Keychain Manager Reference

Security > Authentication

2005-07-07

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

Simultaneously published in the United States and Canada.

Contents

Keychain Manager Reference

Framework:	CoreServices/CoreServices.h, Carbon/Carbon.h
Declared in	KeychainCore.h KeychainHI.h

Overview

The Keychain Manager is an API that provides a uniform way for your application to handle passwords for multiple users, multiple databases, or any situation in which a user must enter single or multiple passwords. You can use the Keychain Manager to provide secure storage for a user's passwords, cryptographic keys, and digital certificates.

This document, which describes KeychainLib 2.0, is relevant to you if your application needs to create and manage passwords and other secure data.

Important: Keychain Manager is being phased out and replaced by Keychain Services. Any new development should use Keychain Services. See Keychain Services Reference.

Carbon fully supports the Keychain Manager.

Functions by Task

Getting Information About the Keychain Manager

`KCGetKeychainManagerVersion` (page 46)
Determines the version of the Keychain Manager installed on the user's system.

Creating and Disposing of Keychain References

`KCMakeKCRefFromAlias` (page 51)
Creates a keychain reference from a keychain alias.

`KCMakeAliasFromKCRef` (page 50)
Creates an alias to a keychain reference.

`KCReleaseKeychain` (page 53)
Disposes of the memory associated with a keychain reference.

KCMakeKCRefFromFSSpec (page 89) Deprecated in Mac OS X v10.5
Creates a keychain reference from a file specification record.

Managing Keychains

KCCreateKeychain (page 26)
Creates an empty keychain.

kccreatekeychain (page 27)

KCSetDefaultKeychain (page 57)
Sets the default keychain.

KCGetDefaultKeychain (page 44)
Obtains the default keychain.

KCGetStatus (page 48)
Determines the permissions that are set in a keychain.

KCGetKeychainName (page 47)
Determines the name of a keychain.

kcgetkeychainname (page 47)

KCCountKeychains (page 25)
Determines the number of available keychains.

KCGetIndKeychain (page 44)
Obtains the reference to an indexed keychain.

Storing and Retrieving Passwords

KCAddAppleSharePassword (page 12)
Adds a new AppleShare server password to the default keychain.

kcaddapplesharepassword (page 14)

KCFindAppleSharePassword (page 28)
Finds the first AppleShare password in the default keychain that matches the specified parameters.

kcfindapplesharepassword (page 30)

KCAddInternetPassword (page 18)
Adds a new Internet server password to the default keychain.

kcaddinternetpassword (page 19)

KCAddInternetPasswordWithPath (page 20)
Adds a new Internet server password with a specified path to the default keychain.

kcaddinternetpasswordwithpath (page 21)

KCFindInternetPassword (page 34)
Finds the first Internet password in the default keychain that matches the specified parameters.
kcfindinternetpassword (page 36)

KCFindInternetPasswordWithPath (page 37)
Finds the first Internet password in the default keychain that matches the specified parameters, including path information.
kcfindinternetpasswordwithpath (page 39)

KCAddGenericPassword (page 16)
Adds a new generic password to the default keychain.
kcaddgenericpassword (page 17)

KCFindGenericPassword (page 32)
Finds the first generic password in the default keychain matching the specified parameters.
kcfindgenericpassword (page 34)

Creating and Disposing of Keychain Item References

KCNewItem (page 51)
Creates a reference to a keychain item.
KCReleaseItem (page 52)
Disposes of the memory occupied by a keychain item reference.

Manipulating Keychain Items

KCAddItem (page 22)
Adds a password or other keychain item to the default keychain.
KCDeleteItem (page 28)
Deletes a password or other keychain item from the default keychain.
KCUpdateItem (page 60)
Updates a password or other keychain item.
KCCopyItem (page 24)
Copies a password or other keychain item from one keychain to another.
KCGetKeychain (page 45)
Determines the location of a password or other keychain item.

Setting and Obtaining Keychain Item Data

KCSetAttribute (page 55)
Sets or edits keychain item data using a keychain item attribute structure.

`KCGetAttribute` (page 41)
Determines keychain item data using a keychain item attribute structure.

`KCSetData` (page 56)
Sets or edits keychain item data.

`KCGetData` (page 43)
Determines keychain item data.

Searching for Keychain Items

`KCFindFirstItem` (page 31)
Finds the first keychain item in a specified keychain that matches specified attributes.

`KCFindNextItem` (page 40)
Finds the next keychain item matching the previously specified search criteria.

`KCReleaseSearch` (page 54)
Disposes of the memory occupied by a search criteria reference.

Managing User Interaction

`KCLock` (page 49)
Locks a keychain.

`KCUnlock` (page 59)
Displays a dialog box that prompts the user for a password before unlocking a keychain.

`kcunlock` (page 60)

`KCChangeSettings` (page 23)
Displays a dialog box enabling the user to change the name, password, or settings of a keychain.

`KCSetInteractionAllowed` (page 58)
Enables or disables Keychain Manager functions that display a user interface.

`KCIsInteractionAllowed` (page 49)
Indicates whether Keychain Manager functions that display a user interaction will do so.

Registering Your Keychain Event Callback Function

`KCAddCallback` (page 15)
Registers your keychain event callback function.

`KCRemoveCallback` (page 54)
Unregisters your keychain event callback function.

Working With Your Keychain Manager Callback Function

`NewKCCallbackUPP` (page 61)
Creates a UPP to your keychain event callback.

`InvokeKCCallbackUPP` (page 11)
Invokes your keychain event callback.

`DisposeKCCallbackUPP` (page 11)
Disposes of a UPP to your keychain event callback.

Unsupported Functions

`KCChooseCertificate` (page 24)
Displays a list of certificates that the user can choose from.

`KCFindX509Certificates` (page 41)
Finds the certificates in a keychain that match specified search criteria.

Functions

DisposeKCCallbackUPP

Disposes of a UPP to your keychain event callback.

Not recommended

```
void DisposeKCCallbackUPP (
   KCCallbackUPP userUPP
);
```

Parameters

userUPP
A Universal Procedure Pointer (UPP) to your keychain event callback function.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85).

Discussion
When you are finished with a UPP to your keychain event callback function, you should dispose of it by calling the `DisposeKCCallbackUPP` function.

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

There is no replacement function available.

Declared In
`KeychainCore.h`

InvokeKCCallbackUPP

Invokes your keychain event callback.

Not recommended

```
OSStatus InvokeKCCallbackUPP (
   KCEvent keychainEvent,
   KCCallbackInfo *info,
   void *userContext,
   KCCallbackUPP userUPP
);
```

Parameters

keychainEvent

The keychain events you want your application to receive. See "Keychain Events Constants" (page 71) for a description of possible values. The Keychain Manager tests the bitmask you pass in the `eventMask` parameter of the function `KCAddCallback` (page 15) to determine which events to pass to your callback function. See "Keychain Events Mask" (page 73) for a description of this bitmask.

info

A pointer to a structure of type `KCCallbackInfo` (page 65) that provides information about the keychain event to your callback function. The Keychain Manager passes a pointer to this structure in the `info` parameter of your callback function.

userContext

A pointer to application-defined storage. The Keychain Manager passes this value in the `userContext` parameter of your callback function. Your application can use this to associate any particular call of the `InvokeKCCallbackUPP` function with any particular call of the keychain event callback function.

userUPP

A Universal Procedure Pointer to your keychain event callback function. For information on how to create a keychain event callback function, see `KCCallbackProcPtr` (page 62).

Return Value

A result code. See "Keychain Manager Result Codes" (page 85).

Discussion

You should not need to use the function `InvokeKCCallbackUPP`, as the system calls your `KCAddCallback` (page 15) callback function for you.

Availability

Available in CarbonLib 1.1 and later.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

There is no replacement function available.

Declared In

`KeychainCore.h`

KCAddAppleSharePassword

Adds a new AppleShare server password to the default keychain.

Not recommended

```
OSStatus KCAddAppleSharePassword (
   AFPServerSignature *serverSignature,
   StringPtr serverAddress,
   StringPtr serverName,
   StringPtr volumeName,
   StringPtr accountName,
   UInt32 passwordLength,
   const void *passwordData,
   KCItemRef *item
);
```

Parameters

serverSignature

A pointer to a 16-byte Apple File Protocol server signature block. Pass a value of type `AFPServerSignature` (page 63). Pass `NULL` to match any server signature. The Keychain Manager identifies the location for the password by the information passed in the `serverAddress` and `serverSignature` parameters. You must pass a valid value in at least one of these parameters.

serverAddress

A pointer to a Pascal string containing the server address, which may be specified as an AppleTalk zone name, a DNS domain name (in the format "xxx.yyy.zzz"), or an IP address (in the format "111.222.333.444"). The Keychain Manager identifies the location for the password by the information passed in the `serverAddress` and `serverSignature` parameters. You must pass a valid value in at least one of these parameters.

serverName

A pointer to a Pascal string containing the server name.

volumeName

A pointer to a Pascal string containing the volume name.

accountName

A pointer to a Pascal string containing the account name.

passwordLength

The length of the buffer pointed to by `passwordData`.

passwordData

A pointer to a buffer which will hold the returned password data. Before calling `KCAddAppleSharePassword`, allocate enough memory for the buffer to hold the data you want to store.

item

On return, a pointer to a reference to the added item. Pass `NULL` if you don't want to obtain this reference.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain could be found. The result code `errKCDuplicateItem` indicates that you tried to add a password that already exists in the keychain. The result code `errKCDataTooLarge` indicates that you tried to add more data than is allowed for a record of this type.

Discussion

The `KCAddAppleSharePassword` function adds a new AppleShare server password to the default keychain that is uniquely identified by the `serverName`, `volumeName`, and `accountName` parameters, and a location specified either by the `serverAddress` or `serverSignature` parameters. The `KCAddAppleSharePassword` function optionally returns a reference to the newly added item.

Most applications do not need to store AppleShare password data, as this is handled transparently by the AppleShare client software. To be compatible with the AppleShare client, you should store a fully-specified File Manager structure `AFPXVolMountInfo` as the password data.

The `KCAddAppleSharePassword` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the password is currently locked.

You can also call the function `kcaddapplesharepassword` to add an AppleShare server password to the default keychain. `kcaddapplesharepassword` requires that you pass a pointer to a C string instead of a pointer to a Pascal string for the `serverAddress`, `serverName`, `volumeName`, `accountName`, and `passwordData` parameters.

Version Notes
Available beginning with KeychainLib 1.0. In KeychainLib 1.0, the `kcaddapplesharepassword` function provides the same functionality as `KCAddAppleSharePassword`, except that it accepts C strings rather than Pascal strings as arguments. In KeychainLib 2.0, you should use `KCAddAppleSharePassword`, since `kcaddapplesharepassword` is provided for convenience only and may be removed from the header file at some point in the future.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

It is recommended that you use internet passwords instead of AppleShare passwords. Use the `SecKeychainAddInternetPassword` function in Keychain Services instead.

Declared In
`KeychainHI.h`

kcaddapplesharepassword

Not recommended

```
OSStatus kcaddapplesharepassword (
   AFPServerSignature *serverSignature,
   const char *serverAddress,
   const char *serverName,
   const char *volumeName,
   const char *accountName,
   UInt32 passwordLength,
   const void *passwordData,
   KCItemRef *item
);
```

Discussion
This function is available for convenience only and may be removed. Use the function `KCAddAppleSharePassword` (page 12) instead.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

It is recommended that you use internet passwords instead of AppleShare passwords. Use the `SecKeychainAddInternetPassword` function in Keychain Services instead.

Declared In
`KeychainHI.h`

KCAddCallback

Registers your keychain event callback function.

Not recommended

```
OSStatus KCAddCallback (
   KCCallbackUPP callbackProc,
   KCEventMask eventMask,
   void *userContext
);
```

Parameters

`callbackProc`

A Universal Procedure Pointer (UPP) to your keychain event callback function, described in `KCCallbackProcPtr` (page 62). You indicate the type of keychain events you want to receive by passing a bitmask of the desired events in the `eventMask` parameter. To create a UPP to your callback function, call the function `NewKCCallbackUPP` (page 61).

`eventMask`

A bitmask indicating the keychain events that your application wishes to be notified of. See "Keychain Events Mask" (page 73) for a description of this bitmask. The Keychain Manager tests this mask to determine the keychain events that you wish to receive, and passes these events in the `keychainEvent` parameter of your callback function. See "Keychain Events Constants" (page 71) for a description of these events.

`userContext`

A pointer to application-defined storage that will be passed to your callback function. Your application can use this to associate any particular call of the `KCAddCallback` function with any particular call of your keychain event callback function.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCDuplicateCallback` indicates that your callback function is already registered.

Discussion
You can register your callback function by passing a UPP to it in the `callbackProc` parameter of the `KCAddCallback` function. Once you have done so, the Keychain Manager calls the function `InvokeKCCallbackUPP` (page 11) when the keychain event specified in the `eventMask` parameter occurs. In turn, the function `InvokeKCCallbackUPP` (page 11) passes the keychain event, information about the event, and application-defined storage to your keychain event callback function.

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainAddCallback` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCAddGenericPassword

Adds a new generic password to the default keychain.

Not recommended

```
OSStatus KCAddGenericPassword (
   StringPtr serviceName,
   StringPtr accountName,
   UInt32 passwordLength,
   const void *passwordData,
   KCItemRef *item
);
```

Parameters

serviceName
: A pointer to a Pascal string containing an application-defined service name.

accountName
: A pointer to a Pascal string containing an application-defined account name.

passwordLength
: The length of the password data to be stored

passwordData
: A pointer to the buffer that holds the returned password data. Before calling the `KCAddGenericPassword` function, allocate enough memory for the buffer to hold the data you want to store.

item
: On return, a pointer to a reference to the added item. Pass `NULL` if you don't want to obtain this reference.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain could be found. The result code `errKCDuplicateItem` indicates that you tried to add a password that already exists in the keychain. The result code `errKCDataTooLarge` indicates that you tried to add more data than is allowed for a record of this type.

Discussion
The `KCAddGenericPassword` function adds a new generic password to the default keychain. Required parameters to identify the password are `serviceName` and `accountName`, which are application-defined strings. The `KCAddGenericPassword` function optionally returns a reference to the newly added item.

You can use the `KCAddGenericPassword` function to add passwords for accounts other than Internet or AppleShare. For example, you might add passwords for your database or scheduling programs.

You can also call the function `kcaddgenericpassword` to add a new generic password to the default keychain. The difference between the two functions is that the `kcaddgenericpassword` function takes a pointer to a C string instead of a Pascal string in the `serverAddress`, `serverName`, `volumeName`, `accountName`, and `passwordData` parameters.

The `KCAddGenericPassword` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the password is currently locked.

Version Notes
Available beginning with KeychainLib 1.0. In KeychainLib 1.0, the `kcaddgenericpassword` function provides the same functionality as the function `KCAddGenericPassword`, except that it accepts C strings rather than Pascal strings as arguments. In KeychainLib 2.0, you should use the `KCAddGenericPassword` function, since the `kcaddgenericpassword` function is provided for convenience only and may be removed from the header file at some point in the future.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainAddGenericPassword` function in Keychain Services instead.

Declared In
`KeychainHI.h`

kcaddgenericpassword

Not recommended

```
OSStatus kcaddgenericpassword (
   const char *serviceName,
   const char *accountName,
   UInt32 passwordLength,
   const void *passwordData,
   KCItemRef *item
);
```

Discussion
This function is available for convenience only and may be removed. Use the function `KCAddGenericPassword` (page 16) instead.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainAddGenericPassword` function in Keychain Services instead.

Declared In
`KeychainHI.h`

KCAddInternetPassword

Adds a new Internet server password to the default keychain.

Not recommended

```
OSStatus KCAddInternetPassword (
   StringPtr serverName,
   StringPtr securityDomain,
   StringPtr accountName,
   UInt16 port,
   OSType protocol,
   OSType authType,
   UInt32 passwordLength,
   const void *passwordData,
   KCItemRef *item
);
```

Parameters

serverName

A pointer to a Pascal string containing the server name.

securityDomain

A pointer to a Pascal string containing the security domain. This parameter is optional, as not all protocols will require it.

accountName

A pointer to a Pascal string containing the account name.

port

The TCP/IP port number. Pass the constant `kAnyPort`, described in "Default Internet Port Constant" (page 71), to specify any port.

protocol

The protocol associated with this password. See "Keychain Protocol Type Constants" (page 82) for a description of possible values. Pass the constant `kAnyProtocol`, described in "Default Internet Protocol And Authentication Type Constants" (page 71), to specify any protocol.

authType

The authentication scheme used. See "Authentication Type Constants" (page 67) for a description of possible values. Pass the constant `kAnyAuthType`, described in "Default Internet Protocol And Authentication Type Constants" (page 71), to specify any authentication scheme.

passwordLength

The length of the buffer pointed to by `passwordData`.

passwordData

A pointer to a buffer that holds the returned password data. Before calling the `KCAddInternetPasswordWithPath` function, allocate enough memory for the buffer to hold the data you want to store.

item

On return, a pointer to a reference to the added item. Pass `NULL` if you don't want to obtain this reference.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain could be found. The result code `errKCDuplicateItem` indicates that you tried to add a password that already exists in the keychain. The result code `errKCDataTooLarge` indicates that you tried to add more data than is allowed for a record of this type.

Discussion
The `KCAddInternetPassword` function adds a new Internet server password to the default keychain. Required parameters to identify the password are `serviceName` and `accountName` (you cannot pass `NULL` for both parameters). In addition, some protocols may require an optional value in the `securityDomain` parameter when authentication is requested. `KCAddInternetPassword` optionally returns a reference to the newly added item.

The `KCAddInternetPassword` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the password is currently locked.

You can also call the function `kcaddinternetpassword` to add a new Internet server password to the default keychain. The `kcaddinternetpassword` function requires that you pass a pointer to a C string instead of a pointer to a Pascal string for the `serverAddress`, `serverName`, `volumeName`, `accountName`, and `passwordData` parameters.

Version Notes
Available beginning with KeychainLib 1.0. In KeychainLib 1.0, the `kcaddinternetpassword` function provides the same functionality as `KCAddInternetPassword`, except that it accepts C strings rather than Pascal strings as arguments. In KeychainLib 2.0, you should use `KCAddInternetPassword`, since `kcaddinternetpassword` is provided for convenience only and may be removed from the header file at some point in the future.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainAddInternetPassword` function in Keychain Services instead.

Declared In
`KeychainHI.h`

kcaddinternetpassword

Not recommended

```
OSStatus kcaddinternetpassword (
   const char *serverName,
   const char *securityDomain,
   const char *accountName,
   UInt16 port,
   OSType protocol,
   OSType authType,
   UInt32 passwordLength,
   const void *passwordData,
   KCItemRef *item
);
```

Discussion
This function is available for convenience only and may be removed. Use the function `KCAddInternetPassword` (page 18) instead.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainAddInternetPassword` function in Keychain Services instead.

Declared In

`KeychainHI.h`

KCAddInternetPasswordWithPath

Adds a new Internet server password with a specified path to the default keychain.

Not recommended

```
OSStatus KCAddInternetPasswordWithPath (
   StringPtr serverName,
   StringPtr securityDomain,
   StringPtr accountName,
   StringPtr path,
   UInt16 port,
   OSType protocol,
   OSType authType,
   UInt32 passwordLength,
   const void *passwordData,
   KCItemRef *item
);
```

Parameters

serverName

A pointer to a Pascal string containing the server name.

securityDomain

A pointer to a Pascal string containing the security domain. This parameter is optional, as not all protocols will require it.

accountName

A pointer to a Pascal string containing the account name.

path

A pointer to a Pascal string containing additional information that specifies a file or directory on the server specified by the `serverName` parameter. In a typical URL, path information begins directly after the first slash ("/") character following the server name. This parameter is optional.

port

The TCP/IP port number. Pass the constant `kAnyPort`, described in "Default Internet Port Constant" (page 71), to specify any port.

protocol

The protocol associated with this password. See "Keychain Protocol Type Constants" (page 82) for a description of possible values. Pass the constant `kAnyProtocol`, described in "Default Internet Protocol And Authentication Type Constants" (page 71), to specify any protocol.

authType

The authentication scheme used. See "Authentication Type Constants" (page 67) for a description of possible values. Pass the constant `kAnyAuthType`, described in "Default Internet Protocol And Authentication Type Constants" (page 71), to specify any authentication scheme.

passwordLength

The length of the buffer pointed to by `passwordData`.

passwordData

A pointer to a buffer which will hold the returned password data. Before calling `KCAddInternetPasswordWithPath`, allocate enough memory for the buffer to hold the data you want to store.

item

On return, a pointer to a reference to the added item. Pass `NULL` if you don't want to obtain this reference.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain could be found. The result code `errKCDuplicateItem` indicates that you tried to add a password that already exists in the keychain. The result code `errKCDataTooLarge` indicates that you tried to add more data than is allowed for a record of this type.

Discussion

The `KCAddInternetPasswordWithPath` function enables you to specify path information when adding a new Internet server password to the default keychain. Required parameters to identify the password are `serviceName` and `accountName` (you cannot pass `NULL` for both parameters). In addition, some protocols may require an optional `securityDomain` when authentication is requested. `KCAddInternetPasswordWithPath` optionally returns a reference to the newly added item.

The `KCAddInternetPasswordWithPath` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the password is currently locked.

You can also call the function `kcaddinternetpasswordwithpath` to add a new Internet server password to the default keychain. The function `kcaddinternetpasswordwithpath` requires that you pass a pointer to a C string instead of a pointer to a Pascal string for the `serverAddress`, `serverName`, `volumeName`, `accountName`, and `passwordData` parameters.

Version Notes

Available beginning with KeychainLib 2.0. In KeychainLib 1.0, the `kcaddinternetpasswordwithpath` function provides the same functionality as the `KCAddInternetPasswordWithPath` function, except that it accepts C strings rather than Pascal strings as arguments. In KeychainLib 2.0, you should use the `KCAddInternetPasswordWithPath` function, since the function `kcaddinternetpasswordwithpath` is provided for convenience only and may be removed from the header file at some point in the future.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainAddInternetPassword` function in Keychain Services instead.

Declared In

`KeychainHI.h`

kcaddinternetpasswordwithpath

Not recommended

```
OSStatus kcaddinternetpasswordwithpath (
   const char *serverName,
   const char *securityDomain,
   const char *accountName,
   const char *path,
   UInt16 port,
   OSType protocol,
   OSType authType,
   UInt32 passwordLength,
   const void *passwordData,
   KCItemRef *item
);
```

Discussion
This function is available for convenience only and may be removed. Use the function `KCAddInternetPasswordWithPath` (page 20) instead.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainAddInternetPassword` function in Keychain Services instead.

Declared In
`KeychainHI.h`

KCAddItem

Adds a password or other keychain item to the default keychain.

Not recommended

```
OSStatus KCAddItem (
   KCItemRef item
);
```

Parameters

item

A reference to the keychain item you wish to add. If you pass an existing item in the keychain, the item is updated. If you pass an item that has not been previously added to the keychain and an identical item already exists in the keychain, `KCAddItem` returns the result code `errKCDuplicateItem`.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain could be found. The result code `errKCInvalidItemRef` indicates that the specified keychain item reference was invalid. The result code `errKCDuplicateItem` indicates that you tried to add a new item that already exists in the keychain.

Discussion
You can use the `KCAddItem` function to add a password or other keychain item to the permanent data store of the default keychain. If you want to add a password to a keychain other than the default, call the function `KCSetDefaultKeychain` (page 57) to change the default keychain. The `KCAddItem` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the item is currently locked.

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

When you use Keychain Services to create an item, it is always added to the specified keychain at creation time.

Declared In
`KeychainHI.h`

KCChangeSettings

Displays a dialog box enabling the user to change the name, password, or settings of a keychain.

Not recommended

```
OSStatus KCChangeSettings (
   KCRef keychain
);
```

Parameters

`keychain`
A reference to an unlocked keychain. Pass in `NULL` to specify the default keychain.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errUserCanceled` indicates that the user pressed the Cancel button in the Change Settings dialog box. The result code `errKCNoDefaultKeychain` indicates that the default keychain could not be found. The result code `errKCInvalidKeychain` indicates that the specified keychain is invalid.

Discussion
Typically, your application should not call the `KCChangeSettings` function. You would only call the `KCChangeSettings` function in response to a user's request to change keychain settings, name, or password. Note that you cannot change a keychain passphrase directly. You must call the `KCChangeSettings` function and allow the user to change it.

Version Notes
Available beginning with KeychainLib 2.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainSetSettings` function in Keychain Services instead.

Declared In
`KeychainHI.h`

KCChooseCertificate

Displays a list of certificates that the user can choose from.

Unsupported

```
OSStatus KCChooseCertificate (
    CFArrayRef items,
    KCItemRef *certificate,
    CFArrayRef policyOIDs,
    KCVerifyStopOn stopOn
);
```

Parameters

items

An array of certificate references.

certificate

If the `items` array only contains one certificate, on return, a pointer to that certificate. In this case, no user interface is displayed.

policyOIDs

An array of trust policy options used for Macintosh file signing. To obtain a pointer to this array, call the function `SecMacGetDefaultPolicyOIDs`.

stopOn

The criteria to use in selecting the certificates to display. See "Certificate Verification Criteria" (page 70) for a description of this mask.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `userCanceledErr` indicates that the user pressed the Cancel button in the user interface.

Discussion

The `KCChooseCertificate` function displays a list of the certificates from which the user can choose. If only one certificate matches the criteria, the reference is passed back in the `certificate` parameter and no user interface is presented.

Version Notes

Available beginning with KeychainLib 2.0.

Carbon Porting Notes

This function is obsolete. There is currently no replacement.

Declared In

`KeychainHI.h`

KCCopyItem

Copies a password or other keychain item from one keychain to another.

Not recommended

```
OSStatus KCCopyItem (
   KCItemRef item,
   KCRef destKeychain,
   KCItemRef *copy
);
```

Parameters

item

A reference to the keychain item you wish to copy.

destKeychain

A reference to the keychain into which the item is to be copied.

copy

A pointer to a reference to the new copied keychain item.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCReadOnly` indicates that the destination keychain is read only. The result code `errKCNoSuchClass` indicates that the item has an invalid keychain item class. The result code `errKCInvalidItemRef` indicates that the specified keychain item reference was invalid.

Discussion

You can use the `KCCopyItem` function to copy a keychain item from one keychain to another. The `KCCopyItem` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the item to be copied is currently locked.

Version Notes

Available beginning with KeychainLib 2.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemCopyContent` function in Keychain Services instead.

Declared In

`KeychainCore.h`

KCCountKeychains

Determines the number of available keychains.

Not recommended

```
UInt16 KCCountKeychains (
   void
);
```

Parameters

Return Value

The number of available keychains. This includes all keychains in the Keychains folder, as well as any other keychains known to the Keychain Manager.

Discussion
This function reports the number of keychains known to the Keychain Manager. These keychains are created by the function `KCCreateKeychain` (page 26).

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainCopySearchList` function in Keychain Services followed by a call to `CFArrayGetCount` instead.

Declared In
`KeychainCore.h`

KCCreateKeychain

Creates an empty keychain.

Not recommended

```
OSStatus KCCreateKeychain (
   StringPtr password,
   KCRef *keychain
);
```

Parameters

password

A pointer to a Pascal string representing the password string which will be used to protect the new keychain. If you pass `NULL`, the Keychain Setup dialog box will be displayed to obtain it.

keychain

A pointer to a reference to the keychain you wish to create. You create a keychain reference by calling the function `KCMakeKCRefFromFSSpec` (page 89) or `KCMakeKCRefFromAlias` (page 51). If you pass a pointer to a keychain reference, the user will not need to be prompted for a name and location; in all other cases, `KCCreateKeychain` will interactively request this information from the user. If you pass a pointer to a `NULL` keychain reference, the Keychain Manager allocates the memory for the keychain reference and returns it in this parameter. Pass a `NULL` pointer if you do not need a reference returned.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `userCanceledErr` indicates that the user pressed the Cancel button in the create keychain. The result code `errKCDuplicateKeychain` indicates that the user tried to create a keychain which already exists. The result code `errKCInvalidKeychain` indicates that the specified keychain is invalid. Additional errors may be returned if the keychain could not be created (for example, a file system or network error may be returned if there is no write access to the storage media).

Discussion
The `KCCreateKeychain` function creates an empty keychain. The `keychain` and `password` parameters are optional. If user interaction to create a keychain is posted, the newly-created keychain is automatically unlocked after creation.

You can also call the function `kccreatekeychain` to create an empty keychain. The function `kccreatekeychain` requires that you pass a pointer to a C string instead of a pointer to a Pascal string in the `password` parameter.

Special Considerations

It is recommended that the `KCCreateKeychain` function not be explicitly called by applications. Instead, you should call one of the add functions in "Storing and Retrieving Passwords" (page 8) to add a password to the default keychain. If a default keychain does not exist, it is created automatically.

When you are finished with a keychain, you must deallocate its memory by calling the function `KCReleaseKeychain` (page 53).

Version Notes

Available beginning with KeychainLib 1.0. In KeychainLib 1.0, the `kccreatekeychain` function provides the same functionality as `KCCreateKeychain`. In KeychainLib 2.0, you should use `KCCreateKeychain`, since `kccreatekeychain` is provided for convenience only and may be removed from the header file at some point in the future.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainCreate` function in Keychain Services instead.

Declared In

`KeychainHI.h`

kccreatekeychain

Not recommended

```
OSStatus kccreatekeychain (
   const char *password,
   KCRef *keychain
);
```

Discussion

This function is available for convenience only and may be removed. Use the function `KCCreateKeychain` (page 26) instead.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainCreate` function in Keychain Services instead.

Declared In

`KeychainHI.h`

KCDeleteItem

Deletes a password or other keychain item from the default keychain.

Not recommended

```
OSStatus KCDeleteItem (
   KCItemRef item
);
```

Parameters

item

A reference to the keychain item you wish to delete. If you pass an item that has not been previously added to the keychain, the function `KCDeleteItem` does nothing and returns `noErr`.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain could be found. The result code `errKCInvalidItemRef` indicates that the specified keychain item reference was invalid.

Discussion

You can use the `KCDeleteItem` function to delete a keychain item from the permanent data store of the default keychain. The `KCDeleteItem` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the item is currently locked.

Special Considerations

The `KCDeleteItem` function does not dispose the memory occupied by the item reference. To do so, call the function `KCReleaseItem` (page 52) when you are finished with an item.

Version Notes

Available beginning with KeychainLib 1.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemDelete` function in Keychain Services instead.

Declared In

`KeychainCore.h`

KCFindAppleSharePassword

Finds the first AppleShare password in the default keychain that matches the specified parameters.

Not recommended

```
OSStatus KCFindAppleSharePassword (
   AFPServerSignature *serverSignature,
   ConstStringPtr serverAddress,
   ConstStringPtr serverName,
   ConstStringPtr volumeName,
   ConstStringPtr accountName,
   UInt32 maxLength,
   void *passwordData,
   UInt32 *actualLength,
   KCItemRef *item
);
```

Parameters

serverSignature

A pointer to a 16-byte Apple File Protocol server signature block. Pass a value of type AFPServerSignature (page 63). Pass NULL to match any server signature. The Keychain Manager identifies the location for the password by the information passed in the serverAddress and serverSignature parameters. You must pass a valid value in at least one of these parameters.

serverAddress

A pointer to a Pascal string containing the server address, which may be specified as an AppleTalk zone name, a DNS domain name (in the format "xxx.yyy.zzz"), or an IP address (in the format "111.222.333.444"). The Keychain Manager identifies the location for the password by the information passed in the serverAddress and serverSignature parameters. You must pass a valid value in at least one of these parameters.

serverName

A pointer to a Pascal string containing the server name. Pass NULL to match any server name.

volumeName

A pointer to a Pascal string containing the volume name. Pass NULL to match any volume name.

accountName

A pointer to a Pascal string containing the account name. Pass NULL to match any account name.

maxLength

The length of the buffer pointed to by passwordData.

passwordData

A pointer to a buffer which will hold the returned password data. Before calling KCFindAppleSharePassword, allocate enough memory for the buffer to hold the data you want to store. Pass NULL if you want to obtain the item reference but not the password data. In this case, you must also pass NULL in the actualLength parameter. On return, a pointer to the returned password data.

actualLength

On return, the actual length of the password data that was retrieved. If the buffer pointed to by passwordData is smaller than the actual length of the data, KCFindAppleSharePassword returns the result code errKCBufferTooSmall. In this case, your application must allocate a new buffer of sufficient size before calling KCFindAppleSharePassword again.

item

On return, a pointer to a reference to the found item. Pass NULL if you don't want to obtain this reference.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain was found. The result code `errKCItemNotFound` indicates that no matching password item was found. The result code `errKCBufferTooSmall` indicates that your application must allocate a new buffer of sufficient size before calling `KCFindAppleSharePassword` again.

Discussion
The `KCFindAppleSharePassword` function finds the first AppleShare password item which matches the attributes you provide. The buffer specified in the `passwordData` parameter must be large enough to hold the password data, otherwise `KCFindAppleSharePassword` returns the result code `errKCBufferTooSmall`. In this case, your application must allocate a new buffer of sufficient size before calling the `KCFindAppleSharePassword` function again. The `KCFindAppleSharePassword` function optionally returns a reference to the found item.

The `KCFindAppleSharePassword` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the password is currently locked.

You can also call the function `kcfindapplesharepassword` to find the first AppleShare server password matching specified attributes. `kcfindapplesharepassword` requires that you pass a pointer to a C string instead of a pointer to a Pascal string for the `serverAddress`, `serverName`, `volumeName`, `accountName`, and `passwordData` parameters.

Version Notes
Available beginning with KeychainLib 1.0. In KeychainLib 1.0, the `kcfindapplesharepassword` function provides the same functionality as `KCFindAppleSharePassword`, except that it accepts C strings rather than Pascal strings as arguments. In KeychainLib 2.0, you should use `KCFindAppleSharePassword`, since `kcfindapplesharepassword` is provided for convenience only and may be removed from the header file at some point in the future.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the Keychain Services function `SecKeychainSearchCreateFromAttributes` followed by the `SecKeychainSearchCopyNext` function instead.

Declared In
`KeychainCore.h`

kcfindapplesharepassword
Not recommended

```
OSStatus kcfindapplesharepassword (
   AFPServerSignature *serverSignature,
   const char *serverAddress,
   const char *serverName,
   const char *volumeName,
   const char *accountName,
   UInt32 maxLength,
   void *passwordData,
   UInt32 *actualLength,
   KCItemRef *item
);
```

Discussion
This function is available for convenience only and may be removed. Use the function `KCFindAppleSharePassword` (page 28) instead.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the Keychain Services function `SecKeychainSearchCreateFromAttributes` followed by the `SecKeychainSearchCopyNext` function instead.

Declared In
`KeychainCore.h`

KCFindFirstItem

Finds the first keychain item in a specified keychain that matches specified attributes.

Not recommended

```
OSStatus KCFindFirstItem (
   KCRef keychain,
   const KCAttributeList *attrList,
   KCSearchRef *search,
   KCItemRef *item
);
```

Parameters

keychain
: A reference to the keychain that you wish to search. If you pass a locked keychain, the Unlock Keychain dialog box is displayed. If you pass `NULL`, the `KCFindFirstItem` function searches all unlocked keychains.

attrList
: A pointer to a list of 0 or more structures containing information about the keychain item attributes to be matched. Pass `NULL` to match any attribute.

search
: On return, a pointer to a reference to the current search criteria.

item
: On return, a pointer to the first matching keychain item.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain could be found. The result code `errKCItemNotFound` indicates that no matching keychain item was found. The result code `errKCNoSuchAttr` indicates that the specified attribute is undefined for this item class.

Discussion
The `KCFindFirstItem` function returns a reference to the first keychain item in a keychain that matches a list of attributes. The `KCFindFirstItem` function also returns a reference to the search criteria used. You should pass the returned search criteria in the `searchRef` parameter of the function `KCFindNextItem` (page 40).

The `KCFindFirstItem` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the item you are searching for is currently locked.

Special Considerations

When you are completely finished with a search, you should the functions `KCReleaseItem` (page 52) and `KCReleaseSearch` (page 54) to release the memory occupied by the keychain item and search criteria reference.

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the Keychain Services function `SecKeychainSearchCreateFromAttributes` followed by a call to `SecKeychainSearchCopyNext` instead.

Declared In
`KeychainCore.h`

KCFindGenericPassword

Finds the first generic password in the default keychain matching the specified parameters.

Not recommended

```
OSStatus KCFindGenericPassword (
   ConstStringPtr serviceName,
   ConstStringPtr accountName,
   UInt32 maxLength,
   void *passwordData,
   UInt32 *actualLength,
   KCItemRef *item
);
```

Parameters

`serviceName`

A pointer to a Pascal string containing an application-defined service name. Pass `NULL` to match any service name.

accountName

A pointer to a Pascal string containing an application-defined account name. Pass `NULL` to match any account name.

maxLength

The length of the buffer pointed to by `passwordData`.

passwordData

A pointer to the buffer that holds the returned password data. Before calling the `KCFindGenericPassword` function, allocate enough memory for the buffer to hold the data you want to store. Pass `NULL` if you want to obtain the item reference but not the password data. In this case, you must also pass `NULL` in the `actualLength` parameter. On return, a pointer to the returned password data.

actualLength

On return, the actual length of the password data that was retrieved. If the buffer pointed to by the `passwordData` parameter is smaller than the actual length of the data, the `KCFindGenericPassword` function returns the result code `errKCBufferTooSmall`. In this case, your application must allocate a new buffer of sufficient size before calling the `KCFindGenericPassword` function again.

item

On return, a pointer to a reference to the found item. Pass `NULL` if you don't want to obtain this reference.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain was found. The result code `errKCItemNotFound` indicates that no matching password item was found. The result code `errKCBufferTooSmall` indicates that your application must allocate a new buffer of sufficient size before calling the function `KCFindGenericPassword` again.

Discussion

The `KCFindGenericPassword` function finds the first generic password item which matches the attributes you provide. The buffer specified in the `passwordData` parameter must be large enough to hold the password data, otherwise the function `KCFindGenericPassword` returns the result code `errKCBufferTooSmall`. In this case, your application must allocate a new buffer of sufficient size before calling the function `KCFindGenericPassword` again. The `KCFindGenericPassword` function optionally returns a reference to the found item.

The `KCFindGenericPassword` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the password is currently locked.

You can also call the function `kcfindgenericpassword` to find the first generic password matching specified attributes. The `kcfindgenericpassword` function requires that you pass a pointer to a C string instead of a pointer to a Pascal string for the `serverAddress`, `serverName`, `volumeName`, `accountName`, and `passwordData` parameters.

Version Notes

Available beginning with KeychainLib 1.0. In KeychainLib 1.0, the `kcfindgenericpassword` function provides the same functionality as the function `KCFindGenericPassword`, except that it accepts C strings rather than Pascal strings as arguments. In KeychainLib 2.0, you should use the `KCFindGenericPassword` function, since the `kcfindgenericpassword` function is provided for convenience only and may be removed from the header file at some point in the future.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainFindGenericPassword` function in Keychain Services instead.

Declared In

`KeychainCore.h`

kcfindgenericpassword

Not recommended

```
OSStatus kcfindgenericpassword (
   const char *serviceName,
   const char *accountName,
   UInt32 maxLength,
   void *passwordData,
   UInt32 *actualLength,
   KCItemRef *item
);
```

Discussion

This function is available for convenience only and may be removed. Use the function `KCFindGenericPassword` (page 32) instead.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainFindGenericPassword` function in Keychain Services instead.

Declared In

`KeychainCore.h`

KCFindInternetPassword

Finds the first Internet password in the default keychain that matches the specified parameters.

Not recommended

```
OSStatus KCFindInternetPassword (
   ConstStringPtr serverName,
   ConstStringPtr securityDomain,
   ConstStringPtr accountName,
   UInt16 port,
   OSType protocol,
   OSType authType,
   UInt32 maxLength,
   void *passwordData,
   UInt32 *actualLength,
   KCItemRef *item
);
```

Parameters

serverName

A pointer to a Pascal string containing the server name. Pass `NULL` to match any server name.

securityDomain

A pointer to a Pascal string containing the security domain. Pass `NULL` to match any domain.

accountName

A pointer to a Pascal string containing the account name. Pass `NULL` to match any account name.

port

The TCP/IP port number. Pass the constant `kAnyPort`, described in "Default Internet Port Constant" (page 71), to match any port.

protocol

The protocol associated with this password. See "Keychain Protocol Type Constants" (page 82) for a description of possible values. Pass the constant `kAnyProtocol`, described in "Default Internet Protocol And Authentication Type Constants" (page 71), to match any protocol.

authType

The authentication scheme used. See "Authentication Type Constants" (page 67) for a description of possible values. Pass the constant `kAnyAuthType`, described in "Default Internet Protocol And Authentication Type Constants" (page 71), to match any authentication scheme.

maxLength

The length of the buffer pointed to by `passwordData`.

passwordData

A pointer to the buffer that holds the returned password data. Before calling the `KCFindInternetPassword` function, allocate enough memory for the buffer to hold the data you want to store. Pass `NULL` if you want to obtain the item reference but not the password data. In this case, you must also pass `NULL` in the `actualLength` parameter. On return, a pointer to the returned password data.

actualLength

On return, the actual length of the password data that was retrieved. If the buffer pointed to by the `passwordData` parameter is smaller than the actual length of the data, the `KCFindInternetPassword` function returns the result code `errKCBufferTooSmall`. In this case, your application must allocate a new buffer of sufficient size before calling the `KCFindInternetPassword` function again.

item

On return, a pointer to a reference to the found item. Pass `NULL` if you don't want to obtain this reference.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain was found. The result code `errKCItemNotFound` indicates that no matching password item was found. The result code `errKCBufferTooSmall` indicates that your application must allocate a new buffer of sufficient size before calling the function `KCFindInternetPassword` again.

Discussion
The `KCFindInternetPassword` function finds the first Internet password item that matches the attributes you provide. The buffer specified in the `passwordData` parameter must be large enough to hold the password data, otherwise the function `KCFindInternetPassword` returns the result code `errKCBufferTooSmall`. In this case, your application must allocate a new buffer of sufficient size before calling the function `KCFindInternetPassword` again. The `KCFindInternetPassword` function optionally returns a reference to the found item.

The `KCFindInternetPassword` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the password is currently locked.

You can also call the function `kcfindinternetpassword` to find the first Internet password item matching specified attributes. The `kcfindinternetpassword` function requires that you pass a pointer to a C string instead of a Pascal string for the `serverAddress`, `serverName`, `volumeName`, `accountName`, and `passwordData` parameters.

Version Notes
Available beginning with KeychainLib 1.0. In KeychainLib 1.0, the `kcfindinternetpassword` function provides the same functionality as the function `KCFindInternetPassword`, except that it accepts C strings rather than Pascal strings as arguments. In KeychainLib 2.0, you should use the `KCFindInternetPassword` function, since `kcfindinternetpassword` is provided for convenience only and may be removed from the header file at some point in the future.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes
Use the `SecKeychainFindInternetPassword` function in Keychain Services instead.

Declared In
`KeychainCore.h`

kcfindinternetpassword
Not recommended

```
OSStatus kcfindinternetpassword (
   const char *serverName,
   const char *securityDomain,
   const char *accountName,
   UInt16 port,
   OSType protocol,
   OSType authType,
   UInt32 maxLength,
   void *passwordData,
   UInt32 *actualLength,
   KCItemRef *item
);
```

Discussion
This function is available for convenience only and may be removed. Use the function `KCFindInternetPassword` (page 34) instead.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainFindInternetPassword` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCFindInternetPasswordWithPath

Finds the first Internet password in the default keychain that matches the specified parameters, including path information.

Not recommended

```
OSStatus KCFindInternetPasswordWithPath (
   ConstStringPtr serverName,
   ConstStringPtr securityDomain,
   ConstStringPtr accountName,
   ConstStringPtr path,
   UInt16 port,
   OSType protocol,
   OSType authType,
   UInt32 maxLength,
   void *passwordData,
   UInt32 *actualLength,
   KCItemRef *item
);
```

Parameters

serverName
A pointer to a Pascal string containing the server name. Pass `NULL` to match any server name.

securityDomain
A pointer to a Pascal string containing the security domain. Pass `NULL` to match any domain.

accountName

A pointer to a Pascal string containing the account name. Pass `NULL` to match any account name.

path

A pointer to a Pascal string containing additional information that specifies a file or directory on the server specified by the `serverName` parameter. In a typical URL, path information begins directly after the first slash ("/") character following the server name. This parameter is optional.

port

The TCP/IP port number. Pass the constant `kAnyPort`, described in "Default Internet Port Constant" (page 71), to match any port.

protocol

The protocol associated with this password. See "Keychain Protocol Type Constants" (page 82) for a description of possible values. Pass the constant `kAnyProtocol`, described in "Default Internet Protocol And Authentication Type Constants" (page 71), to match any protocol.

authType

The authentication scheme used. See "Authentication Type Constants" (page 67) for a description of possible values. Pass the constant `kAnyAuthType`, described in "Default Internet Protocol And Authentication Type Constants" (page 71), to match any authentication scheme.

maxLength

The length of the buffer pointed to by `passwordData`.

passwordData

A pointer to a buffer which will hold the returned password data. Before calling the `KCFindInternetPasswordWithPath` function, allocate enough memory for the buffer to hold the data you want to store. Pass `NULL` if you want to obtain the item reference but not the password data. In this case, you must also pass `NULL` in the `actualLength` parameter. On return, a pointer to the returned password data.

actualLength

On return, the actual length of the password data that was retrieved. If the buffer pointed to by the `passwordData` parameter is smaller than the actual length of the data, the function `KCFindInternetPasswordWithPath` returns the result code `errKCBufferTooSmall`. In this case, your application must allocate a new buffer of sufficient size before calling the function `KCFindInternetPasswordWithPath` again.

item

On return, a pointer to a reference to the found item. Pass `NULL` if you don't want to obtain this reference.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain was found. The result code `errKCItemNotFound` indicates that no matching password item was found. The result code `errKCBufferTooSmall` indicates that your application must allocate a new buffer of sufficient size before calling the function `KCFindInternetPasswordWithPath` again.

Discussion

The `KCFindInternetPasswordWithPath` function finds the first Internet password item which matches the attributes you provide, including path information. The buffer specified in the `passwordData` parameter must be large enough to hold the password data, otherwise the function `KCFindInternetPasswordWithPath` returns the result code `errKCBufferTooSmall`. In this case, your application must allocate a new buffer of sufficient size before calling the `KCFindInternetPasswordWithPath` function again. The `KCFindInternetPasswordWithPath` function optionally returns a reference to the found item.

The `KCFindInternetPasswordWithPath` function automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the password is currently locked.

You can also call the function `kcfindinternetpasswordwithpath` to find the first Internet password item matching specified attributes. The function `kcfindinternetpasswordwithpath` requires that you pass a pointer to a C string instead of a pointer to a Pascal string for the `serverAddress`, `serverName`, `volumeName`, `accountName`, and `passwordData` parameters.

Version Notes
Available beginning with KeychainLib 2.0. In KeychainLib 1.0, the `kcfindinternetpassword` function provides the same functionality as the function `KCFindInternetPassword`, except that it accepts C strings rather than Pascal strings as arguments. In KeychainLib 2.0, you should use `KCFindInternetPassword`, since `kcfindinternetpassword` is provided for convenience only and may be removed from the header file at some point in the future.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainFindInternetPassword` function in Keychain Services instead.

Declared In
`KeychainCore.h`

kcfindinternetpasswordwithpath

Not recommended

```
OSStatus kcfindinternetpasswordwithpath (
   const char *serverName,
   const char *securityDomain,
   const char *accountName,
   const char *path,
   UInt16 port,
   OSType protocol,
   OSType authType,
   UInt32 maxLength,
   void *passwordData,
   UInt32 *actualLength,
   KCItemRef *item
);
```

Discussion
This function is available for convenience only and may be removed. Use the function `KCFindInternetPasswordWithPath` (page 37) instead.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainFindInternetPassword` function in Keychain Services instead.

Declared In
KeychainCore.h

KCFindNextItem

Finds the next keychain item matching the previously specified search criteria.

Not recommended

```
OSStatus KCFindNextItem (
   KCSearchRef search,
   KCItemRef *item
);
```

Parameters

search

A reference to the previously-specified search criteria. Pass the reference passed back in the searchRef parameter of the function KCFindFirstItem (page 31).

item

On return, a pointer to the next matching keychain item, if any.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code errKCNoDefaultKeychain indicates that no default keychain could be found. The result code errKCItemNotFound indicates that no matching keychain item was found. The result code errKCInvalidSearchRef indicates that the specified search reference was invalid.

Discussion
The KCFindNextItem function finds the next keychain item matching the search criteria previously specified by a call to the function KCFindFirstItem (page 31). The KCFindNextItem function returns a reference to the matching item, if any. The KCFindNextItem function automatically calls the function KCUnlock (page 59) to display the Unlock Keychain dialog box if the keychain containing the item you are searching for is currently locked.

Special Considerations

When you are completely finished with a search, you should use the functions KCReleaseItem (page 52) and KCReleaseSearch (page 54) to release the keychain item and search criteria reference.

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the SecKeychainSearchCopyNext function in Keychain Services instead.

Declared In
KeychainCore.h

KCFindX509Certificates

Finds the certificates in a keychain that match specified search criteria.

Unsupported

```
OSStatus KCFindX509Certificates (
    KCRef keychain,
    CFStringRef name,
    CFStringRef emailAddress,
    KCCertSearchOptions options,
    CFMutableArrayRef *certificateItems
);
```

Parameters

keychain

A reference to the keychain you want to search. If the keychain is locked, the Unlock Keychain dialog box is automatically displayed.

name

A pointer to a C string containing the certificate owner's common name.

emailAddress

A pointer to a C string containing the certificate owner's email address.

options

The search criteria you wish to use. See "Certificate Search Options" (page 68) for a description of this mask.

certificateItems

On return, a pointer to a list of the matching certificates. Pass `NULL` if you don't want to obtain these references.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that a default keychain was not found. The result code `errKCBufferTooSmall` indicates that the certificate data was too large for the supplied buffer. In this case, you should allocate a new buffer of sufficient size before calling `KCFindX509Certificates` again. The result code `errKCItemNotFound` indicates that no matching certificate was found.

Version Notes

Available beginning with KeychainLib 2.0.

Carbon Porting Notes

This function is obsolete. There is currently no replacement.

Declared In

`KeychainHI.h`

KCGetAttribute

Determines keychain item data using a keychain item attribute structure.

Not recommended

```
OSStatus KCGetAttribute (
   KCItemRef item,
   KCAttribute *attr,
   UInt32 *actualLength
);
```

Parameters

item

A reference to the keychain item whose attribute data you wish to determine.

attr

A pointer to a structure of type `KCAttribute` (page 63). Before calling the `KCGetAttribute` function, fill in the `tag`, `length`, and `data` fields (the `data` field should contain a pointer to a buffer of sufficient length for the type of data to be returned). On return, the `KCGetAttribute` function passes back the requested data in the `data` field.

actualLength

On return, a pointer to the actual length of the attribute data. This may be more than the length you allocated in the length field of the attribute structure.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCInvalidItemRef` indicates that the specified keychain item reference was invalid. The result code `errKCNoSuchAttr` indicates that you tried to set an attribute which is undefined for this item class. The result code `errKCBufferTooSmall` indicates that your application must allocate a new buffer of sufficient size before calling `KCGetAttribute` again.

Discussion

You can call the function `KCGetAttribute` or the function `KCGetData` (page 43) to obtain keychain item data. The difference between the functions is that the function `KCGetData` (page 43) requires that you pass the length of the data and a pointer to that data as separate parameters rather than fields in a keychain item attribute structure.

If the keychain that contains the item is locked, before calling the `KCGetAttribute` function you should call the function `KCUnlock` (page 59) to prompt the user to unlock the keychain.

You can determine any of the standard item attributes identified by the following tag constants: `kClassKCItemAttr`, `kCreationDateKCItemAttr`, `kModDateKCItemAttr`, `kDescriptionKCItemAttr`, `kCommentKCItemAttr`, `kLabelKCItemAttr`, `kCreatorKCItemAttr`, `kScriptCodeKCItemAttr`, and `kCustomIconKCItemAttr`. There is additional data you can determine, depending upon the type of keychain item whose data you wish to obtain. See "Keychain Item Attribute Tag Constants" (page 75) for more information.

Version Notes

Available beginning with KeychainLib 1.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemCopyAttributesAndData` function in Keychain Services instead.

Declared In

`KeychainCore.h`

KCGetData

Determines keychain item data.

Not recommended

```
OSStatus KCGetData (
   KCItemRef item,
   UInt32 maxLength,
   void *data,
   UInt32 *actualLength
);
```

Parameters

item

A reference to the keychain item whose data you wish to determine.

maxLength

The length of the data buffer pointed to by the `data` parameter.

data

A pointer to the buffer that holds the returned data. Before calling the `KCGetData` function, allocate enough memory for the buffer to hold the data you want to store. On return, a pointer to the attribute data you requested.

actualLength

On return, a pointer to the actual length of the data being retrieved. If the buffer pointed to by the `data` parameter is smaller than the actual length of the data, the `KCGetData` function returns the result code `errKCBufferTooSmall`. In this case, your application must allocate a new buffer of sufficient size before calling `KCGetData` again.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCInvalidItemRef` indicates that the specified keychain item reference was invalid. The result code `errKCBufferTooSmall` indicates that your application must allocate a new buffer of sufficient size before calling `KCGetData` again. The result code `errKCDataNotModifiable` indicates that the data is not available for this item.

Discussion

You can call the function `KCGetData` or the function `KCGetAttribute` (page 41) to obtain keychain item data. The difference between the functions is that the function `KCGetAttribute` (page 41) requires that you pass the length of the data and a pointer to that data as fields in a keychain item attribute structure rather than as separate parameters.

If the keychain that contains the item is locked, before calling the function `KCGetData` you should call the function `KCUnlock` (page 59) to prompt the user to unlock the keychain. You cannot call the `KCGetData` function for a private key.

You can determine any of the standard item attributes identified by the following tag constants: `kClassKCItemAttr`, `kCreationDateKCItemAttr`, `kModDateKCItemAttr`, `kDescriptionKCItemAttr`, `kCommentKCItemAttr`, `kLabelKCItemAttr`, `kCreatorKCItemAttr`, `kScriptCodeKCItemAttr`, and `kCustomIconKCItemAttr`. There is additional data you can determine, depending upon the type of keychain item whose data you wish to obtain. See "Keychain Item Attribute Tag Constants" (page 75) for more information.

Version Notes

Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemCopyContent` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCGetDefaultKeychain

Obtains the default keychain.

Not recommended

```
OSStatus KCGetDefaultKeychain (
   KCRef *keychain
);
```

Parameters

keychain

On return, a pointer to default keychain reference.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that there is no default keychain.

Discussion
You can determine the name of the default keychain by passing the returned keychain reference to the function `KCGetKeychainName` (page 47).

Version Notes
Available beginning with KeychainLib 2.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainCopyDefault` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCGetIndKeychain

Obtains the reference to an indexed keychain.

Not recommended

```
OSStatus KCGetIndKeychain (
   UInt16 index,
   KCRef *keychain
);
```

Parameters

index

An index of the list of available keychains. Pass a value between 1 and the number returned by the function `KCCountKeychains` (page 25).

keychain

On return, pointer to the keychain reference corresponding to the index in the `index` parameter.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoSuchKeychain` indicates that the index value is out of range.

Discussion

To guarantee correct operation, you should call the function `KCCountKeychains` (page 25) once before calling `KCGetIndKeychain`.

Special Considerations

The memory that the keychain reference occupies must be released by calling the function `KCReleaseKeychain` (page 53) when you are finished with it.

Version Notes

Available beginning with KeychainLib 1.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainCopySearchList` function in Keychain Services followed by a call to `CFArrayGetValueAtIndex` instead.

Declared In

`KeychainCore.h`

KCGetKeychain

Determines the location of a password or other keychain item.

Not recommended

```
OSStatus KCGetKeychain (
   KCItemRef item,
   KCRef *keychain
);
```

Parameters

item

A reference to the keychain item whose keychain location you wish to determine. If you pass a reference to a keychain item whose keychain is locked, the `KCGetKeychain` function returns the result code `errKCInvalidItemRef`.

keychain

On return, a pointer to the keychain containing the specified item.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCInvalidItemRef` indicates that the keychain item reference was invalid.

Discussion
The `KCGetKeychain` function determines the location of a keychain item in an unlocked keychain. It does not search locked keychains. Calling the `KCGetKeychain` function displays the Unlock Keychain dialog box if the keychain containing the item is currently locked.

Special Considerations

The keychain reference returned by `KCGetKeychain` should be released by calling the function `KCReleaseItem` (page 52).

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemCopyKeychain` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCGetKeychainManagerVersion

Determines the version of the Keychain Manager installed on the user's system.

Not Recommended

```
OSStatus KCGetKeychainManagerVersion (
   UInt32 *returnVers
);
```

Parameters
returnVers

On return, a pointer to the version number of the Keychain Manager installed on the current system.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85).

Discussion
Your application can call the `KCGetKeychainManagerVersion` function to find out which version of the Keychain Manager is installed on the user's system.

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainGetVersion` function in Keychain Services instead.

Declared In

`KeychainCore.h`

KCGetKeychainName

Determines the name of a keychain.

Not recommended

```
OSStatus KCGetKeychainName (
   KCRef keychain,
   StringPtr keychainName
);
```

Parameters

keychain

A reference to the keychain whose name you wish to obtain.

keychainName

A pointer to a Pascal string. On return, this string contains the name of the keychain.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCInvalidKeychain` indicates that the keychain is invalid.

Discussion

You can also call the function `kcgetkeychainname` to obtain the name of a keychain. `kcgetkeychainname` requires that you pass a pointer to a C string instead of a pointer to a Pascal string in the `keychainName` parameter.

Version Notes

Available beginning with KeychainLib 2.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainGetPath` function in Keychain Services instead.

Declared In

`KeychainCore.h`

kcgetkeychainname

Not recommended

```
OSStatus kcgetkeychainname (
   KCRef keychain,
   char *keychainName
);
```

Discussion
This function is available for convenience only and may be removed. Use the function `KCGetKeychainName` (page 47) instead.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainGetPath` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCGetStatus

Determines the permissions that are set in a keychain.

Not recommended

```
OSStatus KCGetStatus (
   KCRef keychain,
   UInt32 *keychainStatus
);
```

Parameters

keychain

A pointer to the keychain reference whose permissions you wish to determine. Pass `NULL` to obtain the status of the default keychain.

keychainStatus

On return, a pointer to a bitmask that you can test to determine the permissions that are set in a keychain. See "Keychain Status Constants" (page 84) for a description of this mask.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoSuchKeychain` indicates that the specified keychain could not be found. The result code `errKCInvalidKeychain` indicates that the specified keychain is invalid.

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainGetStatus` function in Keychain Services instead.

Declared In
KeychainCore.h

KCIsInteractionAllowed

Indicates whether Keychain Manager functions that display a user interaction will do so.

Not recommended

```
Boolean KCIsInteractionAllowed (
   void
);
```

Parameters

Return Value
A Boolean value indicating whether user interaction is permitted. If true, user interaction is allowed, and Keychain Manager functions that display a user interface can do so as appropriate.

Version Notes
Available beginning with KeychainLib 2.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the SecKeychainGetUserInteractionAllowed function in Keychain Services instead.

Declared In
KeychainCore.h

KCLock

Locks a keychain.

Not recommended

```
OSStatus KCLock (
   KCRef keychain
);
```

Parameters

keychain

A reference to the keychain to lock. Pass NULL to lock all unlocked keychains.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code errKCNoSuchKeychain indicates that specified keychain could not be found. The result code errKCInvalidKeychain indicates that the specified keychain is invalid.

Discussion
Your application should not call the KCLock function unless you are responding to a user's request to lock a keychain. In general, you should leave the keychain unlocked so that the user does not have to unlock it again in another application.

Version Notes
The function `KCLock` replaces the function `KCLockKeychain`, which was available in KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainLock` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCMakeAliasFromKCRef

Creates an alias to a keychain reference.

Not Recommended

```
OSStatus KCMakeAliasFromKCRef (
   KCRef keychain,
   AliasHandle *keychainAlias
);
```

Parameters

keychain

A reference to the keychain for which you want to create an alias.

keychainAlias

On return, a pointer to an alias handle to the file referred to by the keychain reference.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85).

Discussion
You may wish to call the `KCMakeAliasFromKCRef` function to determine the location of a keychain.

Special Considerations

When you are finished with a keychain, you should call the function `KCReleaseKeychain` (page 53) to deallocate its memory. You should not use the keychain after its memory has been deallocated.

Version Notes
Available beginning with KeychainLib 2.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainGetPath` function in Keychain Services followed by calls to the function `FSPathMakeRef` and `FSNewAlias` instead.

Declared In
`KeychainCore.h`

KCMakeKCRefFromAlias

Creates a keychain reference from a keychain alias.

Not Recommended

```
OSStatus KCMakeKCRefFromAlias (
   AliasHandle keychainAlias,
   KCRef *keychain
);
```

Parameters

keychainAlias

A handle to an alias record of the keychain file. Since the keychain is a file, an alias can be made to the keychain file.

keychain

On return, a pointer to a reference to the keychain specified by the alias in the `keychainAlias` parameter.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85).

Special Considerations

When you are finished with a keychain, you should call the function `KCReleaseKeychain` (page 53) to deallocate its memory. You should not use the keychain after its memory has been deallocated.

Version Notes

Available beginning with KeychainLib 2.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainOpen` function in Keychain Services instead. If the keychain doesn't exist, use the `SecKeychainCreate` function in Keychain Services.

Declared In

`KeychainCore.h`

KCNewItem

Creates a reference to a keychain item.

Not recommended

```
OSStatus KCNewItem (
   KCItemClass itemClass,
   OSType itemCreator,
   UInt32 length,
   const void *data,
   KCItemRef *item
);
```

Parameters

itemClass

The type of keychain item that you wish to create. See "Keychain Item Type Constants" (page 81) for a description of possible values and a description of the `KCItemClass` data type.

itemCreator

The creator code of the application that owns this item.

length

The length of the data to be stored in this item.

data

A pointer to a buffer containing the data to be stored in this item. Before calling `KCNewItem`, allocate enough memory for the buffer to hold the data you want to store.

item

On return, a pointer to a reference to the newly-created item.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The Memory Manager result code `memFullErr` indicates that you did not allocate enough memory in the current heap to create the item.

Discussion

After calling the `KCNewItem` function, you should call the function `KCAddItem` (page 22) if you wish to permanently store a password or other keychain item. Note that a copy of the data buffer pointed to by the `data` parameter is stored in the newly-created item.

Special Considerations

When you are done with a keychain item, you should call the function `KCReleaseItem` (page 52) to release its memory. You should not use the item after its memory has been deallocated.

Version Notes

Available beginning with KeychainLib 1.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemCreateFromContent` function in Keychain Services instead.

Declared In

`KeychainCore.h`

KCReleaseItem

Disposes of the memory occupied by a keychain item reference.

Not recommended

```
OSStatus KCReleaseItem (
   KCItemRef *item
);
```

Parameters

item

A pointer to a keychain item reference. Pass the keychain item reference whose memory you want to release. On return, the reference is set to `NULL` and should not be used again.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85).

Discussion
You should call the `KCReleaseItem` function to release the memory occupied by a keychain item reference when you are finished with it.

Version Notes
Available beginning with KeychainLib 1.0

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `CFRelease` function instead.

Declared In
`KeychainCore.h`

KCReleaseKeychain

Disposes of the memory associated with a keychain reference.

Not recommended

```
OSStatus KCReleaseKeychain (
   KCRef *keychain
);
```

Parameters

keychain

A pointer to a keychain reference. Pass the keychain reference whose memory you want to release. On return, the reference is set to `NULL` and should not be used again.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85).

Discussion
You should call the `KCReleaseKeychain` function to release the memory occupied by a keychain reference when you are finished with it. You should not use the reference after it has been released.

Version Notes
Available beginning with KeychainLib 2.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `CFRelease` function instead.

Declared In
`KeychainCore.h`

KCReleaseSearch

Disposes of the memory occupied by a search criteria reference.

Not recommended

```
OSStatus KCReleaseSearch (
   KCSearchRef *search
);
```

Parameters

search

A pointer to a search criteria reference. Pass the search criteria reference whose memory you want to release. On return, the reference is set to `NULL` and should not be used again.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCInvalidSearchRef` indicates that the specified search reference was invalid.

Discussion
You should call the `KCReleaseSearch` function to release the memory occupied by a search criteria reference when you are completely finished with a search performed by calling the functions `KCFindFirstItem` (page 31) or `KCFindNextItem` (page 40).

Version Notes
Available beginning with KeychainLib 1.0

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `CFRelease` function instead.

Declared In
`KeychainCore.h`

KCRemoveCallback

Unregisters your keychain event callback function.

Not recommended

```
OSStatus KCRemoveCallback (
   KCCallbackUPP callbackProc
);
```

Parameters

callbackProc

A Universal Procedure Pointer (UPP) to your keychain event callback function that was previously registered with the function `KCAddCallback` (page 15).

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCInvalidCallback` indicates that the callback function was not previously registered.

Discussion

After you pass a UPP to your keychain event callback function to the `KCRemoveCallback` function, it will no longer be called by the Keychain Manager.

Special Considerations

After calling `KCRemoveCallback`, you should call the function `DisposeKCCallbackUPP` (page 11) to dispose of the UPP to your callback function.

Version Notes

Available beginning with KeychainLib 1.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainRemoveCalllback` function in Keychain Services instead.

Declared In

`KeychainCore.h`

KCSetAttribute

Sets or edits keychain item data using a keychain item attribute structure.

Not recommended

```
OSStatus KCSetAttribute (
   KCItemRef item,
   KCAttribute *attr
);
```

Parameters

item

A reference to the keychain item whose data you wish to set or edit.

attr

A pointer to a structure of type `KCAttribute` (page 63) containing keychain item data you want to set. Before calling the function `KCSetAttribute`, fill in the `tag`, `length`, and `data` fields of this structure with the tag identifying the attribute you wish to modify or set, the length of the attribute data you wish to set, and a pointer to that data, respectively.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCInvalidItemRef` indicates that the keychain item reference was invalid. The result code `errKCNoSuchAttr` indicates that the item attribute you wish to set is undefined for the specified item. The result code `errKCDataTooLarge` indicates that more data was supplied than is allowed for this attribute.

Discussion
You can call the `KCSetAttribute` function or the function `KCSetData` (page 56) to set or modify keychain item data. The difference between the functions is that the `KCSetData` (page 56) function requires that you pass the length of the data and a pointer to that data as separate parameters rather than fields in a keychain item attribute structure.

If the keychain that contains the item is locked, before calling the `KCSetAttribute` function you should call the function `KCUnlock` (page 59) to prompt the user to unlock the keychain. The keychain must permit read/write access in order to modify keychain item data.

You can only set or modify standard item attributes identified by the tag constants `kDescriptionKCItemAttr`, `kCommentKCItemAttr`, `kLabelKCItemAttr`, `kCreatorKCItemAttr`, `kTypeKCItemAttr`, and `kCustomIconKCItemAttr`. In addition, each class of keychain item has attributes specific to that class which may be set or modified. See "Keychain Item Attribute Tag Constants" (page 75) for more information.

Version Notes
Available beginning with KeychainLib 1.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemModifyAttributesAndData` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCSetData

Sets or edits keychain item data.

Not recommended

```
OSStatus KCSetData (
   KCItemRef item,
   UInt32 length,
   const void *data
);
```

Parameters

item

A reference to the keychain item whose data you wish to set.

length

The length of the data buffer pointed to by the `data` parameter.

data

A pointer to a buffer containing the data to be stored in this item. Before calling the `KCSetData` function, allocate enough memory for the buffer to hold the data you want to store.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCInvalidItemRef` indicates that the specified keychain item reference was invalid. The result code `errKCDataTooLarge` indicates that the data was too large for the supplied buffer. The result code `errKCDataNotModifiable` indicates that the data cannot be set for this item.

Discussion

You can call the function `KCSetData` or the function `KCSetAttribute` (page 55) to set or modify keychain item data. The difference between the functions is that the function `KCSetAttribute` (page 55) requires that you pass the length of the data buffer as a field in a keychain item attribute structure rather than as a separate parameter.

If the keychain that contains the item is locked, before calling the `KCSetData` function you should call the function `KCUnlock` (page 59) to prompt the user to unlock the keychain. The keychain must permit read/write access in order to modify keychain item data.

You can set or edit any of the standard item attributes identified by the following tag constants: `kDescriptionKCItemAttr`, `kCommentKCItemAttr`, `kLabelKCItemAttr`, `kCreatorKCItemAttr`, `kTypeKCItemAttr`, and `kCustomIconKCItemAttr`. There is additional data you can set, depending upon the type of keychain item whose data you are manipulating. See "Keychain Item Attribute Tag Constants" (page 75) for more information.

Version Notes

Available beginning with KeychainLib 1.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemModifyContent` function in Keychain Services instead.

Declared In

`KeychainCore.h`

KCSetDefaultKeychain

Sets the default keychain.

Not recommended

```
OSStatus KCSetDefaultKeychain (
   KCRef keychain
);
```

Parameters

keychain

A reference to the keychain you wish to make the default.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoSuchKeychain` indicates that the specified keychain could not be found. The result code `errKCInvalidKeychain` indicates that the specified keychain is invalid.

Discussion
In most cases, your application should not need to set the default keychain, because this is a choice normally made by the user. You should call the `KCSetDefaultKeychain` function to change where a password or other keychain items are added.

The `KCSetDefaultKeychain` function sets the default keychain regardless of whether the keychain is currently locked.

Version Notes
Available beginning with KeychainLib 2.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainSetDefault` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCSetInteractionAllowed

Enables or disables Keychain Manager functions that display a user interface.

Not recommended

```
OSStatus KCSetInteractionAllowed (
   Boolean state
);
```

Parameters

state

A flag that indicates whether the Keychain Manager will display a user interface. If you pass `true`, user interaction is allowed. This is the default value. If `false`, Keychain Manager functions that normally display a user interface will instead return an error.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85).

Discussion
The `KCSetInteractionAllowed` function enables you to control whether the functions `KCLock` (page 49), `KCUnlock` (page 59), and `KCChangeSettings` (page 23) display a user interface. Note that failure to re-enable user interaction will affect other clients of the Keychain Manager. By default, user interaction is permitted.

Version Notes
Available beginning with KeychainLib 2.0.

Availability
Available in CarbonLib 1.1 and later when KeychainLib 2.0 or later is present.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainSetUserInteractionAllowed` function in Keychain Services instead.

Declared In
`KeychainCore.h`

KCUnlock

Displays a dialog box that prompts the user for a password before unlocking a keychain.

Not recommended

```
OSStatus KCUnlock (
   KCRef keychain,
   StringPtr password
);
```

Parameters

keychain

A reference to the keychain to unlock. Pass `NULL` to specify the default keychain. If you pass `NULL` and the default keychain is currently locked, the keychain will appear as the default choice. If you pass a locked keychain, the function `KCUnlock` displays the Unlock Keychain dialog box and the keychain appears as the chosen menu item in the keychain popup menu. If the default keychain is currently unlocked, the Unlock Keychain dialog box is not displayed and the `KCUnlock` function returns `noErr`.

password

A pointer to a Pascal string representing the password string for this keychain. Pass `NULL` if the user password is unknown. In this case, The `KCUnlock` function displays the Unlock Keychain dialog box, and the authentication user interface associated with the keychain about to be unlocked. If you specify an invalid password, you will not be able to unlock the keychain with a specified password until the machine is rebooted. In this case, the `KCUnlock` function returns `errKCInteractionRequired`.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). The result code `noErr` does not guarantee that the specified keychain is unlocked, because the user can select any available keychain and unlock it. The result code `userCanceledErr` indicates that the user pressed the Cancel button in the Unlock Keychain dialog box. The result code `errKCAuthFailed` indicates that authentication failed because of too many unsuccessful retries. The result code `errKCInteractionRequired` indicates that user interaction is required to unlock the keychain. In this case, you will not be able to unlock the keychain with that password until the machine is rebooted.

Discussion
In most cases, your application does not need to call the `KCUnlock` function directly, since most Keychain Manager functions that require an unlocked keychain call `KCUnlock` automatically. If your application needs to verify that a keychain is unlocked, call the function `KCGetStatus` (page 48).

You can also call the function `kcunlock` to display a user interface prompting the user to unlock a keychain. The `kcunlock` function requires that you pass a pointer to a C string instead of a pointer to a Pascal string in the `password` parameter.

Special Considerations

It is recommended that the `KCUnlock` function not be explicitly called by applications. Most functions that require an unlocked keychain call the `KCUnlock` function for you.

The memory that the keychain reference occupies must be released by calling the function `KCReleaseKeychain` (page 53) when you are finished with it.

Version Notes

The `KCUnlock` function replaces the function `KCUnlockKeychain`, which was available in KeychainLib 1.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainUnlock` function in Keychain Services instead.

Declared In

`KeychainHI.h`

kcunlock

Not recommended

```
OSStatus kcunlock (
   KCRef keychain,
   const char *password
);
```

Discussion

This function is available for convenience only and may be removed. Use the function `KCUnlock` (page 59) instead.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainUnlock` function in Keychain Services instead.

Declared In

`KeychainHI.h`

KCUpdateItem

Updates a password or other keychain item.

Not recommended

```
OSStatus KCUpdateItem (
   KCItemRef item
);
```

Parameters

item

A reference to the keychain item whose data you wish to update. If you pass an item that has not been previously added to the keychain, the `KCUpdateItem` function does nothing and returns `noErr`.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85). The result code `errKCNoDefaultKeychain` indicates that no default keychain could be found. The result code `errKCInvalidItemRef` indicates that the specified keychain item reference was invalid.

Discussion

You can use the `KCUpdateItem` function to update a password or other keychain item in a keychain's permanent data store after changing its data. The function `KCUpdateItem` automatically calls the function `KCUnlock` (page 59) to display the Unlock Keychain dialog box if the keychain containing the item is currently locked.

Version Notes

Available beginning with KeychainLib 1.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 1.0 or later is present.

Available in Mac OS X 10.0 and later.

Carbon Porting Notes

Use the `SecKeychainItemModifyContent` function in Keychain Services instead.

Declared In

`KeychainCore.h`

NewKCCallbackUPP

Creates a UPP to your keychain event callback.

Not recommended

```
KCCallbackUPP NewKCCallbackUPP (
   KCCallbackProcPtr userRoutine
);
```

Parameters

userRoutine

A pointer to your keychain event callback function. For information on how to create a keychain event callback, see `KCCallbackProcPtr` (page 62).

Return Value

A UPP to your callback function. You can register your callback function by passing this UPP in the `callbackProc` parameter of the function `KCAddCallback` (page 15). See the description of the `KCCallbackUPP` data type.

Discussion
The `NewKCCallbackUPP` function creates a pointer to your keychain event callback function. You pass a pointer to your callback function in the `callbackProc` parameter of the function `KCAddCallback` (page 15) if you want your application to receive data transfer events.

Special Considerations

When you are finished with a UPP to your keychain event callback function, you should dispose of it by calling the function `DisposeKCCallbackUPP` (page 11).

Availability
Available in CarbonLib 1.1 and later.
Available in Mac OS X 10.0 and later.

Carbon Porting Notes

There is no replacement function available.

Declared In
`KeychainCore.h`

Callbacks

KCCallbackProcPtr

Defines a pointer to your keychain event callback that handles user keychain access events.

```
typedef OSStatus (*KCCallbackProcPtr)
(
        KCEvent keychainEvent,
        KCCallbackInfo * info,
        void * userContext
);
```

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

```
OSStatus MyKCCallbackProc (
        KCEvent keychainEvent,
        KCCallbackInfo * info,
        void * userContext
);
```

Parameters

keychainEvent

The keychain event that your application wishes to be notified of. See "Keychain Events Constants" (page 71) for a description of possible values. The type of event that can trigger your callback depends on the bitmask you passed in the `eventMask` parameter of the function `KCAddCallback` (page 15). For more information, see the discussion.

info

A pointer to a structure of type `KCCallbackInfo` (page 65). On return, the structure contains information about the keychain event that occurred. The Keychain Manager passes this information to your callback function via the `info` parameter of the function `InvokeKCCallbackUPP` (page 11).

userContext

A pointer to application-defined storage that your application previously passed to the function KCAddCallback (page 15). You can use this value to perform operations such as tracking which instance of a function is operating.

Return Value
A result code. See "Keychain Manager Result Codes" (page 85). Your keychain event callback function should process the keychain event and return noErr.

Discussion
Your keychain event callback function handles those keychain events that you indicate. In order to be notified of these events, you must pass a UPP to your notification callback function in the callbackProc parameter of KCAddCallback (page 15). You indicate the type of data transfer events you want to receive via a bitmask in the eventMask parameter. When you no longer wish to receive notification of keychain events, you should call the function KCRemoveCallback (page 54) to dispose of the UPP to your keychain event callback function.

Carbon Porting Notes

Use the SecKeychainCalltback function in Keychain Services instead.

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

Declared In
KeychainCore.h

Data Types

AFPServerSignature

Represents a 16-byte Apple File Protocol server signature block.

```
typedef UInt8 AFPServerSignature[16];
```

Discussion
The AFPServerSignature type represents a 16-byte Apple File Protocol server signature block. You can pass a value of this type in the serverSignature parameter of the functions KCAddAppleSharePassword (page 12) and KCFindAppleSharePassword (page 28) to represent an Apple File Protocol server signature. You can use a value of this type with the keychain item attribute constant kSignatureKCItemAttr to specify an Apple File Protocol server signature.

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

Declared In
KeychainCore.h

KCAttribute

Contains information about a keychain item attribute.

```
typedef SecKeychainAttribute KCAttribute;
```

Discussion
The `KCAttribute` type represents a structure containing information about the attribute of a keychain item. It contains a tag that identifies a particular keychain item attribute value, the length of the attribute value, and a pointer to the attribute value. You can modify attribute data for a keychain item attribute by passing a pointer to this structure in the `attr` parameter of the function `KCSetAttribute` (page 55). The function `KCGetAttribute` (page 41) passes back a pointer to this structure in the `attr` parameter.

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

Declared In
`KeychainCore.h`

KCAttributeList

Lists attributes in a keychain item.

```
typedef SecKeychainAttributeList KCAttributeList;
```

Discussion
The `KCAttributeList` type represents a list of structures containing information about the attributes in a keychain item. You pass a pointer to this list of 0 or more structures in the `attrList` parameter of the function `KCFindFirstItem` (page 31) to indicate the attributes to be matched.

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

Declared In
`KeychainCore.h`

KCAttrType

Identifies a keychain item attribute value.

```
typedef SecKeychainAttrType KCAttrType;
```

Discussion
The `KCAttrType` type represents a tag that identifies a keychain item attribute value. You can use this value in the `tag` field of the structure `KCAttribute` (page 63) to identify the keychain item attribute value you wish to set or obtain. See `Keychain Item Attribute Tag Constants` (page 75) for a description of the Apple-defined tag constants and the data types of the values they identify. Your application can create application-defined tags of type `KCAttrType`.

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

Declared In
`KeychainCore.h`

KCCallbackInfo

Contains information about a keychain event.

```
struct KCCallbackInfo {
    UInt32 version;
    KCItemRef item;
    long processID[2];
    long event[4];
    KCRef keychain;
};
typedef struct KCCallbackInfo KCCallbackInfo;
```

Fields

`version`
: The version of this structure.

`item`
: A reference to the keychain item in which the event occurred. If the event did not involve an item, this field is not valid.

`processID`
: A 64-bit quantity containing the process serial number of the process in which the event occurred. This is not available on Mac OS X.

`event`
: The keychain event that occurred. If the event is a system event as indicated by the constant `kSystemKCEvent`, the Keychain client can process events. If the event is not a system event, this field is not valid. This is not available on Mac OS X.

`keychain`
: A reference to the keychain in which the event occurred. If the event did not involve a keychain, this field is not valid.

Discussion

The `KCCallbackInfo` type represents a structure that contains information about the keychain event of which your application wants to be notified. The Keychain Manager passes a pointer to this structure in the `info` parameter of your callback function via the function `InvokeKCCallbackUPP` (page 11), which invokes your callback function. For information on how to write a keychain event callback function, see `KCCallbackProcPtr` (page 62).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`KeychainCore.h`

KCCallbackUPP

Defines a data type for the `KCCallbackProcPtr` callback pointer.

```
typedef KCCallbackProcPtr KCCallbackUPP;
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`KeychainCore.h`

KCItemRef

Represents a reference to a keychain item.

```
typedef SecKeychainItemRef KCItemRef;
```

Discussion
The `KCItemRef` type represents a reference to an opaque structure that identifies a keychain item. You should call the function `KCNewItem` (page 51) to create a keychain item reference. The function `KCReleaseItem` (page 52) disposes of a keychain item reference when no longer needed.

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

Declared In
`KeychainCore.h`

KCPublicKeyHash

Represents a 20-byte public key hash.

```
typedef UInt8 KCPublicKeyHash[20];
```

Discussion
The `KCPublicKeyHash` type represents a hash of a public key. You can use the constant `kPublicKeyHashKCItemAttr`, described in `Keychain Item Attribute Tag Constants` (page 75), to set or retrieve a certificate attribute value of this type.

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

Declared In
`KeychainCore.h`

KCRef

Represents a reference to a keychain.

```
typedef SecKeychainRef KCRef;
```

Discussion
The `KCRef` type represents a reference to an opaque structure that identifies a keychain. You should call the function `KCMakeKCRefFromFSSpec` (page 89) or `KCMakeKCRefFromAlias` (page 51) to create a keychain reference. The function `KCReleaseKeychain` (page 53) disposes of a keychain reference when no longer needed. You pass a reference of this type to Keychain Manager functions that operate on a keychain in some way.

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

Declared In
`KeychainCore.h`

KCSearchRef

Represents a reference to the current search criteria.

```
typedef SecKeychainSearchRef KCSearchRef;
```

Discussion
The `KCSearchRef` type represents a reference to an opaque structure that identifies the current search criteria. The function `KCFindFirstItem` (page 31) passes back a reference of this type in the `search` parameter for subsequent calls to the function `KCFindNextItem` (page 40). You must release this reference when you are finished with a search by calling the function `KCReleaseSearch` (page 54).

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

Declared In
`KeychainCore.h`

KCStatus

Identifies a mask that you can use in determining the permissions that are set in a keychain.

```
typedef SecKeychainStatus KCStatus;
```

Discussion
The `KCStatus` enumeration defines masks your application can use to determine the read and write permissions for a keychain. The function `KCGetStatus` (page 48) passes back this mask in the `status` parameter.

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

Declared In
`KeychainCore.h`

Constants

Authentication Type Constants

Represent the type of authentication to use in storing and retrieving Internet passwords.

```
enum {
    kKCAuthTypeNTLM = 'ntlm',
    kKCAuthTypeMSN = 'msna',
    kKCAuthTypeDPA = 'dpaa',
    kKCAuthTypeRPA = 'rpaa',
    kKCAuthTypeHTTPDigest = 'httd',
    kKCAuthTypeDefault = 'dflt'
};
typedef FourCharCode KCAuthType;
```

Constants

`kKCAuthTypeNTLM`
Specifies Windows NT LAN Manager authentication.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCAuthTypeMSN`
Specifies Microsoft Network authentication.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCAuthTypeDPA`
Specifies Distributed Password authentication.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCAuthTypeRPA`
Specifies Remote Password authentication.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCAuthTypeHTTPDigest`
Specifies HTTP Digest Access authentication.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCAuthTypeDefault`
Specifies default authentication.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

Discussion

The `KCAuthType` enumeration defines constants you can use to identify the type of authentication to use in storing and retrieving Internet passwords. You can pass a constant of this type in the `authType` parameter of the functions `KCAddInternetPassword` (page 18), `KCAddInternetPasswordWithPath` (page 20), `KCFindInternetPassword` (page 34), and `KCFindInternetPasswordWithPath` (page 37).

Certificate Search Options

Represent a mask that specifies the search criteria to use when finding certificates.

```
typedef UInt32 KCCertSearchOptions;
enum {
    kCertSearchShift = 0,
    kCertSearchSigningIgnored = 0,
    kCertSearchSigningAllowed = 1 << (kCertSearchShift + 0),
    kCertSearchSigningDisallowed = 1 << (kCertSearchShift + 1),
    kCertSearchSigningMask = ((kCertSearchSigningAllowed) |
        (kCertSearchSigningDisallowed)),
    kCertSearchVerifyIgnored = 0,
    kCertSearchVerifyAllowed = 1 << (kCertSearchShift + 2),
    kCertSearchVerifyDisallowed = 1 << (kCertSearchShift + 3),
    kCertSearchVerifyMask = ((kCertSearchVerifyAllowed) |
        (kCertSearchVerifyDisallowed)),
    kCertSearchEncryptIgnored = 0,
    kCertSearchEncryptAllowed = 1 << (kCertSearchShift + 4),
    kCertSearchEncryptDisallowed = 1 << (kCertSearchShift + 5),
    kCertSearchEncryptMask = ((kCertSearchEncryptAllowed) |
        (kCertSearchEncryptDisallowed)),
    kCertSearchDecryptIgnored = 0,
    kCertSearchDecryptAllowed = 1 << (kCertSearchShift + 6),
    kCertSearchDecryptDisallowed = 1 << (kCertSearchShift + 7),
    kCertSearchDecryptMask = ((kCertSearchDecryptAllowed) |
        (kCertSearchDecryptDisallowed)),
    kCertSearchWrapIgnored = 0,
    kCertSearchWrapAllowed = 1 << (kCertSearchShift + 8),
    kCertSearchWrapDisallowed = 1 << (kCertSearchShift + 9),
    kCertSearchWrapMask = ((kCertSearchWrapAllowed) |
        (kCertSearchWrapDisallowed)),
    kCertSearchUnwrapIgnored = 0,
    kCertSearchUnwrapAllowed = 1 << (kCertSearchShift + 10),
    kCertSearchUnwrapDisallowed = 1 << (kCertSearchShift + 11),
    kCertSearchUnwrapMask = ((kCertSearchUnwrapAllowed) |
        (kCertSearchUnwrapDisallowed)),
    kCertSearchPrivKeyRequired = 1 << (kCertSearchShift + 12),
    kCertSearchAny = 0
};
```

Discussion
The `KCCertSearchOptions` enumeration defines masks that you can use in the `options` parameter of the function `KCFindX509Certificates` (page 41).

Certificate Usage Options

Represent a mask that specifies the usage options when adding certificates.

```
typedef UInt32 KCCertAddOptions;
enum {
    kSecOptionReserved = 0x000000FF,
    kCertUsageShift = 8,
    kCertUsageSigningAdd         = 1 << (kCertUsageShift + 0),
    kCertUsageSigningAskAndAdd = 1 << (kCertUsageShift + 1),
```

```
    kCertUsageVerifyAdd = 1 << (kCertUsageShift + 2),
    kCertUsageVerifyAskAndAdd = 1 << (kCertUsageShift + 3),
    kCertUsageEncryptAdd = 1 <<(kCertUsageShift + 4),
    kCertUsageEncryptAskAndAdd = 1 << (kCertUsageShift + 5),
    kCertUsageDecryptAdd = 1 << (kCertUsageShift + 6),
    kCertUsageDecryptAskAndAdd = 1 << (kCertUsageShift + 7),
    kCertUsageKeyExchAdd = 1 << (kCertUsageShift + 8),
    kCertUsageKeyExchAskAndAdd = 1 << (kCertUsageShift + 9),
    kCertUsageRootAdd = 1 << (kCertUsageShift + 10),
    kCertUsageRootAskAndAdd = 1 << (kCertUsageShift + 11),
    kCertUsageSSLAdd = 1 << (kCertUsageShift + 12),
    kCertUsageSSLAskAndAdd = 1 << (kCertUsageShift + 13),
    kCertUsageAllAdd = 0x7FFFFF00
};
```

Certificate Verification Criteria

Identify the verification criteria for use when displaying certificates to the user.

```
typedef UInt16 KCVerifyStopOn;
enum {
    kPolicyKCStopOn = 0,
    kNoneKCStopOn = 1,
    kFirstPassKCStopOn = 2,
    kFirstFailKCStopOn = 3
};
```

Constants

`kPolicyKCStopOn`

Indicates that the function `KCChooseCertificate` (page 24) should use the trust policy options currently in effect.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kNoneKCStopOn`

Indicates that the function `KCChooseCertificate` (page 24) completes after examining all available certificates.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kFirstPassKCStopOn`

Indicates that the function `KCChooseCertificate` (page 24) when one certificate meeting the verification criteria is found.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kFirstFailKCStopOn`

Specifies that the function `KCChooseCertificate` (page 24) completes when one certificate that fails to meet the verification criteria is found.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

Discussion
The `KCVerifyStopOn` enumeration defines constants your application can use to identify the verification criteria to use in selecting certificates. You can pass a constant of this type in the `stopOn` parameter of the function `KCChooseCertificate` (page 24).

Default Internet Port Constant

Represent the internet ports available.

```
enum {
    kAnyPort = 0
};
```

Constants
`kAnyPort`
Indicates that any Internet port can be used.
Available in Mac OS X v10.1 and later.
Declared in `KeychainCore.h`.

Default Internet Protocol And Authentication Type Constants

Represent the internet protocols and authentication types available.

```
enum {
    kAnyProtocol = 0,
    kAnyAuthType = 0
};
```

Constants
`kAnyProtocol`
Indicates that any Internet protocol can be used.
Available in Mac OS X v10.1 and later.
Declared in `KeychainCore.h`.

`kAnyAuthType`
Indicates that any Internet authentication type can be used.
Available in Mac OS X v10.1 and later.
Declared in `KeychainCore.h`.

Keychain Events Constants

Identify keychain events.

```
typedef UInt16 KCEvent;
enum {
    kIdleKCEvent = 0,
    kLockKCEvent = 1,
    kUnlockKCEvent = 2,
    kAddKCEvent = 3,
    kDeleteKCEvent = 4,
    kUpdateKCEvent = 5,
    kPasswordChangedKCEvent = 6,
    kSystemKCEvent = 8,
    kDefaultChangedKCEvent = 9,
    kDataAccessKCEvent = 10,
    kKeychainListChangedKCEvent = 11
};
```

Constants

`kIdleKCEvent`
Indicates a `NULL` event.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kLockKCEvent`
Indicates that the keychain was locked.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kUnlockKCEvent`
Indicates that the keychain was unlocked.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kAddKCEvent`
Indicates that an item was added to a keychain.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kDeleteKCEvent`
Indicates that an item was deleted from a keychain.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kUpdateKCEvent`
Indicates that a keychain item was updated.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kPasswordChangedKCEvent`
Indicates that the identity of the keychain was changed.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kSystemKCEvent`
Indicates that the keychain client can process events.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kDefaultChangedKCEvent`
Indicates that the default keychain has changed.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kDataAccessKCEvent`
Indicates that a process has called the function `KCGetData` (page 43) to access a keychain item's data.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kKeychainListChangedKCEvent`
Indicates that the list of keychains has changed.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

Discussion
The `KCEvent` enumeration defines constants that identify the Keychain-related events your callback function wishes to receive. The Keychain Manager tests a mask that you pass in the `eventMask` parameter of the function `KCAddCallback` (page 15) to determine the data transfer events your notification callback function wishes to receive. It passes these events in the `keychainEvent` parameter of the function `InvokeKCCallbackUPP` (page 11). For a description of the Keychain-related event masks, see `Keychain Events Mask` (page 73).

Keychain Events Mask

Identify a mask that you can use to set the keychain events you wish to receive.

```
typedef UInt16 KCEventMask;
enum {
    kIdleKCEventMask = 1 << kIdleKCEvent,
    kLockKCEventMask = 1 << kLockKCEvent,
    kUnlockKCEventMask = 1 << kUnlockKCEvent,
    kAddKCEventMask = 1 << kAddKCEvent,
    kDeleteKCEventMask = 1 << kDeleteKCEvent,
    kUpdateKCEventMask = 1 << kUpdateKCEvent,
    kPasswordChangedKCEventMask = 1 << kPasswordChangedKCEvent,
    kSystemEventKCEventMask = 1 << kSystemKCEvent,
    kDefaultChangedKCEventMask = 1 << kDefaultChangedKCEvent,
    kDataAccessKCEventMask = 1 << kDataAccessKCEvent,
    kEveryKCEventMask = 0xFFFF
};
```

Constants
`kIdleKCEventMask`
If the bit specified by this mask is set, your callback function will be invoked during a `NULL` event.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kLockKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when the keychain is locked.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kUnlockKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when the keychain is unlocked.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kAddKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when an item is added to the keychain.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kDeleteKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when an item is removed from the keychain.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kUpdateKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when a keychain item is updated.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kPasswordChangedKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when the keychain identity is changed.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kSystemEventKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when the keychain client processes an event.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kDefaultChangedKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when the default keychain is changed.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kDataAccessKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when a process calls the function `KCGetData` (page 43).

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kEveryKCEventMask`

If the bit specified by this mask is set, your callback function will be invoked when any of the above Keychain-related events occur.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

Discussion

The `KCEventMask` enumeration defines masks your application can use to set Keychain event bits. You pass this mask in the `eventMask` parameter of the function `KCAddCallback` (page 15), thereby defining the Keychain-related events to which your callback will respond. The Keychain Manager uses this mask to test which events your callback function will handle. It passes these events in the `keychainEvent` parameter of the function `InvokeKCCallbackUPP` (page 11). For a description of Keychain-related events, see `Keychain Events Constants` (page 71).

Keychain Item Attribute Tag Constants

Represent tags that identify keychain item attribute values.

```
enum {
    kClassKCItemAttr = 'clas',
    kCreationDateKCItemAttr = 'cdat',
    kModDateKCItemAttr = 'mdat',
    kDescriptionKCItemAttr = 'desc',
    kCommentKCItemAttr = 'icmt',
    kCreatorKCItemAttr = 'crtr',
    kTypeKCItemAttr = 'type',
    kScriptCodeKCItemAttr = 'scrp',
    kLabelKCItemAttr = 'labl',
    kInvisibleKCItemAttr = 'invi',
    kNegativeKCItemAttr = 'nega',
    kCustomIconKCItemAttr = 'cusi',
    kAccountKCItemAttr = 'acct',
    kServiceKCItemAttr = 'svce',
    kGenericKCItemAttr = 'gena',
    kSecurityDomainKCItemAttr = 'sdmn',
    kServerKCItemAttr = 'srvr',
    kAuthTypeKCItemAttr = 'atyp',
    kPortKCItemAttr = 'port',
    kPathKCItemAttr = 'path',
    kVolumeKCItemAttr = 'vlme',
    kAddressKCItemAttr = 'addr',
    kSignatureKCItemAttr = 'ssig',
    kProtocolKCItemAttr = 'ptcl',
    kSubjectKCItemAttr = 'subj',
    kCommonNameKCItemAttr = 'cn  ',
    kIssuerKCItemAttr = 'issu',
    kSerialNumberKCItemAttr = 'snbr',
    kEMailKCItemAttr = 'mail',
    kPublicKeyHashKCItemAttr = 'hpky',
    kIssuerURLKCItemAttr = 'iurl',
    kEncryptKCItemAttr = 'encr',
    kDecryptKCItemAttr = 'decr',
    kSignKCItemAttr = 'sign',
    kVerifyKCItemAttr = 'veri',
    kWrapKCItemAttr = 'wrap',
    kUnwrapKCItemAttr = 'unwr',
    kStartDateKCItemAttr = 'sdat',
    kEndDateKCItemAttr = 'edat'
};
typedef FourCharCode KCItemAttr;
```

Constants

`kClassKCItemAttr`

Identifies the class attribute. You use this tag to set or get a value of type `KCItemClass` that indicates whether the item is an AppleShare, Internet, or generic password, or a certificate. See "KCPublicKeyHash" (page 66) for a description of possible values.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kCreationDateKCItemAttr`

Identifies the creation date attribute. You use this tag to set or get a value of type `UInt32` that indicates the date the item was created.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kModDateKCItemAttr`

Identifies the modification date attribute. You use this tag to set or get a value of type `UInt32` that indicates the last time the item was updated.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kDescriptionKCItemAttr`

Identifies the description attribute. You use this tag to set or get a value of type `string` that represents a user-visible string describing this item.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kCommentKCItemAttr`

Identifies the comment attribute. You use this tag to set or get a value of type `string` that represents a user-editable string containing comments for this item.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kCreatorKCItemAttr`

Identifies the creator attribute. You use this tag to set or get a value of type `OSType` that represents the item's creator.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kTypeKCItemAttr`

Identifies the type attribute. You use this tag to set or get a value of type `OSType` that represents the item's type.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kScriptCodeKCItemAttr`

Identifies the script code attribute. You use this tag to set or get a value of type `ScriptCode` that represents the script code for all strings.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kLabelKCItemAttr`

Identifies the label attribute. You use this tag to set or get a value of type `string` that represents a user-editable string containing the label for this item.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kInvisibleKCItemAttr`

Identifies the invisible attribute. You use this tag to set or get a value of type `Boolean` that indicates whether the item is invisible.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kNegativeKCItemAttr`

Identifies the negative attribute. You use this tag to set or get a value of type `Boolean` that indicates whether there is a valid password associated with this keychain item. This is useful if your application doesn't want a password for some particular service to be stored in the keychain, but prefers that it always be entered by the user. The item (typically invisible and with zero-length data) acts as a placeholder to say “don't use me.”

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kCustomIconKCItemAttr`

Identifies the custom icon attribute. You use this tag to set or get a value of type `Boolean` that indicates whether the item has an application-specific icon. To do this, you must also set the attribute value identified by the tag kTypeKCItemAttr to a file type for which there is a corresponding icon in the desktop database, and set the attribute value identified by the tag `kCreatorKCItemAttr` to an appropriate application creator type. If a custom icon corresponding to the item's type and creator can be found in the desktop database, it will be displayed by Keychain Access. Otherwise, default icons are used.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kAccountKCItemAttr`

Identifies the account attribute. You use this tag to set or get a value of type `Str63` that represents the user account. It also applies to generic and AppleShare passwords.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kServiceKCItemAttr`

Identifies the service attribute for a generic password. You use this tag to set or get a value of type `Str63` that represents the service.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kGenericKCItemAttr`

Identifies the generic attribute for a generic password. You use this tag to set or get a value of untyped bytes that represents a user-defined attribute.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kSecurityDomainKCItemAttr`

Identifies the security domain attribute for an internet password. You use this tag to set or get a value of type `Str63` that represents the Internet security domain.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kServerKCItemAttr`

Identifies the server attribute for an internet password. You use this tag to set or get a value of type `string` that represents the Internet server’s domain name or IP address.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kAuthTypeKCItemAttr`
Identifies the authentication type attribute for an internet password. You use this tag to set or get a value of type `KCAuthType` that represents the Internet authentication scheme.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kPortKCItemAttr`
Identifies the port attribute for an internet password. You use this tag to set or get a value of type `UInt16` that represents the Internet port.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kPathKCItemAttr`
Identifies the path attribute for an internet password. You use this tag to set or get a value of type `Str255` that represents the path.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kVolumeKCItemAttr`
Identifies the volume attribute for an AppleShare password. You use this tag to set or get a value of type `Str63` that represents the AppleShare volume.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kAddressKCItemAttr`
Identifies the address attribute for an AppleShare password. You use this tag to set or get a value of type `string` that represents the zone name, or the IP or domain name that represents the server address.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kSignatureKCItemAttr`
Identifies the server signature attribute for an AppleShare password. You use this tag to set or get a value of type `KCPublicKeyHash` (page 66) that represents the server signature block.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kProtocolKCItemAttr`
Identifies the protocol attribute for an AppleShare or internet password. You use this tag to set or get a value of type `KCProtocolType` that represents the Internet protocol.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kSubjectKCItemAttr`
Identifies the subject attribute for a certificate. You use this tag to set or get `DER`-encoded data that represents the subject distinguished name.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kCommonNameKCItemAttr`
Identifies the common name attribute for a certificate. You use this tag to set or get a `UTF8`-encoded string that represents the common name.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kIssuerKCItemAttr`
Identifies the issuer attribute for a certificate. You use this tag to set or get a `DER`-encoded data that represents the issuer distinguished name.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kSerialNumberKCItemAttr`
Identifies the serial number attribute for a certificate. You use this tag to set or get a `DER`-encoded data that represents the serial number.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kEMailKCItemAttr`
Identifies the email attribute for a certificate. You use this tag to set or get an `ASCII`-encoded string that represents the issuer's email address.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kPublicKeyHashKCItemAttr`
Identifies the public key hash attribute for a certificate. You use this tag to set or get a value of type `KCPublicKeyHash` (page 66) that represents the hash of the public key.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kIssuerURLKCItemAttr`
Identifies the issuer URL attribute for a certificate. You use this tag to set or get an `ASCII`-encoded string that represents the URL of the certificate issuer.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kEncryptKCItemAttr`
Identifies the encrypt attribute for a certificate or key. You use this tag to set or get a value of type `Boolean` that indicates whether the item can encrypt.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kDecryptKCItemAttr`
Identifies the decrypt attribute for a certificate or key. You use this tag to set or get a value of type `Boolean` that indicates whether the item can decrypt.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kSignKCItemAttr`
: Identifies the sign attribute for a certificate or key. You use this tag to set or get a value of type `Boolean` that indicates whether the item can sign.

  Available in Mac OS X v10.0 and later.

  Declared in `KeychainCore.h`.

`kVerifyKCItemAttr`
: Identifies the verify attribute for a certificate or key. You use this tag to set or get a value of type `Boolean` that indicates whether the item can verify.

  Available in Mac OS X v10.0 and later.

  Declared in `KeychainCore.h`.

`kWrapKCItemAttr`
: Identifies the wrap attribute for a certificate or key. You use this tag to set or get a value of type `Boolean` that indicates whether the item can wrap.

  Available in Mac OS X v10.0 and later.

  Declared in `KeychainCore.h`.

`kUnwrapKCItemAttr`
: Identifies the unwrap attribute for a certificate or key. You use this tag to set or get a value of type `Boolean` that indicates whether the item can unwrap.

  Available in Mac OS X v10.0 and later.

  Declared in `KeychainCore.h`.

`kStartDateKCItemAttr`
: Identifies the start date attribute for a certificate or key. You use this tag to set or get a value of type `UInt32` that indicates the start date.

  Available in Mac OS X v10.0 and later.

  Declared in `KeychainCore.h`.

`kEndDateKCItemAttr`
: Identifies the end date attribute for a certificate or key. You use this tag to set or get a value of type `UInt32` that indicates the end date.

  Available in Mac OS X v10.0 and later.

  Declared in `KeychainCore.h`.

Discussion
The `KCItemAttr` enumeration defines the Apple-defined tag constants that identify keychain item attribute values. Your application can use one of these tags in the `tag` field of the structure `KCAttribute` (page 63) to identify the keychain item attribute value you wish to set or retrieve. Your application can create application-defined tags of type `KCAttrType` (page 64).

Keychain Item Type Constants

Identify the type of keychain item.

```
enum {
    kCertificateKCItemClass = 'cert',
    kAppleSharePasswordKCItemClass = 'ashp',
    kInternetPasswordKCItemClass = 'inet',
    kGenericPasswordKCItemClass = 'genp'
};
typedef FourCharCode KCItemClass;
```

Constants

`kCertificateKCItemClass`
Specifies that the item is a digital certificate.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kAppleSharePasswordKCItemClass`
Specifies that the item is an AppleShare password.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kInternetPasswordKCItemClass`
Specifies that the item is an Internet password.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kGenericPasswordKCItemClass`
Specifies that the item is a generic password.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

Discussion

The `KCItemClass` enumeration defines constants your application can use to specify the type of the keychain item you wish to create, dispose, add, delete, update, copy, or locate. You pass a constant of this type to the functions `KCNewItem` (page 51), `KCReleaseItem` (page 52), `KCAddItem` (page 22), `KCDeleteItem` (page 28), `KCUpdateItem` (page 60), `KCCopyItem` (page 24), and `KCGetKeychain` (page 45). You can also use these constants with the tag constant `kClassKCItemAttr`, described in `Keychain Item Attribute Tag Constants` (page 75).

Keychain Protocol Type Constants

Identify the protocol to use in storing and retrieving Internet passwords.

```
enum {
    kKCProtocolTypeFTP = 'ftp ',
    kKCProtocolTypeFTPAccount = 'ftpa',
    kKCProtocolTypeHTTP = 'http',
    kKCProtocolTypeIRC = 'irc ',
    kKCProtocolTypeNNTP = 'nntp',
    kKCProtocolTypePOP3 = 'pop3',
    kKCProtocolTypeSMTP = 'smtp',
    kKCProtocolTypeSOCKS = 'sox ',
    kKCProtocolTypeIMAP = 'imap',
    kKCProtocolTypeLDAP = 'ldap',
    kKCProtocolTypeAppleTalk = 'atlk',
    kKCProtocolTypeAFP = 'afp ',
    kKCProtocolTypeTelnet = 'teln'
};
typedef FourCharCode KCProtocolType;
```

Constants

`kKCProtocolTypeFTP`
Specifies the File Transfer Protocol.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCProtocolTypeFTPAccount`
Specifies the File Transfer Protocol Account.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCProtocolTypeHTTP`
Specifies the HyperText Transfer Protocol.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCProtocolTypeIRC`
Specifies the Internet Relay Channel Protocol.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCProtocolTypeNNTP`
Specifies the Network News Transfer Protocol.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCProtocolTypePOP3`
Specifies the Post Office 3 Protocol.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCProtocolTypeSMTP`
Specifies the Simple Mail Transfer Protocol.
Available in Mac OS X v10.0 and later.
Declared in `KeychainCore.h`.

`kKCProtocolTypeSOCKS`
Specifies the Secure Proxy Server Protocol.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kKCProtocolTypeIMAP`
Specifies the Internet Message Access Protocol.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kKCProtocolTypeLDAP`
Specifies the Lightweight Directory Access Protocol.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kKCProtocolTypeAppleTalk`
Specifies the AppleTalk Protocol.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kKCProtocolTypeAFP`
Specifies the AppleTalk File Protocol.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kKCProtocolTypeTelnet`
Specifies the Telnet Protocol.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

Discussion
The `KCProtocolType` enumeration defines constants you can use to identify the type of authentication to use in storing and retrieving Internet passwords. You can pass a constant of this type in the `protocol` parameter of the functions `KCAddInternetPassword` (page 18), `KCAddInternetPasswordWithPath` (page 20), `KCFindInternetPassword` (page 34), and `KCFindInternetPasswordWithPath` (page 37).

Keychain Status Constants

Identify the keychain status.

```
enum {
    kUnlockStateKCStatus = 1,
    kRdPermKCStatus = 2,
    kWrPermKCStatus = 4
};
```

Constants
`kUnlockStateKCStatus`
If the bit specified by this mask is set (bit 0), the keychain is unlocked.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kRdPermKCStatus`

If the bit specified by this mask is set (bit 1), the keychain is unlocked with read permission.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

`kWrPermKCStatus`

If the bit specified by this mask is set (bit 2), the keychain is unlocked with write permission.

Available in Mac OS X v10.0 and later.

Declared in `KeychainCore.h`.

Result Codes

The most common result codes returned by Keychain Manager are listed below.

Result Code	Value	Description
`errKCNotAvailable`	-25291	Indicates that the Keychain Manager was not loaded. Available in Mac OS X v10.0 and later.
`errKCReadOnly`	-25292	Returned by the function `KCCopyItem` to indicate that the keychain file is read-only and cannot be edited. Available in Mac OS X v10.0 and later.
`errKCAuthFailed`	-25293	Returned by the function `KCUnlock` to indicate that the authentication failed (too many unsuccessful retries). Available in Mac OS X v10.0 and later.
`errKCNoSuchKeychain`	-25294	Returned by the functions `KCUnlock`, `KCSetDefaultKeychain`, `KCGetStatus`, and `KCGetIndKeychain` to indicate that the specified keychain was not found. Available in Mac OS X v10.0 and later.
`errKCInvalidKeychain`	-25295	Returned by the functions `KCUnlock`, `KCSetDefaultKeychain`, `KCGetStatus`, `KCGetKeychainName`, `KCChangeSettings`, and `KCCreateKeychain` to indicate that the keychain is not valid. Available in Mac OS X v10.0 and later.
`errKCDuplicateKeychain`	-25296	Returned by the function `KCCreateKeychain` to indicate that your application tried to create a keychain that already exists. Available in Mac OS X v10.0 and later.
`errKCDuplicateCallback`	-25297	Returned by the function `KCAddCallback` to indicate that your callback function was already registered. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
errKCInvalidCallback	-25298	Returned by the function KCRemoveCallback to indicate that the callback function was not previously registered. Available in Mac OS X v10.0 and later.
errKCDuplicateItem	-25299	Returned by the functions KCAddAppleSharePassword, KCAddInternetPassword, KCAddInternetPasswordWithPath, KCAddGenericPassword, and KCAddItem to indicate that you tried to add an existing keychain item to the keychain. Available in Mac OS X v10.0 and later.
errKCItemNotFound	-25300	Returned by the functions KCFindAppleSharePassword, KCFindInternetPassword, KCFindInternetPasswordWithPath, KCFindGenericPassword, KCFindNextItem, and KCFindFirstItem to indicate that no matching item was found. Available in Mac OS X v10.0 and later.
errKCBufferTooSmall	-25301	Returned by the functions KCFindAppleSharePassword, KCFindInternetPassword, KCFindInternetPasswordWithPath, KCFindGenericPassword, KCGetAttribute, KCGetData, and KCFindX509Certificates to indicate that the buffer was not large enough to contain the password data. Available in Mac OS X v10.0 and later.
errKCDataTooLarge	-25302	Returned by the functions KCAddAppleSharePassword, KCAddInternetPassword, KCAddInternetPasswordWithPath, KCAddGenericPassword, KCSetAttribute, and KCSetData to indicate that the data is too large. Available in Mac OS X v10.0 and later.
errKCNoSuchAttr	-25303	Returned by the functions KCSetAttribute, KCGetAttribute, and KCFindFirstItem to indicate that no such attribute exists. Available in Mac OS X v10.0 and later.
errKCInvalidItemRef	-25304	Returned by the functions KCSetAttribute, KCGetAttribute, KCSetData, KCGetData, KCAddItem, KCDeleteItem, KCUpdateItem, KCCopyItem, and KCGetKeychain to indicate that the keychain item reference is invalid. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
errKCInvalidSearchRef	-25305	Returned by the functions KCFindNextItem and KCReleaseSearch to indicate that the specified search reference is invalid. Available in Mac OS X v10.0 and later.
errKCNoSuchClass	-25306	Returned by the function KCCopyItem to indicate that the item class does not exist. Available in Mac OS X v10.0 and later.
errKCNoDefaultKeychain	-25307	Returned by the functions KCChangeSettings, KCSetDefaultKeychain, KCGetDefaultKeychain, KCAddAppleSharePassword, KCAddInternetPassword, KCAddInternetPasswordWithPath, KCAddGenericPassword, KCFindAppleSharePassword, KCFindInternetPassword, KCFindInternetPasswordWithPath, KCFindGenericPassword, KCCopyItem, KCAddItem, KCDeleteItem, KCUpdateItem, KCFindNextItem, KCFindFirstItem, and KCFindX509Certificates to indicate that there is no default keychain. Available in Mac OS X v10.0 and later.
errKCInteractionNotAllowed	-25308	Returned by the functions KCCreateKeychain, KCChangeSettings, KCUnlock, and KCGetData (the latter two only when the Unlock Dialog and Allow Access dialog boxes are needed) to indicate that there is no start-up keychain. Available in Mac OS X v10.0 and later.
errKCReadOnlyAttr	-25309	Returned by the function KCSetAttribute to indicate that the keychain item attribute is read-only. Available in Mac OS X v10.0 and later.
errKCWrongKCVersion	-25310	Indicates that the wrong version of Keychain Manager is installed to perform this operation. Available in Mac OS X v10.0 and later.
errKCKeySizeNotAllowed	-25311	Indicates that the key size is illegal. Available in Mac OS X v10.0 and later.
errKCNoStorageModule	-25312	Returned by functions that prompts the loading of the Keychain Manager to indicate that the storage module is not found. Available in Mac OS X v10.0 and later.
errKCNoCertificateModule	-25313	Returned when a function is required for a certificate and the certificate module is not found. Available in Mac OS X v10.0 and later.

Result Code	Value	Description
`errKCNoPolicyModule`	-25314	Returned when a function is required for a trust policy and the policy module is not found. Available in Mac OS X v10.0 and later.
`errKCInteractionRequired`	-25315	Returned by the function `KCUnlock` to indicate that user interaction is required for this operation. Available in Mac OS X v10.0 and later.
`errKCDataNotAvailable`	-25316	Indicates that the requested data is not available. Available in Mac OS X v10.0 and later.
`errKCDataNotModifiable`	-25317	Returned by the functions `KCSetData` and `KCGetData` to indicate that the data cannot be modified. Available in Mac OS X v10.0 and later.
`errKCCreateChainFailed`	-25318	Returned by the functions `KCChooseCertificate` and `KCFindX509Certificates` to indicate that the attempt to create a new keychain failed. Available in Mac OS X v10.0 and later.

Deprecated Keychain Manager Functions

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

Deprecated in Mac OS X v10.5

KCMakeKCRefFromFSSpec

Creates a keychain reference from a file specification record. (Deprecated in Mac OS X v10.5.)

Not Recommended

```
OSStatus KCMakeKCRefFromFSSpec (
   FSSpec *keychainFSSpec,
   KCRef *keychain
);
```

Parameters

keychainFSSpec

A pointer to a keychain file specification record.

keychain

On return, a pointer to a reference to the keychain specified by the file in the `keychainFSSpec` parameter.

Return Value

A result code. See "Keychain Manager Result Codes" (page 85).

Special Considerations

When you are finished with a keychain, you should call the function `KCReleaseKeychain` (page 53) to deallocate its memory. You should not use the keychain after its memory has been deallocated.

Version Notes

Available beginning with KeychainLib 2.0.

Availability

Available in CarbonLib 1.1 and later when KeychainLib 2.0 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

Use the `SecKeychainOpen` function in Keychain Services instead. If the keychain doesn't exist, use the `SecKeychainCreate` function in Keychain Services.

Declared In
`KeychainCore.h`

Document Revision History

This table describes the changes to *Keychain Manager Reference*.

Date	Notes
2005-07-07	Corrected a typo.
2003-10-01	Corrected description of KCSetDefaultKeychain function.
2003-09-01	Updated the EDD.
	Removed type definitions that are also defined in Keychain Services Reference to avoid cross reference conflicts.
2002-12-01	Updated the EDD.
	Reworded the description of the second parameter for the `KCCreateKeychain` function.
	Deprecated functions and provided references to replacement functions.
	Made changes to parameter descriptions of the following functions: `KCFindAppleSharePassword`, `KCFindInternetPassword`, `KCFindInternetPasswordWithPath` and `KCFindGenericPassword`.
	Made many minor wording changes.
2001-07-01	First version of this document.

Document Revision History

Index

N