecoder,
    CFStringRef inKey,
    CFNumberType inNumberType,
    void *outNumberValue
);
```

**Parameters**

*inDecoder*

The archive holding the number to retrieve.

*inKey*

A Core Foundation string key identifying the number to retrieve.

*inNumberType*

A CFNumber type identifying the type of number value to be retrieved. For example, `kCFNumberSInt32Type`. See [CFNumber Reference](#) in Core Foundation Reference Documentation for additional possible values.

*outNumberValue*

Before calling, *outNumberValue* must point to a number variable of the type and size you specified in *inNumberType*. On return, *outNumberValue* points to the retrieved number.

**Return Value**

A result code.

**Discussion**

This function is a convenience wrapper that calls [HIArchiveCopyDecodedCFTYPE](#) (page 6) to obtain a CFNumber value.

**Availability**

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

**Declared In**

HIArchive.h

## HIArchiveEncodeBoolean

Stores a Boolean value in an archive.

```
OSStatus HIArchiveEncodeBoolean (
    HIArchiveRef inEncoder,
    CFStringRef inKey,
    Boolean inBoolean
);
```

### Parameters

*inEncoder*

The archive to store the Boolean value.

*inKey*

A Core Foundation string key identifying the Boolean value.

*inBoolean*

The Boolean value.

### Return Value

A result code.

### Discussion

This function is a convenience wrapper that calls [HIArchiveEncodeCFTYPE](#) (page 10) with a CFBoolean value.

### Availability

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

### Declared In

HIArchive.h

## HIArchiveEncodeCFTYPE

Stores a CFTYPE object in an archive.

```
OSStatus HIArchiveEncodeCFTYPE (
    HIArchiveRef inEncoder,
    CFStringRef inKey,
    CFTYPERef inCFTYPE
);
```

### Parameters

*inEncoder*

The archive to store the CFTYPE object.

*inKey*

A Core Foundation string key identifying the CFTYPE object.

*inCFTYPE*

The CFTYPE object to store in the archive.

### Return Value

A result code.

**Discussion**

You can only encode base CType objects that correspond to archivable NSFoundation objects. For example, type CFStringRef is supported, but type HIShapeRef is not.

You also use this function for storing HIObjects and objects subclassed from HIObject. Currently only the following HIObject subclass types support archiving:

- HIObjectRef
- HIViewRef
- WindowRef
- ControlRef
- MenuRef

**Availability**

Available in Mac OS X v10.4 and later.  
 Not available to 64-bit applications.

**Declared In**

HIArchive.h

**HIArchiveEncodeNumber**

Stores a number in an archive.

```
OSStatus HIArchiveEncodeNumber (
    HIArchiveRef inEncoder,
    CFStringRef inKey,
    CFNumberType inNumberType,
    const void *inNumberValue
);
```

**Parameters**

*inEncoder*

The archive to store the number.

*inKey*

A Core Foundation string key identifying the number.

*inNumberType*

A CFNumber type identifying the type of number value to be stored, for example, kCFNumberSInt32Type. See CFNumber Reference in Core Foundation Reference Documentation for additional possible values.

*inNumberValue*

A pointer to the number value.

**Return Value**

A result code.

**Discussion**

This function is a convenience wrapper that calls [HIArchiveEncodeCType](#) (page 10) with a CFNumber value.

**Availability**

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

**Declared In**

HIArchive.h

**HIArchiveGetTypeID**

Obtains the CType ID for HIArchive objects.

```
CTypeID HIArchiveGetTypeID (  
    void  
);
```

**Return Value**

The Core Foundation type ID for the HIArchive object type.

**Availability**

Available in Mac OS X v10.4 and later.

Not available to 64-bit applications.

**Declared In**

HIArchive.h

## Data Types

**HIArchiveRef**

Defines an uncompressed archive object.

```
typedef struct OpaqueHIArchiveRef* HIArchiveRef;
```

**Discussion**

The structure pointed to by this reference is opaque. `HIArchiveRef` is a CType, and therefore responds to `CFRetain` and `CFRelease` calls.

**Availability**

Available in Mac OS X v10.4 and later.

**Declared In**

HIArchive.h

## Constants

**Archive Decoding Option Constant**

Defines options available when calling [HIArchiveCreateForDecoding](#) (page 7).

```
enum {
    kHIArchiveDecodeSuperclassForUnregisteredObjects = (1 << 0)
};
```

**Constants**

`kHIArchiveDecodeSuperclassForUnregisteredObjects`

If the class of the `HIObject` you are attempting to decode is not a registered subclass, this option allows [HIArchiveCopyDecodedCFTType](#) (page 6) to instantiate the object as its superclass, if it exists. For example, if your application has not yet registered `com.myCorp.mycustomView` before attempting to unarchive an instance of that `HIView`, `HIArchive` instantiates the data as class `com.apple.hiview`. Only data written to the superclass is decoded; any data unique to the unregistered subclass is ignored. Specifying this option also signals the `HIObject` to load its custom archive data so you can access it by calling `HIObjectCopyCustomArchiveData`.

This option can be useful when creating an archive editor that doesn't implement all the objects contained in a client archive.

Available in Mac OS X v10.4 and later.

Declared in `HIArchive.h`.

## Result Codes

Result Code	Value	Description
<code>noErr</code>	0	No error. Available in Mac OS X v10.0 and later.
<code>hiArchiveTypeMismatchErr</code>	-6780	The encoding or decoding archive was passed into a noncorresponding function. (For example, an archive created for encoding was passed into a decoding function.) Available in Mac OS X v10.4 and later.
<code>hiArchiveKeyNotAvailableErr</code>	-6781	The requested key does not exist in the specified archive. Available in Mac OS X v10.4 and later.
<code>hiArchiveEncodingCompleteErr</code>	-6782	<a href="#">HIArchiveCopyEncodedData</a> (page 7) was called on this archive, so no more data can be added. Available in Mac OS X v10.4 and later.
<code>hiArchiveHIObjectIgnoresArchivingErr</code>	-6783	The <code>HIObject</code> you wanted to encode does not support the <code>HIArchive</code> protocol. Available in Mac OS X v10.4 and later.



# Document Revision History

---

This table describes the changes to *HIArchive Reference*.

Date	Notes
2005-08-11	Moved conceptual information into the HIArchive Programming Guide.
2005-04-29	New document that describes the HIArchive APIs.

**REVISION HISTORY**

Document Revision History



# Index

---

## A

---

Archive Decoding Option Constant [12](#)

## H

---

HIArchiveCopyDecodedCType **function** [6](#)

HIArchiveCopyEncodedData **function** [7](#)

HIArchiveCreateForDecoding **function** [7](#)

HIArchiveCreateForEncoding **function** [8](#)

HIArchiveDecodeBoolean **function** [8](#)

HIArchiveDecodeNumber **function** [9](#)

HIArchiveEncodeBoolean **function** [10](#)

HIArchiveEncodeCType **function** [10](#)

HIArchiveEncodeNumber **function** [11](#)

hiArchiveEncodingCompleteErr **constant** [13](#)

HIArchiveGetTypeID **function** [12](#)

hiArchiveHIObjectIgnoresArchivingErr **constant**  
[13](#)

hiArchiveKeyNotAvailableErr **constant** [13](#)

HIArchiveRef **data type** [12](#)

hiArchiveTypeMismatchErr **constant** [13](#)

## K

---

kHIArchiveDecodeSuperclassForUnregisteredObjects  
**constant** [13](#)

## N

---

noErr **constant** [13](#)