Keychain Services Reference

Security > Authentication

2008-11-19

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

Simultaneously published in the United States and Canada.

Contents

Keychain Services Reference

Framework:	Security/Security.h
Declared in	SecACL.h SecAccess.h SecBase.h SecImportExport.h SecKey.h SecKeychain.h SecKeychainItem.h SecKeychainSearch.h SecTrustedApplication.h cssmapple.h cssmtype.h

Overview

Keychain Services is a programming interface that enables you to find, add, modify, and delete keychain items.

Functions by Task

Getting Information About Keychain Services and Types

`SecKeychainGetVersion` (page 38)
Determines the version of Keychain Services installed on the user's system.

`SecKeychainGetTypeID` (page 37)
Returns the unique identifier of the opaque type to which a `SecKeychainRef` object belongs.

`SecKeychainItemGetTypeID` (page 47)
Returns the unique identifier of the opaque type to which a `SecKeychainItemRef` object belongs.

`SecKeychainSearchGetTypeID` (page 56)
Returns the unique identifier of the opaque type to which a `SecKeychainSearchRef` object belongs.

`SecAccessGetTypeID` (page 16)
Returns the unique identifier of the opaque type to which a `SecAccessRef` object belongs.

`SecACLGetTypeID` (page 19)
Returns the unique identifier of the opaque type to which a `SecACLRef` object belongs.

`SecTrustedApplicationGetTypeID` (page 63)
Returns the unique identifier of the opaque type to which a `SecTrustedApplication` object belongs.

Creating and Deleting a Keychain

`SecKeychainCreate` (page 30)
Creates an empty keychain.

`SecKeychainDelete` (page 31)
Deletes one or more keychains from the default keychain search list, and removes the keychain itself if it is a file.

Managing Keychains

`SecKeychainOpen` (page 53)
Opens a keychain.

`SecKeychainSetDefault` (page 57)
Sets the default keychain.

`SecKeychainCopyDefault` (page 27)
Retrieves a pointer to the default keychain.

`SecKeychainGetStatus` (page 37)
Retrieves status information of a keychain.

`SecKeychainGetPath` (page 36)
Determines the path of a keychain.

`SecKeychainSetSettings` (page 59)
Changes the settings of a keychain.

`SecKeychainCopySettings` (page 29)
Obtains a keychain's settings.

Locking and Unlocking Keychains

`SecKeychainLock` (page 52)
Locks a keychain.

`SecKeychainLockAll` (page 53)
Locks all keychains belonging to the current user.

`SecKeychainUnlock` (page 61)
Unlocks a keychain.

Managing User Interaction

`SecKeychainSetUserInteractionAllowed` (page 60)
Enables or disables the user interface for Keychain Services functions that automatically display a user interface.

`SecKeychainGetUserInteractionAllowed` (page 38)
Indicates whether Keychain Services functions that normally display a user interaction are allowed to do so.

Managing Keychain Access

`SecKeychainSetAccess` (page 56)
Sets the application access for a keychain.

`SecKeychainCopyAccess` (page 26)
Retrieves the application access of a keychain.

Storing and Retrieving Passwords

`SecKeychainAddInternetPassword` (page 24)
Adds a new Internet password to a keychain.

`SecKeychainFindInternetPassword` (page 33)
Finds the first Internet password based on the attributes passed.

`SecKeychainAddGenericPassword` (page 23)
Adds a new generic password to a keychain.

`SecKeychainFindGenericPassword` (page 31)
Finds the first generic password based on the attributes passed.

Searching for Keychain Items

`SecKeychainSetSearchList` (page 59)
Specifies the list of keychains to use in the default keychain search list.

`SecKeychainCopySearchList` (page 29)
Retrieves a keychain search list.

`SecKeychainSearchCreateFromAttributes` (page 55)
Creates a search object matching a list of zero or more attributes.

`SecKeychainSearchCopyNext` (page 54)
Finds the next keychain item matching the given search criteria.

Creating and Deleting Keychain Items

`SecKeychainItemCreateFromContent` (page 43)
Creates a new keychain item from the supplied parameters.

`SecKeychainItemCreateCopy` (page 42)
Copies a keychain item from one keychain to another.

`SecKeychainItemDelete` (page 44)
Deletes a keychain item from the default keychain's permanent data store.

Exporting and Importing Keychain Items

`SecKeychainItemExport` (page 44)
Exports one or more certificates, keys, or identities.

`SecKeychainItemImport` (page 48)
Imports one or more certificates, keys, or identities and adds them to a keychain.

Managing Keychain Items

`SecKeychainItemCopyAttributesAndData` (page 39)
Retrieves the data and/or attributes stored in the given keychain item.

`SecKeychainItemModifyAttributesAndData` (page 50)
Updates an existing keychain item after changing its attributes or data.

`SecKeychainItemFreeAttributesAndData` (page 45)
Releases the memory used by the keychain attribute list and/or the keychain data retrieved in a call to `SecKeychainItemCopyAttributesAndData`.

`SecKeychainItemCopyContent` (page 41)
Copies the data and attributes stored in the given keychain item.

`SecKeychainItemModifyContent` (page 50)
Updates an existing keychain item after changing its attributes and/or data.

`SecKeychainItemFreeContent` (page 46)
Releases the memory used by the keychain attribute list and/or the keychain data retrieved in a call to the `SecKeychainItemCopyContent` (page 41) function.

`SecKeychainAttributeInfoForItemID` (page 26)
Obtains tags for all possible attributes of a given item class.

`SecKeychainFreeAttributeInfo` (page 34)
Releases the memory acquired by calling the `SecKeychainAttributeInfoForItemID` function.

`SecKeychainItemCopyKeychain` (page 42)
Returns the keychain object of a given keychain item.

`SecKeychainItemSetAccess` (page 51)
Sets the access of a given keychain item.

`SecKeychainItemCopyAccess` (page 39)
Copies the access of a given keychain item.

Creating an Access Object

`SecAccessCreate` (page 13)
Creates a new access object.

`SecAccessCreateFromOwnerAndACL` (page 15)
Creates a new access object using the owner and access control list you provide.

Managing Access Objects

`SecAccessCopyACLList` (page 12)
Retrieves all the access control list entries of a given access object.

`SecAccessCopySelectedACLList` (page 13)
Retrieves selected access control lists from a given access object.

`SecAccessGetOwnerAndACL` (page 15)
Retrieves the owner and the access control list of a given access object.

Managing Access Control List Objects

`SecACLCreateFromSimpleContents` (page 17)
Creates a new access control list entry from the application list, description, and prompt selector provided and adds it to an item's access object.

`SecACLRemove` (page 20)
Removes the specified access control list entry.

`SecACLCopySimpleContents` (page 17)
Returns the application list, description, and CSSM prompt selector for a given access control list entry.

`SecACLSetSimpleContents` (page 21)
Sets the application list, description, and prompt selector for a given access control list entry.

`SecACLGetAuthorizations` (page 19)
Retrieves the CSSM authorization tags of a given access control list entry.

`SecACLSetAuthorizations` (page 20)
Sets the CSSM authorization tags for a given access control list entry.

Managing Trusted Applications

`SecTrustedApplicationCopyData` (page 62)
Retrieves the data of a trusted application object.

`SecTrustedApplicationCreateFromPath` (page 62)
Creates a trusted application object based on the application specified by path.

`SecTrustedApplicationSetData` (page 63)
Sets the data of a given trusted application object.

Managing Preference Domains

`SecKeychainGetPreferenceDomain` (page 36)
Gets the current keychain preference domain.

`SecKeychainSetPreferenceDomain` (page 58)
Sets the keychain preference domain.

`SecKeychainCopyDomainDefault` (page 28)
Retrieves the default keychain from a specified preference domain.

`SecKeychainSetDomainDefault` (page 57)
Sets the default keychain for a specified preference domain.

`SecKeychainCopyDomainSearchList` (page 28)
Retrieves the keychain search list for a specified preference domain.

`SecKeychainSetDomainSearchList` (page 58)
Sets the keychain search list for a specified preference domain.

CSSM Bridge Functions

`SecKeychainGetCSPHandle` (page 35)
Returns the CSSM CSP handle for the given keychain object.

`SecKeychainGetDLDBHandle` (page 35)
Returns the CSSM database handle for a given keychain object.

`SecKeychainItemGetDLDBHandle` (page 47)
Returns the CSSM database handle for a given keychain item object.

`SecKeychainItemGetUniqueRecordID` (page 47)
Returns a CSSM unique record for the given keychain item object.

Adding and Removing Callbacks

`SecKeychainAddCallback` (page 22)
Registers your keychain event callback function

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

Functions

SecAccessCopyACLList

Retrieves all the access control list entries of a given access object.

```
OSStatus SecAccessCopyACLList (
   SecAccessRef accessRef,
   CFArrayRef *aclList
);
```

Parameters

accessRef
The access object from which to retrieve the information.

aclList
On return, a pointer to a reference of a newly created `CFArray` of `SecACLRef` instances. You should call the `CFRelease` function on this array when you are finished with it.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
An access object can have any number of access control list (ACL) entries for specific operations or sets of operations. To retrieve ACL entries for specific operations, use the `SecAccessCopySelectedACLList` (page 13) function.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecAccess.h`

SecAccessCopySelectedACLList

Retrieves selected access control lists from a given access object.

```
OSStatus SecAccessCopySelectedACLList (
   SecAccessRef accessRef,
   CSSM_ACL_AUTHORIZATION_TAG action,
   CFArrayRef *aclList
);
```

Parameters

accessRef

The access object from which to retrieve the information.

action

An access control list authorization tag; the function returns only those access control list entries that apply to the operation indicated by this tag.

aclList

On return, a pointer to the selected access control lists. Release this by calling the `CFRelease` function.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
An access object can have any number of access control list (ACL) entries for specific operations or sets of operations. To retrieve all the ACL entries for an access object, use the `SecAccessCopyACLList` (page 12) function.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecAccess.h`

SecAccessCreate

Creates a new access object.

```
OSStatus SecAccessCreate (
   CFStringRef descriptor,
   CFArrayRef trustedlist,
   SecAccessRef *accessRef
);
```

Parameters

descriptor

A `CFString` object representing the name of the keychain item as it should appear in security dialogs. Note that this is not necessarily the same name as appears for that item in the Keychain Access application.

trustedlist

A reference to an array of trusted application objects (values of type `SecTrustedApplicationRef`) specifying which applications should be allowed to access the item without triggering confirmation dialogs. Use the `SecTrustedApplicationCreateFromPath` (page 62) function to create trusted application objects. If you pass `NULL` for this parameter, the access control list is automatically set to the application creating the item. To set no applications, pass a `CFArrayRef` with no elements.

accessRef

On return, points to the new access object. Release this object by calling the `CFRelease` function when you no longer need it.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

Each protected keychain item (such as a password or private key) has an associated access object. The access object contains access control list (ACL) entries, which specify trusted applications and the operations for which those operations are trusted. When an application attempts to access a keychain item for a particular purpose (such as to sign a document), the system determines whether that application is trusted to access the item for that purpose. If it is trusted, then the application is given access and no user confirmation is required. If the application is not trusted, but there is an ACL entry for that operation, then the user is asked to confirm whether the application should be given access. If there is no ACL entry for that operation, then access is denied and it is up to the calling application to try something else or to notify the user.

This function creates an access object with three ACL entries: The first, referred to as *owner access*, determines who can modify the access object itself. By default, there are no trusted applications for owner access; the user is always prompted for permission if someone tries to change access controls. The second is for operations considered safe, such as encrypting data. This ACL entry applies to all applications. The third ACL entry is for operations that should be restricted, such as decrypting, signing, deriving keys, and exporting keys. This ACL entry applies to the trusted applications listed in the `trustedlist` parameter.

To retrieve all the ACL entries of an access object, use the `SecAccessCopyACLList` (page 12) function. To retrieve specific ACL entries, use the `SecAccessCopySelectedACLList` (page 13) function. To create a new ACL entry and add it to an access object, use `SecACLCreateFromSimpleContents` (page 17). To modify an existing ACL entry, use `SecACLSetSimpleContents` (page 21). To modify the operations for which an ACL entry is used, call the `SecACLSetAuthorizations` (page 20) function.

Because an ACL object is always associated with an access object, when you modify an ACL entry, you are modifying the access object as well. Therefore, there is no need for a separate function to write a modified ACL object back into the access object.

Availability

Available in Mac OS X v10.2 and later.

Declared In
SecAccess.h

SecAccessCreateFromOwnerAndACL

Creates a new access object using the owner and access control list you provide.

```
OSStatus SecAccessCreateFromOwnerAndACL (
   const CSSM_ACL_OWNER_PROTOTYPE *owner,
   uint32 aclCount,
   const CSSM_ACL_ENTRY_INFO *acls,
   SecAccessRef *accessRef
);
```

Parameters

owner
A pointer to a CSSM access control list owner.

aclCount
An unsigned 32-bit integer representing the number of items in the access control list.

acls
A pointer to the CSSM access control list.

accessRef
On return, points to the new access object. Release this by calling the `CFRelease` function.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
This function creates an access object from CSSM structures. You can use this function to create an access object for use with other Certificate, Key, and Trust API functions if you want to use CSSM to create the access control list. CSSM allows more complex access controls than you can construct with the Certificate, Key, and Trust API. For more information about the CSSM API, see *Common Security: CDSA and CSSM, version 2 (with corrigenda)* from The Open Group (http://www.opengroup.org/security/cdsa.htm).

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecAccess.h

SecAccessGetOwnerAndACL

Retrieves the owner and the access control list of a given access object.

```
OSStatus SecAccessGetOwnerAndACL (
   SecAccessRef accessRef,
   CSSM_ACL_OWNER_PROTOTYPE_PTR *owner,
   uint32 *aclCount,
   CSSM_ACL_ENTRY_INFO_PTR *acls
);
```

Parameters

accessRef

An access object from which to retrieve the owner and access control list.

owner

On return, a pointer to a CSSM access control list owner.

aclCount

On return, a pointer to an unsigned 32-bit integer representing the number of items in the access control list.

acls

On return, a pointer to the CSSM access control list.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
This function returns CSSM structures for use with CSSM API functions.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecAccess.h`

SecAccessGetTypeID

Returns the unique identifier of the opaque type to which a `SecAccessRef` object belongs.

```
CFTypeID SecAccessGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecAccessRef` (page 65) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecAccessRef` (page 65) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecAccess.h`

SecACLCopySimpleContents

Returns the application list, description, and CSSM prompt selector for a given access control list entry.

```
OSStatus SecACLCopySimpleContents (
   SecACLRef acl,
   CFArrayRef *applicationList,
   CFStringRef *description,
   CSSM_ACL_KEYCHAIN_PROMPT_SELECTOR *promptSelector
);
```

Parameters

acl

An ACL object that identifies the access control list entry from which you want information.

applicationList

On return, points to an array of `SecTrustedApplication` instances identifying applications that are allowed access to the keychain item without user confirmation. If this parameter returns `NULL`, then any application can use this item. If this parameter returns a valid pointer but the array is empty, then there are no trusted applications. Call `CFRelease` for this object when you no longer need it.

description

On return, the name of the keychain item that appears in the dialog box when the user is prompted for permission to use the item. Note that this name is not necessarily the same as the one displayed for the item by the Keychain Access application. Call `CFRelease` for this object when you no longer need it.

promptSelector

On return, points to the prompt selector flag for the given access control list entry. If the `CSSM_ACL_KEYCHAIN_PROMPT_REQUIRE_PASSPHRASE` bit is set, the user is prompted for the keychain password each time a non-trusted application attempts to access this item, even if the keychain is already unlocked.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

An access control list entry applies to a specific use or set of uses for a specific keychain item. The ACL object includes a list of trusted applications (see `SecTrustedApplicationCreateFromPath` (page 62)), the name of the keychain item as it appears in user prompts, the prompt selector flag, and a list of one or more operations to which this ACL object applies. Use the `SecACLGetAuthorizations` (page 19) function to get the list of operations for an ACL object.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecACL.h`

SecACLCreateFromSimpleContents

Creates a new access control list entry from the application list, description, and prompt selector provided and adds it to an item's access object.

```
OSStatus SecACLCreateFromSimpleContents (
   SecAccessRef access,
   CFArrayRef applicationList,
   CFStringRef description,
   const CSSM_ACL_KEYCHAIN_PROMPT_SELECTOR *promptSelector,
   SecACLRef *newAcl
);
```

Parameters

access

The access object to which to add the information.

applicationList

An array of trusted application objects (that is, `SecTrustedApplication` instances) identifying applications that are allowed access to the keychain item without user confirmation. Use the `SecTrustedApplicationCreateFromPath` (page 62) function to create trusted application objects. If you set this parameter to `NULL`, then any application can use this item. If you pass an empty array, then there are no trusted applications. Call `CFRelease` for this object when you no longer need it.

description

The human readable name to be used to refer to this item when the user is prompted.

promptSelector

A pointer to a prompt selector. If you set the `CSSM_ACL_KEYCHAIN_PROMPT_REQUIRE_PASSPHRASE` bit, the user is prompted for the keychain password each time a non-trusted application attempts to access this item, even if the keychain is already unlocked.

newAcl

On return, points to an access control list object, which is a reference to the new access control list entry.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

The ACL object returned by this function is a reference to an access control list (ACL) entry. The ACL entry includes a list of trusted applications (see `SecTrustedApplicationCreateFromPath` (page 62)), the name of the keychain item as it appears in user prompts, the prompt selector flag, and a list of one or more operations to which this ACL entry applies. By default, a new ACL entry applies to all operations (the CSSM authorization tag is set to `CSSM_ACL_AUTHORIZATION_ANY`). Use the `SecACLSetAuthorizations` (page 20) function to set the list of operations for an ACL object.

The system allows exactly one owner ACL entry in each access object. The `SecACLCreateFromSimpleContents` function fails if you attempt to add a second owner ACL. To change owner access controls, use the `SecAccessCopySelectedACLList` (page 13) function to find the owner ACL (that is, the only ACL with a CSSM authorization tag of `CSSM_ACL_AUTHORIZATION_CHANGE_ACL`) and the `SecACLSetSimpleContents` (page 21) function to change it as needed.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecACL.h`

SecACLGetAuthorizations

Retrieves the CSSM authorization tags of a given access control list entry.

```
OSStatus SecACLGetAuthorizations (
   SecACLRef acl,
   CSSM_ACL_AUTHORIZATION_TAG *tags,
   uint32 *tagCount
);
```

Parameters

acl

An ACL object that identifies the access control list entry from which you wish to retrieve the authorization tags.

tags

A pointer to an array of CSSM authorization tags. You must allocate this array before calling the function. On return, this array contains the authorization tags of the specified ACL entry.

tagCount

On input, points to the number of elements in the array you passed in the `tags` parameter. On return, points to the number of tags actually returned.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

An ACL object includes a list of trusted applications (see `SecTrustedApplicationCreateFromPath` (page 62)), the name of the keychain item as it appears in user prompts, the prompt selector flag, and a list of one or more operations to which this ACL object applies. Use this function to retrieve the list of operations for an ACL object. Use the `SecACLCopySimpleContents` (page 17) function to retrieve the other information.

The `SecACLGetAuthorizations` function returns an error if there are more tags to return than the number of elements you allocated in the `tags` array. A 20-element array should suffice for most purposes; however, you can test for the `errSecBufferTooSmall` error and increase the size of the array before calling the function again if necessary.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecACL.h`

SecACLGetTypeID

Returns the unique identifier of the opaque type to which a `SecACLRef` object belongs.

```
CFTypeID SecACLGetTypeID (
   void
);
```

Return Value

A value that identifies the opaque type of a `SecACLRef` (page 65) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecACLRef` (page 65) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecACL.h`

SecACLRemove

Removes the specified access control list entry.

```
OSStatus SecACLRemove (
   SecACLRef aclRef
);
```

Parameters

aclRef

An ACL object that identifies the access control list entry to remove.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
The system allows exactly one owner ACL entry in each access object. The `SecACLRemove` function fails if you attempt to remove the owner ACL entry. To change owner access controls, use the `SecAccessCopySelectedACLList` (page 13) function to find the owner ACL (that is, the only ACL with a CSSM authorization tag of `CSSM_ACL_AUTHORIZATION_CHANGE_ACL`) and the `SecACLSetSimpleContents` (page 21) function to change it as needed.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecACL.h`

SecACLSetAuthorizations

Sets the CSSM authorization tags for a given access control list entry.

```
OSStatus SecACLSetAuthorizations (
   SecACLRef acl,
   CSSM_ACL_AUTHORIZATION_TAG *tags,
   uint32 tagCount
);
```

Parameters

acl

An ACL object that identifies the access control list entry for which you wish to set authorization tags.

tags

An array of CSSM authorization tags.

tagCount

The number of tags in the CSSM authorization tag array.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
An ACL object includes a list of trusted applications (see `SecTrustedApplicationCreateFromPath` (page 62)), the name of the keychain item as it appears in user prompts, the prompt selector flag, and a list of one or more operations to which this ACL object applies. Use this function to set a list of operations for an ACL object, or set the `CSSM_ACL_AUTHORIZATION_ANY` tag to allow all operations. Use the `SecACLSetSimpleContents` (page 21) function to set the other information.

Because an ACL object is always associated with an access object, when you modify an ACL entry, you are modifying the access object as well. There is no need for a separate function to write a modified ACL object back into the access object.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecACL.h`

SecACLSetSimpleContents

Sets the application list, description, and prompt selector for a given access control list entry.

```
OSStatus SecACLSetSimpleContents (
   SecACLRef acl,
   CFArrayRef applicationList,
   CFStringRef description,
   const CSSM_ACL_KEYCHAIN_PROMPT_SELECTOR *promptSelector
);
```

Parameters

acl

An ACL object that identifies the access control list entry.

applicationList

An array of trusted application objects (that is, `SecTrustedApplication` instances) identifying applications that are allowed access to the keychain item without user confirmation. Use the `SecTrustedApplicationCreateFromPath` (page 62) function to create trusted application objects. If you set this parameter to `NULL`, then any application can use this item. If you pass an empty array, then there are no trusted applications. Call `CFRelease` for this object when you no longer need it.

description

The name of the keychain item that appears in the dialog box when the user is prompted for permission to use the item. Note that this name is not necessarily the same as the one displayed for the item by the Keychain Access application. Call `CFRelease` for this object when you no longer need it.

promptSelector

The prompt selector flag for the given access control list entry. Set the `CSSM_ACL_KEYCHAIN_PROMPT_REQUIRE_PASSPHRASE` bit to have the user prompted for the keychain password each time a non-trusted application attempts to access this item, even if the keychain is already unlocked.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
Because an ACL object is always associated with an access object, when you modify an ACL entry, you are modifying the access object as well. There is no need for a separate function to write a modified ACL object back into the access object.

Use the `SecACLGetAuthorizations` (page 19) function to get the list of operations for an ACL object.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecACL.h`

SecKeychainAddCallback

Registers your keychain event callback function

```
OSStatus SecKeychainAddCallback (
   SecKeychainCallback callbackFunction,
   SecKeychainEventMask eventMask,
   void *userContext
);
```

Parameters

callbackFunction

A pointer to your keychain event callback function, described in `SecKeychainCallback` (page 64).

eventMask

A bit mask indicating the keychain events of which your application wishes to be notified. Keychain Services 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.

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 this function with any particular call of your keychain event callback function.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
It is important to note that the current Foundation or Core Foundation run loop must be active when making this call or the callbacks are not registered. In multithreaded programs, the notifications are registered in the run loop of the thread calling `SecKeychainAddCallback`; therefore, delivery of notifications depends on the functioning of that thread's run loop. If that thread terminates, or is so busy that it doesn't operate its run loop in a timely manner, notifications will be delayed, and may eventually be dropped without any notification.

For that reason, it is inadvisable for your program to depend on delivery of notifications caused by your own actions (such as depending on receiving a deletion notification before updating a UI view) unless your program is multithreaded and can take notifications on a thread different from the one generating the events.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainAddGenericPassword

Adds a new generic password to a keychain.

```
OSStatus SecKeychainAddGenericPassword (
   SecKeychainRef keychain,
   UInt32 serviceNameLength,
   const char *serviceName,
   UInt32 accountNameLength,
   const char *accountName,
   UInt32 passwordLength,
   const void *passwordData,
   SecKeychainItemRef *itemRef
);
```

Parameters

keychain
: A reference to the keychain in which to store a generic password. Pass `NULL` to specify the default keychain.

serviceNameLength
: The length of the `serviceName` character string.

serviceName
: A UTF-8 encoded character string representing the service name.

accountNameLength
: The length of the `accountName` character string.

accountName
: A UTF-8 encoded character string representing the account name.

passwordLength
: The length of the `passwordData` buffer.

passwordData
: A pointer to a buffer containing the password data to be stored in the keychain. Before calling this function, allocate enough memory for the buffer to hold the data you want to store.

itemRef
: On return, a pointer to a reference to the new keychain item. Pass `NULL` if you don't want to obtain this object. You must allocate the memory for this pointer. The memory that this pointer occupies must be released by calling the `CFRelease` function when finished with it.

Return Value
A result code. See "Keychain Services Result Codes" (page 99). The result code `errSecNoDefaultKeychain` indicates that no default keychain could be found. The result code `errSecDuplicateItem` indicates that you tried to add a password that already exists in the keychain. The result code `errSecDataTooLarge` indicates that you tried to add more data than is allowed for a structure of this type.

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

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

This function sets the initial access rights for the new keychain item so that the application creating the item is given trusted access.

This function automatically calls the function `SecKeychainUnlock` (page 61) to display the Unlock Keychain dialog box if the keychain is currently locked.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainAddInternetPassword

Adds a new Internet password to a keychain.

```
OSStatus SecKeychainAddInternetPassword (
   SecKeychainRef keychain,
   UInt32 serverNameLength,
   const char *serverName,
   UInt32 securityDomainLength,
   const char *securityDomain,
   UInt32 accountNameLength,
   const char *accountName,
   UInt32 pathLength,
   const char *path,
   UInt16 port,
   SecProtocolType protocol,
   SecAuthenticationType authenticationType,
   UInt32 passwordLength,
   const void *passwordData,
   SecKeychainItemRef *itemRef
);
```

Parameters

keychain

A reference to the keychain in which to store an Internet password. Pass `NULL` to specify the user's default keychain.

serverNameLength

The length of the `serverName` character string.

serverName
A UTF-8 encoded character string representing the server name.

securityDomainLength
The length of the `securityDomain` character string.

securityDomain
A UTF-8 encoded character string representing the security domain. This parameter is optional. Pass NULL if the protocol does not require it.

accountNameLength
The length of the `accountName` character string.

accountName
A UTF-8 encoded character string representing the account name.

pathLength
The length of the `path` character string.

path
A UTF-8 encoded character string representing the path.

port
The TCP/IP port number. If no specific port number is associated with this password, pass `0`.

protocol
The protocol associated with this password. See "Keychain Protocol Type Constants" (page 94) for a description of possible values.

authenticationType
The authentication scheme used. See "Keychain Authentication Type Constants" (page 74) for a description of possible values. Pass the constant `kSecAuthenticationTypeDefault`, to specify the default authentication scheme.

passwordLength
The length of the `passwordData` buffer.

passwordData
A pointer to a buffer containing the password data to be stored in the keychain.

itemRef
On return, a pointer to a reference to the new keychain item. Pass `NULL` if you don't want to obtain this object. You must allocate the memory for this pointer. The memory that this pointer occupies must be released by calling the `CFRelease` function when finished with it.

Return Value
A result code. See "Keychain Services Result Codes" (page 99). The result code `errSecNoDefaultKeychain` indicates that no default keychain could be found. The result code `errSecDuplicateItem` indicates that you tried to add a password that already exists in the keychain. The result code `errSecDataTooLarge` indicates that you tried to add more data than is allowed for a structure of this type.

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

This function sets the initial access rights for the new keychain item so that the application creating the item is given trusted access.

This function automatically calls the function `SecKeychainUnlock` (page 61) to display the Unlock Keychain dialog box if the keychain is currently locked.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
ImageClient

Declared In
`SecKeychain.h`

SecKeychainAttributeInfoForItemID

Obtains tags for all possible attributes of a given item class.

```
OSStatus SecKeychainAttributeInfoForItemID (
   SecKeychainRef keychain,
   UInt32 itemID,
   SecKeychainAttributeInfo **info
);
```

Parameters

keychain
A keychain object.

itemID
The relation identifier of the item tags. An *itemID* is a `CSSM_DB_RECORDTYPE` type as defined in `cssmtype.h`.

info
On return, a pointer to the keychain attribute information. Your application should call the `SecKeychainFreeAttributeInfo` (page 34) function to release this structure when done with it.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
This call returns more attributes than are supported by the old style Keychain API and passing them into older calls yields an invalid attribute error. The recommended call to retrieve the attribute values is the `SecKeychainItemCopyAttributesAndData` (page 39) function.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainCopyAccess

Retrieves the application access of a keychain.

```
OSStatus SecKeychainCopyAccess (
   SecKeychainRef keychain,
   SecAccessRef *access
);
```

Parameters

keychain

A reference to the keychain from which to copy the access object. Pass `NULL` to specify the default keychain.

access

A pointer to an access object. On return, this points to the access object of the specified keychain. See "Managing Access Objects" (page 11) for information on manipulating access objects.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Special Considerations

Although this function is available in Mac OS X v10.2, it was unimplemented before Mac OS X v10.3 and returned an `unimpErr` error code if called.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecKeychain.h`

SecKeychainCopyDefault

Retrieves a pointer to the default keychain.

```
OSStatus SecKeychainCopyDefault (
   SecKeychainRef *keychain
);
```

Parameters

keychain

On return, a pointer to the default keychain object. The memory that this pointer occupies must be released by calling the `CFRelease` function when finished with it.

Return Value

A result code. See "Keychain Services Result Codes" (page 99). The result code `errSecNoDefaultKeychain` indicates that there is no default keychain.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

SSLSample

Declared In

`SecKeychain.h`

SecKeychainCopyDomainDefault

Retrieves the default keychain from a specified preference domain.

```
OSStatus SecKeychainCopyDomainDefault (
   SecPreferencesDomain domain,
   SecKeychainRef *keychain
);
```

Parameters

domain

The preference domain from which you wish to retrieve the default keychain. See "Keychain Preference Domain Constants" (page 93) for possible domain values.

keychain

On return, a pointer to the keychain object of the default keychain in the specified preference domain.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

A preference domain is a set of security-related preferences, such as the default keychain and the current keychain search list. Use this function if you want to retrieve the default keychain for a specific preference domain. Use the `SecKeychainCopyDefault` (page 27) function if you want the default keychain for the current preference domain. See the `SecKeychainSetPreferenceDomain` (page 58) function for a discussion of current and default preference domains.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`SecKeychain.h`

SecKeychainCopyDomainSearchList

Retrieves the keychain search list for a specified preference domain.

```
OSStatus SecKeychainCopyDomainSearchList (
   SecPreferencesDomain domain,
   CFArrayRef *searchList
);
```

Parameters

domain

The preference domain from which you wish to retrieve the keychain search list. See "Keychain Preference Domain Constants" (page 93) for possible domain values.

searchList

On return, a pointer to the keychain search list of the specified preference domain.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion
A preference domain is a set of security-related preferences, such as the default keychain and the current keychain search list. Use this function if you want to retrieve the keychain search list for a specific preference domain. Use the `SecKeychainCopySearchList` (page 29) function if you want the keychain search list for the current preference domain. See the `SecKeychainSetPreferenceDomain` (page 58) function for a discussion of current and default preference domains.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SecKeychain.h`

SecKeychainCopySearchList

Retrieves a keychain search list.

```
OSStatus SecKeychainCopySearchList (
   CFArrayRef *searchList
);
```

Parameters

searchList

The returned keychain search list. When finished with the array, you must call `CFRelease` to release the memory.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainCopySettings

Obtains a keychain's settings.

```
OSStatus SecKeychainCopySettings (
   SecKeychainRef keychain,
   SecKeychainSettings *outSettings
);
```

Parameters

keychain

A reference to the keychain from which to copy its settings.

outSettings

On return, a pointer to a keychain settings structure. Since this structure is versioned, you must allocate the memory for it and fill in the version of the structure before passing it to the function.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainCreate

Creates an empty keychain.

```
OSStatus SecKeychainCreate (
   const char *pathName,
   UInt32 passwordLength,
   const void *password,
   Boolean promptUser,
   SecAccessRef initialAccess,
   SecKeychainRef *keychain
);
```

Parameters

pathName
A constant character string representing the POSIX path indicating where to store the keychain.

passwordLength
An unsigned 32-bit integer representing the length of the buffer pointed to by `password`. Pass `0` if the value of `password` is `NULL` and the value of `promptUser` is `TRUE`.

password
A pointer to the buffer containing the password which is used to protect the new keychain. The password must be in canonical UTF8 encoding. Pass `NULL` if the value of `passwordLength` is `0` and the value of `promptUser` is `TRUE`.

promptUser
A Boolean value representing whether to display a password dialog to the user. Set this value to `TRUE` to display a password dialog or `FALSE` otherwise. If you pass `TRUE`, any values passed for `passwordLength` and `password` are ignored, and a dialog for the user to enter a password is presented.

initialAccess
An access object indicating the initial access rights for the keychain. A keychain's access rights determine which applications have permission to use the keychain. You may pass `NULL` for the standard access rights.

keychain
On return, a pointer to a keychain object. The memory that the keychain object pointer occupies must be released by calling `CFRelease` when you are finished with it. Pass `NULL` if you do not need the pointer to the keychain object returned.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

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

The system ensures that a default keychain is created for the user at login, thus, in most cases, you do not need to call this function yourself. Users can create additional keychains, or change the default, by using the Keychain Access application. However, a missing default keychain is not recreated automatically, and you may receive an `errSecNoDefaultKeychain` error from other functions if a default keychain does not exist. In that case, you can use this function followed by `SecKeychainSetDefault` (page 57), to create a new default keychain. You can also call this function to create a private temporary keychain for your application's use, in cases where no user interaction can occur.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainDelete

Deletes one or more keychains from the default keychain search list, and removes the keychain itself if it is a file.

```
OSStatus SecKeychainDelete (
   SecKeychainRef keychainOrArray
);
```

Parameters

`keychainOrArray`

A single keychain object or a reference to an array of keychains you wish to delete. To delete more than one keychain, create a `CFArray` of keychain references (type `SecKeychainRef`) and pass a reference to the array. In Mac OS X v10.3 and later, passing `NULL` to this parameter returns an `errSecInvalidKeychain` error code.

In Mac OS X v10.2, this parameter was named `keychain` and only took a single keychain object. Passing `NULL` to this parameter deleted the user's default keychain.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
The keychain may be a file stored locally, a smart card, or retrieved from a network server using non-file-based database protocols. This function deletes the keychain only if it is a local file.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainFindGenericPassword

Finds the first generic password based on the attributes passed.

```
OSStatus SecKeychainFindGenericPassword (
   CFTypeRef keychainOrArray,
   UInt32 serviceNameLength,
   const char *serviceName,
   UInt32 accountNameLength,
   const char *accountName,
   UInt32 *passwordLength,
   void **passwordData,
   SecKeychainItemRef *itemRef
);
```

Parameters

keychainOrArray

A reference to an array of keychains to search, a single keychain, or `NULL` to search the user's default keychain search list.

serviceNameLength

The length of the `serviceName` character string.

serviceName

A UTF-8 encoded character string representing the service name.

accountNameLength

The length of the `accountName` character string.

accountName

A UTF-8 encoded character string representing the account name.

passwordLength

On return, the length of the buffer pointed to by `passwordData`.

passwordData

On return, a pointer to a buffer that holds the password data. Pass `NULL` if you want to obtain the item object but not the password data. In this case, you must also pass `NULL` in the `passwordLength` parameter. You should use the `SecKeychainItemFreeContent` (page 46) function to free the memory pointed to by this parameter.

itemRef

On return, a pointer to the item object of the generic password. Pass `NULL` if you don't want to obtain this object.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

This function finds the first generic password item that matches the attributes you provide. Most attributes are optional; you should pass only as many as you need to narrow the search sufficiently for your application's intended use. This function optionally returns a reference to the found item.

This function decrypts the password before returning it to you. If the calling application is not in the list of trusted applications, the user is prompted before access is allowed. If the access controls for this item do not allow decryption, the function returns the `errSecAuthFailed` result code.

This function automatically calls the function `SecKeychainUnlock` (page 61) to display the Unlock Keychain dialog box if the keychain is currently locked.

Availability

Available in Mac OS X v10.2 and later.

Declared In
SecKeychain.h

SecKeychainFindInternetPassword

Finds the first Internet password based on the attributes passed.

```
OSStatus SecKeychainFindInternetPassword (
   CFTypeRef keychainOrArray,
   UInt32 serverNameLength,
   const char *serverName,
   UInt32 securityDomainLength,
   const char *securityDomain,
   UInt32 accountNameLength,
   const char *accountName,
   UInt32 pathLength,
   const char *path,
   UInt16 port,
   SecProtocolType protocol,
   SecAuthenticationType authenticationType,
   UInt32 *passwordLength,
   void **passwordData,
   SecKeychainItemRef *itemRef
);
```

Parameters

keychainOrArray
: A reference to an array of keychains to search, a single keychain or `NULL` to search the user's default keychain search list.

serverNameLength
: The length of the `serverName` character string.

serverName
: A UTF-8 encoded character string representing the server name.

securityDomainLength
: The length of the `securityDomain` character string.

securityDomain
: A UTF-8 encoded character string representing the security domain. This parameter is optional, as not all protocols require it. Pass `NULL` if it is not required.

accountNameLength
: The length of the `accountName` character string.

accountName
: A UTF-8 encoded character string representing the account name.

pathLength
: The length of the `path` character string.

path
: A UTF-8 encoded character string representing the path.

port
: The TCP/IP port number. Pass `0` to ignore the port number.

protocol

The protocol associated with this password. See "Keychain Protocol Type Constants" (page 94) for a description of possible values.

authenticationType

The authentication scheme used. See "Keychain Authentication Type Constants" (page 74) for a description of possible values. Pass the constant `kSecAuthenticationTypeDefault`, to specify the default authentication scheme.

passwordLength

On return, the length of the buffer pointed to by `passwordData`.

passwordData

On return, a pointer to a buffer containing the password data. Pass `NULL` if you want to obtain the item object but not the password data. In this case, you must also pass `NULL` in the `passwordLength` parameter. You should use the `SecKeychainItemFreeContent` (page 46) function to free the memory pointed to by this parameter.

itemRef

On return, a pointer to the item object of the Internet password. Pass `NULL` if you don't want to obtain this object.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
This function finds the first Internet password item that matches the attributes you provide. This function optionally returns a reference to the found item.

This function decrypts the password before returning it to you. If the calling application is not in the list of trusted applications, the user is prompted before access is allowed. If the access controls for this item do not allow decryption, the function returns the `errSecAuthFailed` result code.

This function automatically calls the function `SecKeychainUnlock` (page 61) to display the Unlock Keychain dialog box if the keychain is currently locked.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
ImageClient

Declared In
`SecKeychain.h`

SecKeychainFreeAttributeInfo

Releases the memory acquired by calling the `SecKeychainAttributeInfoForItemID` function.

```
OSStatus SecKeychainFreeAttributeInfo (
   SecKeychainAttributeInfo *info
);
```

Parameters

info

A pointer to the keychain attribute information to release.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychain.h

SecKeychainGetCSPHandle

Returns the CSSM CSP handle for the given keychain object.

```
OSStatus SecKeychainGetCSPHandle (
   SecKeychainRef keychain,
   CSSM_CSP_HANDLE *cspHandle
);
```

Parameters

keychain
A keychain object.

cspHandle
On return, a pointer to the CSSM CSP handle for the given keychain. The handle is valid until the keychain object is released.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychain.h

SecKeychainGetDLDBHandle

Returns the CSSM database handle for a given keychain object.

```
OSStatus SecKeychainGetDLDBHandle (
   SecKeychainRef keychain,
   CSSM_DL_DB_HANDLE *dldbHandle
);
```

Parameters

keychain
A keychain object.

dldbHandle
On return, a pointer to the CSSM database handle for the given keychain. The handle is valid until the keychain object is released.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainGetPath

Determines the path of a keychain.

```
OSStatus SecKeychainGetPath (
   SecKeychainRef keychain,
   UInt32 *ioPathLength,
   char *pathName
);
```

Parameters

keychain
A reference to a keychain whose path you wish to obtain.

ioPathLength
On input, a pointer to the size of the character string `pathName`. On return, the size of `pathName` without the zero termination.

pathName
On input, a pointer to a buffer that you have allocated. On output, the buffer contains the POSIX path of the keychain as a UTF-8 encoded string. The function returns `errSecBufferTooSmall` if the provided buffer is too small.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
SSLSample

Declared In
`SecKeychain.h`

SecKeychainGetPreferenceDomain

Gets the current keychain preference domain.

```
OSStatus SecKeychainGetPreferenceDomain (
   SecPreferencesDomain *domain
);
```

Parameters

domain
On return, a pointer to the keychain preference domain. See "Keychain Preference Domain Constants" (page 93) for possible domain values.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
A preference domain is a set of security-related preferences, such as the default keychain and the current keychain search list. The default preference domain for system daemons (that is, for daemons running in the root session) is the system domain. The default preference domain for all other programs is the user domain. Use the `SecKeychainSetPreferenceDomain` (page 58) function to change the preference domain.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SecKeychain.h`

SecKeychainGetStatus

Retrieves status information of a keychain.

```
OSStatus SecKeychainGetStatus (
   SecKeychainRef keychain,
   SecKeychainStatus *keychainStatus
);
```

Parameters

keychain
A keychain object of the keychain whose status you wish to determine for the user session. Pass `NULL` to obtain the status of the default keychain.

keychainStatus
On return, a pointer to the status of the specified keychain. See "Keychain Status Masks" (page 98) for valid status constants.

Return Value
A result code. See "Keychain Services Result Codes" (page 99). The result code `errSecNoSuchKeychain` indicates that the specified keychain could not be found. The result code `errSecInvalidKeychain` indicates that the specified keychain is invalid.

Discussion
This function retrieves the status of a specified keychain. You can use this function to determine if the keychain is unlocked, readable, or writable. Note that the lock status of a keychain can change at any time due to user or system activity. Because the system automatically prompts the user to unlock a keychain when necessary, you do not usually have to worry about the lock status of a keychain. If you do need to track the lock status of a keychain, use the `SecKeychainAddCallback` (page 22) function to register for keychain notifications.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainGetTypeID

Returns the unique identifier of the opaque type to which a `SecKeychainRef` object belongs.

```
CFTypeID SecKeychainGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecKeychainRef` (page 68) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecKeychainRef` (page 68) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainGetUserInteractionAllowed

Indicates whether Keychain Services functions that normally display a user interaction are allowed to do so.

```
OSStatus SecKeychainGetUserInteractionAllowed (
   Boolean *state
);
```

Parameters

state

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

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainGetVersion

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

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

Parameters

returnVers

On return, a pointer to the version number of Keychain Services installed on the current system. See "Keychain Settings Version" (page 98) for a list of values.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

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

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainItemCopyAccess

Copies the access of a given keychain item.

```
OSStatus SecKeychainItemCopyAccess (
   SecKeychainItemRef itemRef,
   SecAccessRef *access
);
```

Parameters

itemRef

A reference to a keychain item.

access

On return, points to the keychain item's access object. Release this object by calling the `CFRelease` function.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
Each protected keychain item (such as a password or private key) has an associated access object. The access object contains access control list (ACL) entries, which specify trusted applications and the operations for which those operations are trusted. You can use this function together with the `SecKeychainItemSetAccess` (page 51) function to copy access controls from one keychain item to another. You can use the functions in the section "Managing Access Control List Objects" (page 11) to modify the contents of an access object.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainItem.h`

SecKeychainItemCopyAttributesAndData

Retrieves the data and/or attributes stored in the given keychain item.

```
OSStatus SecKeychainItemCopyAttributesAndData (
   SecKeychainItemRef itemRef,
   SecKeychainAttributeInfo *info,
   SecItemClass *itemClass,
   SecKeychainAttributeList **attrList,
   UInt32 *length,
   void **outData
);
```

Parameters

itemRef

A reference to the keychain item from which you wish to retrieve data or attributes.

info

A pointer to a list of tags of attributes to retrieve.

itemClass

A pointer to the item's class. You should pass `NULL` if not required. See "Keychain Item Class Constants" (page 87) for valid constants.

attrList

On input, the list of attributes in this item to get; on output the attributes are filled in. You should call the function `SecKeychainItemFreeAttributesAndData` (page 45) when you no longer need the attributes and data.

length

On return, a pointer to the actual length of the data.

outData

A pointer to a buffer containing the data in this item. Pass `NULL` if not required. You should call the function `SecKeychainItemFreeAttributesAndData` (page 45) when you no longer need the attributes and data.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

This function returns the data and attributes of a specific keychain item. You can use the `SecKeychainSearchCopyNext` (page 54) function to search for a keychain item if you don't already have the item's reference object. To find and obtain data from a password keychain item, use the `SecKeychainFindInternetPassword` (page 33) or `SecKeychainFindGenericPassword` (page 31) function.

You should pair the `SecKeychainItemCopyAttributesAndData` function with the `SecKeychainItemModifyAttributesAndData` (page 50) function, as these functions handle more attributes than are support by the old Keychain Manager and passing them into older calls yields an invalid attribute error. Use the functions `SecKeychainItemModifyContent` (page 50) and `SecKeychainItemCopyContent` (page 41) when dealing with older Keychain Manager functions.

If the keychain item data is encrypted, this function decrypts the data before returning it to you. If the calling application is not in the list of trusted applications, the user is prompted before access is allowed. If the access controls for this item do not allow decryption, the function returns the `errSecAuthFailed` result code.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecKeychainItem.h`

SecKeychainItemCopyContent

Copies the data and attributes stored in the given keychain item.

```
OSStatus SecKeychainItemCopyContent (
   SecKeychainItemRef itemRef,
   SecItemClass *itemClass,
   SecKeychainAttributeList *attrList,
   UInt32 *length,
   void **outData
);
```

Parameters

itemRef

A reference to the keychain item to modify.

itemClass

A pointer to the item's class. You should pass `NULL` if it is not required. See "Keychain Item Class Constants" (page 87) for valid constants.

attrList

A pointer to the list of attributes to get in this item on input; on output the attributes are filled in. You must call `SecKeychainItemFreeContent` (page 46) when you no longer need the attributes and data.

length

On return, the length of the buffer pointed to by the *outData* parameter.

outData

On return, a pointer to a buffer containing the data in this item. You must call `SecKeychainItemFreeContent` (page 46) when you no longer need the attributes and data.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

This function returns the data and attributes of a specific keychain item. You can use the `SecKeychainSearchCopyNext` (page 54) function to search for a keychain item if you don't already have the item's reference object. To find and obtain data from a password keychain item, use the `SecKeychainFindInternetPassword` (page 33) or `SecKeychainFindGenericPassword` (page 31) function.

You should pair the `SecKeychainItemModifyContent` (page 50) function with the `SecKeychainItemCopyContent` function when dealing with older Keychain Manager functions. The `SecKeychainItemCopyAttributesAndData` (page 39) and `SecKeychainItemModifyAttributesAndData` (page 50) functions handle more attributes than are support by the old Keychain Manager; however, passing them into older calls yields an invalid attribute error.

If the keychain item data is encrypted, this function decrypts the data before returning it to you. If the calling application is not in the list of trusted applications, the user is prompted before access is allowed. If the access controls for this item do not allow decryption, the function returns the `errSecAuthFailed` result code.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

ImageClient

Declared In
SecKeychainItem.h

SecKeychainItemCopyKeychain

Returns the keychain object of a given keychain item.

```
OSStatus SecKeychainItemCopyKeychain (
   SecKeychainItemRef itemRef,
   SecKeychainRef *keychainRef
);
```

Parameters

itemRef

A keychain item object.

keychainRef

On return, a pointer to a keychain object referencing the given keychain item. Release this by calling the CFRelease function.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychainItem.h

SecKeychainItemCreateCopy

Copies a keychain item from one keychain to another.

```
OSStatus SecKeychainItemCreateCopy (
   SecKeychainItemRef itemRef,
   SecKeychainRef destKeychainRef,
   SecAccessRef initialAccess,
   SecKeychainItemRef *itemCopy
);
```

Parameters

itemRef

A reference to the keychain item to copy.

destKeychainRef

A reference to the keychain in which to insert the copied keychain item. Pass NULL to specify the default keychain.

initialAccess

The initial access for the copied keychain item. Use the SecAccessCreate (page 13) function to create an access object or the SecKeychainItemCopyAccess (page 39) function to copy an access object from another keychain item. If you pass NULL for this parameter, the access defaults to the application creating the item.

itemCopy

On return, a pointer to a copy of the keychain item referenced by the `itemRef` parameter. You must release this object by calling the `CFRelease` function.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainItem.h`

SecKeychainItemCreateFromContent

Creates a new keychain item from the supplied parameters.

```
OSStatus SecKeychainItemCreateFromContent (
   SecItemClass itemClass,
   SecKeychainAttributeList *attrList,
   UInt32 length,
   const void *data,
   SecKeychainRef keychainRef,
   SecAccessRef initialAccess,
   SecKeychainItemRef *itemRef
);
```

Parameters

itemClass

A constant identifying the class of item to create. See "Keychain Item Class Constants" (page 87) for valid constants.

attrList

A pointer to the list of attributes for the item to create.

length

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

data

A pointer to a buffer containing the data to store.

keychainRef

A reference to the keychain in which to add the item. Pass `NULL` to specify the default keychain.

initialAccess

An access object for this keychain item. Use the `SecAccessCreate` (page 13) function to create an access object or the `SecKeychainItemCopyAccess` (page 39) function to copy an access object from another keychain item. If you pass `NULL` for this parameter, the access defaults to the application creating the item.

itemRef

On return, a pointer to a reference to the newly created keychain item. This parameter is optional. When the item object is no longer required, call `CFRelease` to deallocate memory occupied by the item.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
Each item stored in the keychain contains data (such as a certificate), which is indexed by the item's attributes. Use this function to create a keychain item from its attributes and data. To create keychain items that hold passwords, use the `SecKeychainAddInternetPassword` (page 24) or `SecKeychainAddGenericPassword` (page 23) functions.

A `SecKeychainItemRef` object for a certificate that is stored in a keychain can be safely cast to a `SecCertificateRef` for use with the Certificate, Key, and Trust API.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainItem.h`

SecKeychainItemDelete

Deletes a keychain item from the default keychain's permanent data store.

```
OSStatus SecKeychainItemDelete (
   SecKeychainItemRef itemRef
);
```

Parameters

itemRef

A keychain item object of the item to delete. Use the `CFRelease` function when you are completely finished with this item.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
If the keychain item has not previously been added to the keychain, this function does nothing and returns `noErr`.

Do not delete a keychain item and recreate it in order to modify it; instead, use the `SecKeychainItemModifyContent` (page 50) or `SecKeychainItemModifyAttributesAndData` (page 50) function to modify an existing keychain item. When you delete a keychain item, you lose any access controls and trust settings added by the user or by other applications.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainItem.h`

SecKeychainItemExport

Exports one or more certificates, keys, or identities.

```
OSStatus SecKeychainItemExport (
   CFTypeRef keychainItemOrArray,
   SecExternalFormat outputFormat,
   SecItemImportExportFlags flags,
   const SecKeyImportExportParameters *keyParams,
   CFDataRef *exportedData
);
```

Parameters

keychainItemOrArray

The keychain item or items to export. You can export only the following types of keychain items: *SecCertificateRef*, `SecKeyRef`, and `SecIdentityRef`. If you are exporting exactly one item, you can specify a `SecKeychainItemRef` object. Otherwise this parameter is a `CFArrayRef` object containing a number of items of type `SecKeychainItemRef`.

outputFormat

The format of the external representation of the item. Set this parameter to `kSecFormatUnknown` to use the default for that item type. Possible values for this parameter and default values are enumerated in "Keychain Item Import/Export Formats" (page 90).

flags

A flag indicating whether the exported item should have PEM armour. PEM armour refers to a way of expressing binary data as an ASCII string so that it can be transferred over text-only channels such as email. Set this flag to `kSecItemPemArmour` if you want PEM armouring.

keyParams

A pointer to a structure containing a set of input parameters for the function. If no key items are being exported, these parameters are optional and you can set the `keyParams` parameter to `NULL`.

exportedData

On return, points to the external representation of the keychain item or items.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

This function works only with keys, certificates, and identities. An identity is the combination of a certificate and its associated private key. Although public keys are commonly stored in certificates, they can be stored separately in the keychain as well; for example, when you call the `SecKeyCreatePair` function to create a key pair, both the public and private keys are stored in the keychain. Use the `SecKeychainSearchCopyNext` (page 54) function to find a key or certificate. Use the `SecIdentitySearchCopyNext` function in the Certificate, Key, and Trust API to find an identity.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`SecImportExport.h`

SecKeychainItemFreeAttributesAndData

Releases the memory used by the keychain attribute list and/or the keychain data retrieved in a call to `SecKeychainItemCopyAttributesAndData`.

```
OSStatus SecKeychainItemFreeAttributesAndData (
   SecKeychainAttributeList *attrList,
   void *data
);
```

Parameters

attrList

A pointer to the attribute list to release. Pass `NULL` if there is no attribute list to release.

data

A pointer to the data buffer to release. Pass `NULL` if there is no data to release.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainItem.h`

SecKeychainItemFreeContent

Releases the memory used by the keychain attribute list and/or the keychain data retrieved in a call to the `SecKeychainItemCopyContent` (page 41) function.

```
OSStatus SecKeychainItemFreeContent (
   SecKeychainAttributeList *attrList,
   void *data
);
```

Parameters

attrList

A pointer to the attribute list to release. Pass `NULL` if there is no attribute list to release.

data

A pointer to the data buffer to release. Pass `NULL` if there is no data to release.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
Because the `SecKeychainFindInternetPassword` (page 33) and `SecKeychainFindGenericPassword` (page 31) functions call the `SecKeychainItemCopyContent` (page 41) function, you must call `SecKeychainItemFreeContent` to release the data buffers after calls to those functions as well.

Because the `SecKeychainItemCopyContent` function does not allocate buffers until they are needed, you should not call the `SecKeychainItemFreeContent` function unless data is actually returned to you.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
ImageClient

Declared In
SecKeychainItem.h

SecKeychainItemGetDLDBHandle

Returns the CSSM database handle for a given keychain item object.

```
OSStatus SecKeychainItemGetDLDBHandle (
   SecKeychainItemRef keyItemRef,
   CSSM_DL_DB_HANDLE *dldbHandle
);
```

Parameters

keyItemRef

A keychain item object.

dldbHandle

On return, a pointer to a CSSM database handle for the keychain database containing the given item. The handle is valid until the keychain item object is released.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychainItem.h

SecKeychainItemGetTypeID

Returns the unique identifier of the opaque type to which a SecKeychainItemRef object belongs.

```
CFTypeID SecKeychainItemGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a SecKeychainItemRef (page 68) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a SecKeychainItemRef (page 68) object. You can compare this value to the CFTypeID identifier obtained by calling the CFGetTypeID function on a specific object. These values might change from release to release or platform to platform.

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychainItem.h

SecKeychainItemGetUniqueRecordID

Returns a CSSM unique record for the given keychain item object.

```
OSStatus SecKeychainItemGetUniqueRecordID (
   SecKeychainItemRef itemRef,
   const CSSM_DB_UNIQUE_RECORD **uniqueRecordID
);
```

Parameters

itemRef

A keychain item object.

uniqueRecordID

On return, a pointer to a CSSM unique record for the given item. The unique record is valid until the item object is released.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecKeychainItem.h`

SecKeychainItemImport

Imports one or more certificates, keys, or identities and adds them to a keychain.

```
OSStatus SecKeychainItemImport (
   CFDataRef importedData,
   CFStringRef fileNameOrExtension,
   SecExternalFormat *inputFormat,
   SecExternalItemType *itemType,
   SecItemImportExportFlags flags,
   const SecKeyImportExportParameters *keyParams,
   SecKeychainRef importKeychain,
   CFArrayRef *outItems
);
```

Parameters

importedData

The external representation of the items to import.

fileNameOrExtension

The name or extension of the file from which the external representation was obtained. Pass `NULL` if you don't know the name or extension.

inputFormat

On input, points to the format of the external representation. Pass `kSecFormatUnknown` if you do not know the exact format. On output, points to the format that the function has determined the external representation to be in. Pass `NULL` if you don't know the format and don't want the format returned to you.

itemType

On input, points to the item type of the item or items contained in the external representation. Pass `kSecItemTypeUnknown` if you do not know the item type. On output, points to the item type that the function has determined the external representation to contain. Pass `NULL` if you don't know the item type and don't want the type returned to you.

flags

Unused; pass in 0.

keyParams

A pointer to a structure containing a set of input parameters for the function. If no key items are being imported, these parameters are optional and you can set the keyParams parameter to NULL.

importKeychain

A keychain object indicating the keychain to which the key or certificate should be imported. If you pass NULL, the item is not imported. Use the SecKeychainCopyDefault (page 27) function to get a reference to the default keychain. If this parameter is NULL, the kSecKeyImportOnlyOne bit in the flags parameter is ignored. Otherwise, if the kSecKeyImportOnlyOne bit is set and there is more than one key in the incoming external representation, no items are imported to the specified keychain and the error errSecMultiplePrivKeys is returned.

outItems

On output, points to an array of SecKeychainItemRef objects for the imported items. You must provide a valid pointer to a CFArrayRef object to receive this information. If you pass NULL for this parameter, the function does not return the imported items. Release this object by calling the CFRelease function when you no longer need it.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

When you pass this function a CFDataRef object containing the external representation of one or more keys, certificates, or identities, SecKeychainItemImport attempts to determine the format and contents of the data. To ensure that this process is successful, you should specify values for one or more of the parameters fileNameOrExtension, inputFormat, and itemType. To have the function add the imported items to a keychain, specify a non-NULL value for the importKeychain parameter. To have the function return SecKeychainItemRef objects for the imported items, specify a non-NULL value for the outItems parameter.

Because the SecKeychainItemImport function determines whether the item is PEM armoured by inspecting the data, the flags parameter is not used in this function call.

After the function returns, you can determine the nature of the keychain items from the values returned in the inputFormat and itemType parameters. Depending on the nature of each item, once it is imported to a keychain you can safely cast the SecKeychainItemRef object to a SecKeyRef, SecCertificateRef, or SecIdentityRef object.

Note that when you import data in PKCS12 format, typically one SecIdentityRef object is returned in the outItems parameter. The data might also include one or more SecCertificateRef objects. The output data will not include any SecKeyRef objects unless the incoming data includes a key with no matching certificate.

When the output item type is kSecItemTypeAggregate, you can use the CFGetTypeID function to determine the Core Foundation type of each item and the functions in "Getting Information About Keychain Services and Types" (page 7) to determine the keychain item type of each item. For example, the following code determines whether the item is a certificate:

```
CFTypeID theID = CFGetTypeID(theItem);
if (SecCertificateGetTypeID() == theID)
```

You can pass in NULL for both outItems and importKeychain to determine what is inside a given external data representation. When you do, the function returns the input format and the item type without modifying the data in any way.

Availability
Available in Mac OS X v10.4 and later.

Declared In
SecImportExport.h

SecKeychainItemModifyAttributesAndData

Updates an existing keychain item after changing its attributes or data.

```
OSStatus SecKeychainItemModifyAttributesAndData (
   SecKeychainItemRef itemRef,
   const SecKeychainAttributeList *attrList,
   UInt32 length,
   const void *data
);
```

Parameters

itemRef
A reference to the keychain item to modify.

attrList
A pointer to the list of attributes to set. Pass NULL if you have no attributes to set.

length
The length of the buffer pointed to by the *data* parameter. Pass 0 if you pass NULL in the data parameter.

data
A pointer to a buffer containing the data to store. Pass NULL if you do not need to modify the data.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
The keychain item is written to the keychain's permanent data store. If the keychain item has not previously been added to a keychain, a call to this function does nothing and returns noErr.

You should pair the SecKeychainItemCopyAttributesAndData (page 39) function with the SecKeychainItemModifyAttributesAndData function, as these functions handle more attributes than are support by the old Keychain Manager and passing them into older calls yields an invalid attribute error. Use the functions SecKeychainItemModifyContent (page 50) and SecKeychainItemCopyContent (page 41) when dealing with older Keychain Manager functions.

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychainItem.h

SecKeychainItemModifyContent

Updates an existing keychain item after changing its attributes and/or data.

```
OSStatus SecKeychainItemModifyContent (
   SecKeychainItemRef itemRef,
   const SecKeychainAttributeList *attrList,
   UInt32 length,
   const void *data
);
```

Parameters

itemRef

A reference to the keychain item to modify.

attrList

A pointer to the list of attributes to set. Pass `NULL` if you have no attributes to set.

length

The length of the buffer pointed to by the `data` parameter. Pass `0` if you pass `NULL` in the `data` parameter.

data

A pointer to a buffer containing the data to store. Pass `NULL` if you do not need to modify the data.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

The keychain item is written to the keychain's permanent data store. If the keychain item has not previously been added to a keychain, a call to this function does nothing and returns `noErr`.

You should pair the `SecKeychainItemModifyContent` function with the `SecKeychainItemCopyContent` (page 41) function when dealing with older Keychain Manager functions. The `SecKeychainItemCopyAttributesAndData` (page 39) and `SecKeychainItemModifyAttributesAndData` (page 50) functions handle more attributes than are support by the old Keychain Manager; however, passing them into older calls yields an invalid attribute error.

Availability

Available in Mac OS X v10.2 and later.

Related Sample Code

ImageClient

Declared In

`SecKeychainItem.h`

SecKeychainItemSetAccess

Sets the access of a given keychain item.

```
OSStatus SecKeychainItemSetAccess (
   SecKeychainItemRef itemRef,
   SecAccessRef access
);
```

Parameters

itemRef

A reference to a keychain item.

access

An access object to replace the keychain item's current access object. Use the `SecAccessCreate` (page 13) function to create an access object.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
Each protected keychain item (such as a password or private key) has an associated access object. The access object contains access control list (ACL) entries, which specify trusted applications and the operations for which those operations are trusted. When an application attempts to access a keychain item for a particular purpose (such as to sign a document), the system determines whether that application is trusted to access the item for that purpose. If it is trusted, then the application is given access and no user confirmation is required. If the application is not trusted, but there is an ACL entry for that operation, then the user is asked to confirm whether the application should be given access. If there is no ACL entry for that operation, then access is denied and it is up to the calling application to try something else or to notify the user.

For more information about ACL entries, see the `SecACLCreateFromSimpleContents` (page 17) function.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainItem.h`

SecKeychainLock

Locks a keychain.

```
OSStatus SecKeychainLock (
   SecKeychainRef keychain
);
```

Parameters

keychain

A reference to the keychain to lock. Pass `NULL` to lock the default keychain.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).The result code `errSecNoSuchKeychain` indicates that specified keychain could not be found. The result code `errSecInvalidKeychain` indicates that the specified keychain is invalid.

Discussion
Your application should not call this 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.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainLockAll

Locks all keychains belonging to the current user.

```
OSStatus SecKeychainLockAll (
   void
);
```

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
Your application should not call this 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.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainOpen

Opens a keychain.

```
OSStatus SecKeychainOpen (
   const char *pathName,
   SecKeychainRef *keychain
);
```

Parameters

pathName
A constant character string representing the POSIX path to the keychain to open.

keychain
On return, a pointer to the keychain object. The memory that this pointer occupies must be released by calling the `CFRelease` function when finished with it.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
You may use this function to retrieve a pointer to a keychain object given the path of the keychain. You do not need to close the keychain, but you should release the memory that the pointer occupies when you are finished with it.

Availability
Available in Mac OS X v10.2 and later.

Related Sample Code
SSLSample

Declared In
`SecKeychain.h`

SecKeychainRemoveCallback

Unregisters your keychain event callback function.

```
OSStatus SecKeychainRemoveCallback (
   SecKeychainCallback callbackFunction
);
```

Parameters

callbackFunction

The callback function pointer to remove.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

Once removed, keychain events are not sent to the owner of the callback.

Availability

Available in Mac OS X v10.2 and later.

Declared In

SecKeychain.h

SecKeychainSearchCopyNext

Finds the next keychain item matching the given search criteria.

```
OSStatus SecKeychainSearchCopyNext (
   SecKeychainSearchRef searchRef,
   SecKeychainItemRef *itemRef
);
```

Parameters

searchRef

A reference to the current search criteria. The search object is created in the SecKeychainSearchCreateFromAttributes (page 55) function and must be released by calling the CFRelease function when you are done with it.

itemRef

On return, a pointer to a keychain item object of the next matching keychain item, if any. You must release this object by calling the CFRelease function.

Return Value

A result code. When there are no more items that match, errSecItemNotFound is returned. See "Keychain Services Result Codes" (page 99).

Discussion

Each item stored in the keychain contains data (such as a certificate), which is indexed by the item's attributes. Use the SecKeychainSearchCreateFromAttributes (page 55) function to specify attributes to search for. If the SecKeychainSearchCopyNext function finds a match, you can use the SecKeychainItemCopyAttributesAndData (page 39) function to retrieve the item's data.

A SecKeychainItemRef object for a certificate that is stored in a keychain can be safely cast to a SecCertificateRef for use with the Certificate, Key, and Trust API.

To find and obtain data from a password keychain item, use the `SecKeychainFindInternetPassword` (page 33) or `SecKeychainFindGenericPassword` (page 31) function.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainSearch.h`

SecKeychainSearchCreateFromAttributes

Creates a search object matching a list of zero or more attributes.

```
OSStatus SecKeychainSearchCreateFromAttributes (
   CFTypeRef keychainOrArray,
   SecItemClass itemClass,
   const SecKeychainAttributeList *attrList,
   SecKeychainSearchRef *searchRef
);
```

Parameters

keychainOrArray

A reference to an array of keychains to search, a single keychain, or `NULL` to search the user's current keychain search list. Use the function `SecKeychainCopySearchList` (page 29) to retrieve the user's default search list.

itemClass

The keychain item class. See "Keychain Item Class Constants" (page 87) for valid constants.

attrList

A pointer to a list of zero or more keychain attribute records to match. Pass `NULL` to match any keychain attribute.

searchRef

On return, a pointer to the current search object. You are responsible for calling the `CFRelease` function to release this object when finished with it.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
Each item stored in the keychain contains data (such as a certificate), which is indexed by the item's attributes. You look up an item in a keychain by its attributes. If you find a match, you can then retrieve the item's data. Use the search object created by this function as input to the `SecKeychainSearchCopyNext` (page 54) function to find a a keychain item and the `SecKeychainItemCopyAttributesAndData` (page 39) function to retrieve the item's data.

To find and obtain data from a password keychain item, use the `SecKeychainFindInternetPassword` (page 33) or `SecKeychainFindGenericPassword` (page 31) function.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainSearch.h`

SecKeychainSearchGetTypeID

Returns the unique identifier of the opaque type to which a `SecKeychainSearchRef` object belongs.

```
CFTypeID SecKeychainSearchGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecKeychainSearchRef` (page 69) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecKeychainSearchRef` (page 69) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychainSearch.h`

SecKeychainSetAccess

Sets the application access for a keychain.

```
OSStatus SecKeychainSetAccess (
   SecKeychainRef keychain,
   SecAccessRef access
);
```

Parameters

keychain
A reference to the keychain for which to set the access. Pass `NULL` to specify the default keychain.

access
An access object of type `SecAccessRef` containing access control lists for the keychain. See "Creating an Access Object" (page 10) for instructions on creating an access object.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
In addition to the ACLs for individual keychain items, the keychain itself has ACLs. However, they are currently unused and this function is unimplemented.

Special Considerations

Although this function is available in Mac OS X v10.2 and later, it is unimplemented and returns an `unimpErr` error code if called.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainSetDefault

Sets the default keychain.

```
OSStatus SecKeychainSetDefault (
   SecKeychainRef keychain
);
```

Parameters

keychain

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

Return Value

A result code. See "Keychain Services Result Codes" (page 99). The result code `errSecNoSuchKeychain` indicates that the specified keychain could not be found. The result code `errSecInvalidKeychain` 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 may call this function to change where a password or other keychain items are added, but since this is a user choice, you should set the default keychain back to the user specified keychain when you are done.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecKeychain.h`

SecKeychainSetDomainDefault

Sets the default keychain for a specified preference domain.

```
OSStatus SecKeychainSetDomainDefault (
   SecPreferencesDomain domain,
   SecKeychainRef keychain
);
```

Parameters

domain

The preference domain for which you wish to set the default keychain. See "Keychain Preference Domain Constants" (page 93) for possible domain values.

keychain

A reference to the keychain you wish to set as default in the specified preference domain.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

A preference domain is a set of security-related preferences, such as the default keychain and the current keychain search list. Use this function if you want to set the default keychain for a specific preference domain. Use the `SecKeychainSetDefault` (page 57) function if you want to set the default keychain for the current preference domain. See the `SecKeychainSetPreferenceDomain` (page 58) function for a discussion of current and default preference domains.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SecKeychain.h

SecKeychainSetDomainSearchList

Sets the keychain search list for a specified preference domain.

```
OSStatus SecKeychainSetDomainSearchList (
   SecPreferencesDomain domain,
   CFArrayRef searchList
);
```

Parameters

domain

The preference domain for which you wish to set the default keychain search list. See "Keychain Preference Domain Constants" (page 93)for possible domain values.

searchList

A pointer to a keychain search list to set in the preference domain.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
A preference domain is a set of security-related preferences, such as the default keychain and the current keychain search list. Use this function if you want to set the keychain search list for a specific preference domain. Use the `SecKeychainSetSearchList` (page 59) function if you want to set the keychain search list for the current preference domain. See the `SecKeychainSetPreferenceDomain` (page 58) function for a discussion of current and default preference domains.

Availability
Available in Mac OS X v10.3 and later.

Declared In
SecKeychain.h

SecKeychainSetPreferenceDomain

Sets the keychain preference domain.

```
OSStatus SecKeychainSetPreferenceDomain (
   SecPreferencesDomain domain
);
```

Parameters

domain

The keychain preference domain to set. See "Keychain Preference Domain Constants" (page 93) for possible domain values.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
A preference domain is a set of security-related preferences, such as the default keychain and the current keychain search list. The default preference domain for system daemons (that is, for daemons running in the root session) is the system domain. The default preference domain for all other programs is the user domain.

This function changes the preference domain for all subsequent function calls; for example, if you change from the system domain to the user domain and then call `SecKeychainLock` (page 52) specifying `NULL` for the keychain, the function locks the default system keychain rather than the default user keychain. You might want to use this function, for example, when launching a system daemon from a user session so that the daemon uses system preferences rather than user preferences.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`SecKeychain.h`

SecKeychainSetSearchList

Specifies the list of keychains to use in the default keychain search list.

```
OSStatus SecKeychainSetSearchList (
   CFArrayRef searchList
);
```

Parameters

searchList

An array of keychain references (of type `SecKeychainRef`) specifying the list of keychains to use in the default keychain search list. Passing an empty array clears the search list.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
The default keychain search list is used by several functions; see for example SecKeychainSearchCreateFromAttributes (page 55), SecKeychainFindInternetPassword (page 33), or SecKeychainFindGenericPassword (page 31). To obtain the current default keychain search list, use the `SecKeychainCopySearchList` (page 29) function.

The default keychain search list is displayed as the keychain list in the Keychain Access utility. If you use `SecKeychainSetSearchList` to change the keychain search list, the list displayed in Keychain Access changes accordingly.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

SecKeychainSetSettings

Changes the settings of a keychain.

```
OSStatus SecKeychainSetSettings (
   SecKeychainRef keychain,
   const SecKeychainSettings *newSettings
);
```

Parameters

keychain

A reference to a keychain whose settings you wish to change. Pass `NULL` to change the settings of the default keychain.

newSettings

A pointer to a keychain settings structure that defines whether the keychain locks when sleeping, or locks after a set time period of inactivity.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecKeychain.h`

SecKeychainSetUserInteractionAllowed

Enables or disables the user interface for Keychain Services functions that automatically display a user interface.

```
OSStatus SecKeychainSetUserInteractionAllowed (
   Boolean state
);
```

Parameters

state

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

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

Certain Keychain Services functions that require the presence of a keychain automatically display a Keychain Not Found dialog if there is none. Functions that require the keychain to be unlocked automatically display the Unlock Keychain dialog. The `SecKeychainSetUserInteractionAllowed` function enables you to control whether these functions display a user interface. By default, user interaction is permitted.

If you are writing an application that must run unattended on a server, you may wish to disable the user interface so that any subsequent keychain calls that normally bring up the unlock UI will instead return immediately with an `errSecInteractionRequired` result). In this case you must programmatically create a keychain or unlock the keychain when necessary.

Special Considerations

If you disable user interaction before calling a Keychain Services function, be sure to reenable it when you are finished. Failure to reenable user interaction will affect other clients of the Keychain Services.

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychain.h

SecKeychainUnlock

Unlocks a keychain.

```
OSStatus SecKeychainUnlock (
   SecKeychainRef keychain,
   UInt32 passwordLength,
   const void *password,
   Boolean usePassword
);
```

Parameters

keychain

A reference to the keychain to unlock. Pass NULL to specify the default keychain. If you pass a locked keychain, this function displays the Unlock Keychain dialog box if you have not provided a password. If the specified keychain is currently unlocked, the Unlock Keychain dialog box is not displayed and this function returns noErr. The memory that the keychain object occupies must be released by calling the function CFRelease when you are finished with it.

passwordLength

An unsigned 32-bit integer representing the length of the password buffer.

password

A buffer containing the password for the keychain. Pass NULL if the user password is unknown. In this case, this function displays the Unlock Keychain dialog to request the user for the keychain password.

usePassword

A Boolean value indicating whether the password parameter is used. You should pass TRUE if you are passing a password or FALSE if it is to be ignored.

Return Value
A result code. See "Keychain Services Result Codes" (page 99). The result code userCanceledErr indicates that the user pressed the Cancel button in the Unlock Keychain dialog box. The result code errSecAuthFailed indicates that authentication failed because of too many unsuccessful retries. The result code errSecInteractionRequired indicates that user interaction is required to unlock the keychain.

Discussion
In most cases, your application does not need to call this function directly, since most Keychain Services functions that require an unlocked keychain do so for you. If your application needs to verify that a keychain is unlocked, call the function SecKeychainGetStatus (page 37).

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychain.h

SecTrustedApplicationCopyData

Retrieves the data of a trusted application object.

```
OSStatus SecTrustedApplicationCopyData (
   SecTrustedApplicationRef appRef,
   CFDataRef *data
);
```

Parameters

appRef

A trusted application object from which to retrieve data. Use the `SecTrustedApplicationCreateFromPath` (page 62) function to create a trusted application object.

data

On return, points to a data object for the data of the trusted application object. Call the `CFRelease` function to release this object when you are finished with it.

Return Value

A result code. See "Keychain Services Result Codes" (page 99).

Discussion

The trusted application object created by the `SecTrustedApplicationCreateFromPath` (page 62) function includes data that uniquely identifies the application, such as a cryptographic hash of the application. The operating system can use this data to verify that the application has not been altered since the trusted application object was created. When an application requests access to an item in the keychain for which it is designated as a trusted application, for example, the operating system checks this data before granting access. You can use the `SecTrustedApplicationCopyData` function to extract this data from the trusted application object for storage or for transmittal to another location (such as over a network). Use the `SecTrustedApplicationSetData` (page 63) function to insert the data back into a trusted application object. Note that this data is in a private format; there is no supported way to read or interpret it.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecTrustedApplication.h`

SecTrustedApplicationCreateFromPath

Creates a trusted application object based on the application specified by path.

```
OSStatus SecTrustedApplicationCreateFromPath (
   const char *path,
   SecTrustedApplicationRef *app
);
```

Parameters

path

The path to the application or tool to trust. For application bundles, use the path to the bundle directory. Pass `NULL` to refer to the application or tool making this call.

app

On return, points to the newly created trusted application object. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
This function creates a trusted application object, which both identifies an application and provides data that can be used to ensure that the application has not been altered since the object was created. The application object is used as input to the `SecAccessCreate` (page 13) function, which creates an access object. The access object, in turn, is used as input to the `SecKeychainItemSetAccess` (page 51) function to specify the set of applications that are trusted to access a specific keychain item.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecTrustedApplication.h`

SecTrustedApplicationGetTypeID

Returns the unique identifier of the opaque type to which a `SecTrustedApplication` object belongs.

```
CFTypeID SecTrustedApplicationGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecTrustedApplicationRef` object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecTrustedApplicationRef` object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecTrustedApplication.h`

SecTrustedApplicationSetData

Sets the data of a given trusted application object.

```
OSStatus SecTrustedApplicationSetData (
   SecTrustedApplicationRef appRef,
   CFDataRef data
);
```

Parameters

appRef
A trusted application object.

data
A reference to the data to set in the trusted application.

Return Value
A result code. See "Keychain Services Result Codes" (page 99).

Discussion
If you used the `SecTrustedApplicationCopyData` (page 62) function to extract the data from a trusted application object for storage or to transmit it to a different location, you can use the `SecTrustedApplicationSetData` function to insert the data into a new trusted application object. Doing so would create an object that identifies the same application as the original trusted application object.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecTrustedApplication.h`

Callbacks

SecKeychainCallback

Defines a pointer to a customized callback function that Keychain Services calls when a keychain event has occurred.

```
typedef OSStatus (*SecKeychainCallback) (
    SecKeychainEvent keychainEvent,
    SecKeychainCallbackInfo *info,
    void *context
);
```

You would declare your keychain callback function like this if you were to name it `MyKeychainCallback`:

```
OSStatus MyKeychainCallback (
    SecKeychainEvent keychainEvent,
    SecKeychainCallbackInfo *info,
    void *context
);
```

Parameters

keychainEvent

The keychain event of which your application wishes to be notified. The type of event that can trigger your callback depends on the bit mask you passed in the `eventMask` parameter of the function `SecKeychainAddCallback` (page 22).

info

A pointer to a structure of type `SecKeychainCallbackInfo`. On return, the structure contains information about the keychain event that occurred. The Keychain Manager passes this information to your callback function through this parameter.

context

A pointer to application-defined storage that your application previously passed to the function `SecKeychainAddCallback` (page 22). 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 Services Result Codes" (page 99).

Discussion
To add your callback function, use the `SecKeychainAddCallback` (page 22) function. To remove your callback function, use the `SecKeychainRemoveCallback` (page 54) function.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecKeychain.h`

Data Types

SecAccessRef

Identifies a keychain or keychain item's access information.

```
typedef struct OpaqueSecAccessRef *SecAccessRef;
```

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecBase.h`

SecACLRef

Represents information about an access control list entry.

```
typedef struct OpaqueSecTrustRef *SecACLRef;
```

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecBase.h`

SecAFPServerSignature

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

```
typedef UInt8 SecAFPServerSignature[16];
```

Discussion
This 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` and `KCFindAppleSharePassword` to represent an Apple File Protocol server signature. You can use a value of this type with the keychain item attribute constant `kSecSignatureItemAttr` to specify an Apple File Protocol server signature.

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychainItem.h

SecKeychainAttribute

Contains keychain attributes.

```
struct SecKeychainAttribute
{
    SecKeychainAttrType tag;
    UInt32 length;
    void *data;
};
typedef struct SecKeychainAttribute SecKeychainAttribute;
typedef SecKeychainAttribute *SecKeychainAttributePtr;
```

Fields

`tag`
A 4-byte attribute tag. See "Keychain Item Attribute Constants" (page 78) for valid attribute types.

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

`data`
A pointer to the attribute data.

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

Declared In
SecBase.h

SecKeychainAttributeInfo

Represents an attribute.

```
struct SecKeychainAttributeInfo
{
    UInt32 count;
    UInt32 *tag;
    UInt32 *format;
};
typedef struct SecKeychainAttributeInfo SecKeychainAttributeInfo;
```

Fields

`count`
The number of tag-format pairs in the respective arrays.

`tag`
A pointer to the first attribute tag in the array.

`format`
A pointer to the first attribute format in the array.

Discussion
Each tag and format item form a pair.

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecBase.h

SecKeychainAttributeList

Represents a list of keychain attributes.

```
struct SecKeychainAttributeList
{
    UInt32 count;
    SecKeychainAttribute *attr;
};
typedef struct SecKeychainAttributeList SecKeychainAttributeList;
```

Fields
`count`
An unsigned 32-bit integer that represents the number of keychain attributes in the array.
`attr`
A pointer to the first keychain attribute in the array.

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

Declared In
SecBase.h

SecKeychainAttrType

Represents a keychain attribute type.

```
typedef OSType SecKeychainAttrType;
```

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

Declared In
SecBase.h

SecKeychainCallbackInfo

Contains information about a keychain event.

```
struct SecKeychainCallbackInfo
{
    UInt32 version;
    SecKeychainItemRef item;
    SecKeychainRef keychain;
    pid_t pid;
};
typedef struct SecKeychainCallbackInfo SecKeychainCallbackInfo;
```

Fields

`version`

The version of this structure. See "Keychain Settings Version" (page 98) for valid constants.

`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.

`keychain`

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

`pid`

The ID of the process that generated this event.

Discussion

This structure contains information about the keychain event of which your application wants to be notified. Keychain Services passes a pointer to this structure in the `info` parameter of your callback function. For information on how to write a keychain event callback function, see `SecKeychainCallback` (page 64).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecKeychain.h`

SecKeychainItemRef

Contains information about a keychain item.

```
typedef struct OpaqueSecKeychainItemRef *SecKeychainItemRef;
```

Discussion

A `SecKeychainItemRef` object for a certificate that is stored in a keychain can be safely cast to a `SecCertificateRef` for use with the Certificate, Key, and Trust API.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`SecBase.h`

SecKeychainRef

Contains information about a keychain.

```
typedef struct OpaqueSecKeychainRef *SecKeychainRef;
```

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

Declared In
`SecBase.h`

SecKeychainSearchRef

Contains information about a keychain search.

```
typedef struct OpaqueSecKeychainSearchRef *SecKeychainSearchRef;
```

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

Declared In
`SecBase.h`

SecKeychainSettings

Contains information about keychain settings.

```
struct SecKeychainSettings
{
    UInt32          version;
    Boolean         lockOnSleep;
    Boolean         useLockInterval;
    UInt32          lockInterval;
};
typedef struct SecKeychainSettings    SecKeychainSettings;
```

Fields

`version`
: An unsigned 32-bit integer representing the keychain version.

`lockOnSleep`
: A Boolean value indicating whether the keychain locks when the system sleeps.

`useLockInterval`
: A Boolean value indicating whether the keychain automatically locks after a certain period of time.

`lockInterval`
: An unsigned 32-bit integer representing the number of seconds before the keychain locks. If you set `useLockInterval` to `FALSE`, set `lockInterval` to `INT_MAX` to indicate that the keychain never locks.

Discussion
This structure contains information about a keychain's settings such as locking on sleep and the lock time interval. You can use the `SecKeychainSetSettings` (page 59) and `SecKeychainCopySettings` (page 29) functions to set and copy a keychain's settings.

Availability
Available in Mac OS X v10.2 and later.

Declared In
SecKeychain.h

SecKeyImportExportParameters

Contains input parameters for import and export functions.

```
typedef struct
{
    /* for import and export */
    uint32_t                version;
    SecKeyImportExportFlags flags;
    CFTypeRef               passphrase;
    CFStringRef             alertTitle;
    CFStringRef             alertPrompt;

    /* for import only */
    SecAccessRef            accessRef;
    CSSM_KEYUSE             keyUsage;
    CSSM_KEYATTR_FLAGS      keyAttributes;
} SecKeyImportExportParameters;
```

Fields

`version`
The version of this structure; the current value is `SEC_KEY_IMPORT_EXPORT_PARAMS_VERSION`.

`flags`
A set of flag bits, defined in "Keychain Item Import/Export Formats" (page 90).

`passphrase`
A password, used for `kSecFormatPKCS12` and `kSecFormatWrapped` formats only. (A password is sometimes referred to as a passphrase to emphasize the fact that a longer string that includes non-letter characters, such as numbers, punctuation, and spaces, is more secure than a simple word.) Legal types are `CFStringRef` and `CFDataRef`. PKCS12 requires passwords to be in Unicode format; passing in a `CFStringRef` as the password is the safest way to ensure that this requirement is met (and that the result is compatible with other implementations). If a `CFDataRef` object is supplied as the password for a PKCS12 export operation, the data is assumed to be in UTF8 form and is converted as appropriate.

When importing or exporting keys (`SecKeyRef` objects) in one of the wrapped formats (`kSecFormatWrappedOpenSSL`, `kSecFormatWrappedSSH`, or `kSecFormatWrappedPKCS8`) or in `PKCS12` format, you must either explicitly specify the `passphrase` field or set the `kSecKeySecurePassphrase` bit in the `Flags` field (to prompt the user for the password).

`alertTitle`
Title of secure password alert panel. When importing or exporting a key, if you set the `kSecKeySecurePassphrase` flag bit, you can optionally use this field to specify a string for the password panel's title bar.

`alertPrompt`
Prompt in secure password alert panel. When importing or exporting a key, if you set the `kSecKeySecurePassphrase` flag bit, you can optionally use this field to specify a string for the prompt that appears in the password panel.

`accessRef`

Specifies the initial access controls of imported private keys. If more than one private key is being imported, all private keys get the same initial access controls. If this field is `NULL` when private keys are being imported, then the access object for the keychain item for an imported private key depends on the `kSecKeyNoAccessControl` bit in the `flags` parameter. If this bit is `0` (or `keyParams` is `NULL`), the default access control is used. If this bit is `1`, no access object is attached to the keychain item for imported private keys.

`keyUsage`

A word of bits constituting the low-level use flags for imported keys as defined in `cssmtype.h`. If this field is `0` or `keyParams` is `NULL`, the default value is `CSSM_KEYUSE_ANY`.

`keyAttributes`

A word of bits constituting the low-level attribute flags for imported keys. The default value is `CSSM_KEYATTR_SENSITIVE | CSSM_KEYATTR_EXTRACTABLE`; the `CSSM_KEYATTR_PERMANENT` bit is also added to the default if a non-`NULL` value is specified for the `importKeychain` parameter.

The following are valid values for these flags: `CSSM_KEYATTR_PERMANENT`, `CSSM_KEYATTR_SENSITIVE`, and `CSSM_KEYATTR_EXTRACTABLE`.

If the `CSSM_KEYATTR_PERMANENT` bit is set, the `importKeychain` parameter is not valid, and if any keys are found in the external representation, then the error `errSecInvalidKeychain` is returned.

The `CSSM_KEYATTR_SENSITIVE` bit indicates that the key can only be extracted in wrapped form.

Important: If you do not set the `CSSM_KEYATTR_EXTRACTABLE` bit, you cannot extract the imported key from the keychain in any form, including in wrapped form.

The `CSSM_KEYATTR_FLAGS` enumeration is defined in `cssmtype.h`. Note that the `CSSM_KEYATTR_RETURN_xxx` bits are always forced to `CSSM_KEYATTR_RETURN_REF` regardless of how they are specified in the `keyAttributes` field.

Discussion
This structure is passed in the `keyParams` parameter as input to the functions `SecKeychainItemExport` (page 44) and `SecKeychainItemImport` (page 48).

PKCS12 is an abbreviation for Public-Key Cryptography Standard # 12. This standard, by RSA Security, provides a format for external representation of keys and certificates and is described in *PKCS 12 v1.0: Personal Information Exchange Syntax*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`SecImportExport.h`

SecTrustedApplicationRef

Contains information about a trusted application.

```
typedef struct OpaqueSecTrustedApplicationRef *SecTrustedApplicationRef;
```

Availability
Available in Mac OS X v10.2 and later.

Declared In
`SecBase.h`

Constants

Mac OS X Keychain Services API Constants

Authorization Tag Type Constants

Defines constants that specify which operations an access control list entry applies to.

```
typedef sint32 CSSM_ACL_AUTHORIZATION_TAG;
enum {
    CSSM_ACL_AUTHORIZATION_TAG_VENDOR_DEFINED_START =
                                        0x00010000,
    CSSM_ACL_AUTHORIZATION_ANY =        CSSM_WORDID__STAR_,
    CSSM_ACL_AUTHORIZATION_LOGIN =      CSSM_WORDID_LOGIN,
    CSSM_ACL_AUTHORIZATION_GENKEY =     CSSM_WORDID_GENKEY,
    CSSM_ACL_AUTHORIZATION_DELETE =     CSSM_WORDID_DELETE,
    CSSM_ACL_AUTHORIZATION_EXPORT_WRAPPED =
                                        CSSM_WORDID_EXPORT_WRAPPED,
    CSSM_ACL_AUTHORIZATION_EXPORT_CLEAR =CSSM_WORDID_EXPORT_CLEAR,
    CSSM_ACL_AUTHORIZATION_IMPORT_WRAPPED =
                                        CSSM_WORDID_IMPORT_WRAPPED,
    CSSM_ACL_AUTHORIZATION_IMPORT_CLEAR =CSSM_WORDID_IMPORT_CLEAR,
    CSSM_ACL_AUTHORIZATION_SIGN =       CSSM_WORDID_SIGN,
    CSSM_ACL_AUTHORIZATION_ENCRYPT =    CSSM_WORDID_ENCRYPT,
    CSSM_ACL_AUTHORIZATION_DECRYPT =    CSSM_WORDID_DECRYPT,
    CSSM_ACL_AUTHORIZATION_MAC =        CSSM_WORDID_MAC,
    CSSM_ACL_AUTHORIZATION_DERIVE =     CSSM_WORDID_DERIVE
};
/* Apple-defined ACL authorization tags */
enum {
    CSSM_ACL_AUTHORIZATION_CHANGE_ACL =
                CSSM_ACL_AUTHORIZATION_TAG_VENDOR_DEFINED_START,
    CSSM_ACL_AUTHORIZATION_CHANGE_OWNER
};
```

Constants

`CSSM_ACL_AUTHORIZATION_TAG_VENDOR_DEFINED_START`

All vendor specific constants must be in the number range starting at this value.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_ANY`

No restrictions. This ACL entry applies to all operations available to the caller.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_LOGIN`

Use for a CSP (smart card) login.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_GENKEY`
Generate a key.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_DELETE`
Delete this item.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_EXPORT_WRAPPED`
Export a wrapped (that is, encrypted) key. This tag is checked on the key being exported; in addition, the `CSSM_ACL_AUTHORIZATION_ENCRYPT` tag is checked for any key used in the wrapping operation.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_EXPORT_CLEAR`
Export an unencrypted key.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_IMPORT_WRAPPED`
Import an encrypted key. This tag is checked on the key being imported; in addition, the `CSSM_ACL_AUTHORIZATION_DECRYPT` tag is checked for any key used in the unwrapping operation.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_IMPORT_CLEAR`
Import an unencrypted key.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_SIGN`
Digitally sign data.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_ENCRYPT`
Encrypt data.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_DECRYPT`
Decrypt data.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_MAC`
Create or verify a message authentication code.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_DERIVE`
Derive a new key from another key.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_ACL_AUTHORIZATION_CHANGE_ACL`
Change an access control list entry.

Available in Mac OS X v10.0 and later.

Declared in `cssmapple.h`.

`CSSM_ACL_AUTHORIZATION_CHANGE_OWNER`
For internal system use only. Use the `CSSM_ACL_AUTHORIZATION_CHANGE_ACL` tag for changes to owner ACL entries.

Available in Mac OS X v10.0 and later.

Declared in `cssmapple.h`.

Import/Export Parameters Version

Defines the version of an import/export parameters structure.

```
#define SEC_KEY_IMPORT_EXPORT_PARAMS_VERSION        0
```

Constants

`SEC_KEY_IMPORT_EXPORT_PARAMS_VERSION`
Defines the version number for a `SecImportExportParameters` structure used as input to the functions `SecKeychainItemExport` (page 44) and `SecKeychainItemImport` (page 48).

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

Keychain Authentication Type Constants

Defines constants you can use to identify the type of authentication to use for an Internet password.

```
typedef FourCharCode SecAuthenticationType;
enum
{
    kSecAuthenticationTypeNTLM              = AUTH_TYPE_FIX_ ('ntlm'),
    kSecAuthenticationTypeMSN               = AUTH_TYPE_FIX_ ('msna'),
    kSecAuthenticationTypeDPA               = AUTH_TYPE_FIX_ ('dpaa'),
    kSecAuthenticationTypeRPA               = AUTH_TYPE_FIX_ ('rpaa'),
    kSecAuthenticationTypeHTTPBasic         = AUTH_TYPE_FIX_ ('http'),
    kSecAuthenticationTypeHTTPDigest        = AUTH_TYPE_FIX_ ('httd'),
    kSecAuthenticationTypeHTMLForm          = AUTH_TYPE_FIX_ ('form'),
    kSecAuthenticationTypeDefault           = AUTH_TYPE_FIX_ ('dflt')
};
```

Constants

`kSecAuthenticationTypeNTLM`
Specifies Windows NT LAN Manager authentication.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecAuthenticationTypeMSN`
Specifies Microsoft Network default authentication.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecAuthenticationTypeDPA`
Specifies Distributed Password authentication.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecAuthenticationTypeRPA`
Specifies Remote Password authentication.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecAuthenticationTypeHTTPBasic`
Specifies HTTP Basic authentication. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecAuthenticationTypeHTTPDigest`
Specifies HTTP Digest Access authentication.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecAuthenticationTypeHTMLForm`
Specifies HTML form based authentication. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecAuthenticationTypeDefault`
Specifies the default authentication type.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

Keychain Event Constants

Defines the keychain-related event.

```
typedef UInt32 SecKeychainEvent;
enum
{
    kSecLockEvent               = 1,
    kSecUnlockEvent             = 2,
    kSecAddEvent                = 3,
    kSecDeleteEvent             = 4,
    kSecUpdateEvent             = 5,
    kSecPasswordChangedEvent    = 6,
    kSecDefaultChangedEvent     = 9,
    kSecDataAccessEvent         = 10,
    kSecKeychainListChangedEvent = 11
};
```

Constants

`kSecLockEvent`

Indicates a keychain was locked. It is impossible to distinguish between a lock event caused by an explicit request and one caused by a keychain that locked itself because of a timeout. Therefore, the `pid` parameter in the `SecKeychainCallbackInfo` structure does not contain useful information for this event. Note that when the login session terminates, all keychains become effectively locked; however, no `kSecLockEvent` events are generated in this case.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecUnlockEvent`

Indicates a keychain was successfully unlocked. It is impossible to distinguish between an unlock event caused by an explicit request and one that occurred automatically because the keychain was needed to perform an operation. In either case, however, the `pid` parameter in the `SecKeychainCallbackInfo` structure does return the ID of the process whose actions caused the unlock event.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecAddEvent`

Indicates an item was added to a keychain.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecDeleteEvent`

Indicates an item was deleted from a keychain.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecUpdateEvent`

Indicates a keychain item was updated.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecPasswordChangedEvent`

Indicates the keychain password was changed.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecDefaultChangedEvent`
Indicates that a different keychain was specified as the default.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecDataAccessEvent`
Indicates a process has accessed a keychain item's data.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecKeychainListChangedEvent`
Indicates the list of keychains has changed.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

Keychain Event Mask Constants

Defines bit masks for keychain event constants

```
typedef UInt32 SecKeychainEventMask;
enum
{
    kSecLockEventMask            = 1 << kSecLockEvent,
    kSecUnlockEventMask          = 1 << kSecUnlockEvent,
    kSecAddEventMask             = 1 << kSecAddEvent,
    kSecDeleteEventMask          = 1 << kSecDeleteEvent,
    kSecUpdateEventMask          = 1 << kSecUpdateEvent,
    kSecPasswordChangedEventMask = 1 << kSecPasswordChangedEvent,
    kSecDefaultChangedEventMask  = 1 << kSecDefaultChangedEvent,
    kSecDataAccessEventMask      = 1 << kSecDataAccessEvent,
    kSecKeychainListChangedMask  = 1 << kSecKeychainListChangedEvent,
    kSecEveryEventMask           = 0xffffffff
};
```

Constants

`kSecLockEventMask`
If the bit specified by this mask is set, your callback function is invoked when a keychain is locked.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecUnlockEventMask`
If the bit specified by this mask is set, your callback function is invoked when a keychain is unlocked.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

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

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecDeleteEventMask`
: If the bit specified by this mask is set, your callback function is invoked when an item is deleted from a keychain.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychain.h`.

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

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychain.h`.

`kSecPasswordChangedEventMask`
: If the bit specified by this mask is set, your callback function is invoked when the keychain password is changed.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychain.h`.

`kSecDefaultChangedEventMask`
: If the bit specified by this mask is set, your callback function is invoked when a different keychain is specified as the default.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychain.h`.

`kSecDataAccessEventMask`
: If the bit specified by this mask is set, your callback function is invoked when a process accesses a keychain item's data.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychain.h`.

`kSecKeychainListChangedMask`
: If the bit specified by this mask is set, your callback function is invoked when a keychain list is changed.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychain.h`.

`kSecEveryEventMask`
: If all the bits are set, your callback function is invoked whenever any event occurs.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychain.h`.

Keychain Item Attribute Constants

Specifies a keychain item's attributes.

```
typedef FourCharCode SecItemAttr;
enum
{
    kSecCreationDateItemAttr        = 'cdat',
    kSecModDateItemAttr             = 'mdat',
    kSecDescriptionItemAttr         = 'desc',
    kSecCommentItemAttr             = 'icmt',
    kSecCreatorItemAttr             = 'crtr',
    kSecTypeItemAttr                = 'type',
    kSecScriptCodeItemAttr          = 'scrp',
    kSecLabelItemAttr               = 'labl',
    kSecInvisibleItemAttr           = 'invi',
    kSecNegativeItemAttr            = 'nega',
    kSecCustomIconItemAttr          = 'cusi',
    kSecAccountItemAttr             = 'acct',
    kSecServiceItemAttr             = 'svce',
    kSecGenericItemAttr             = 'gena',
    kSecSecurityDomainItemAttr      = 'sdmn',
    kSecServerItemAttr              = 'srvr',
    kSecAuthenticationTypeItemAttr  = 'atyp',
    kSecPortItemAttr                = 'port',
    kSecPathItemAttr                = 'path',
    kSecVolumeItemAttr              = 'vlme',
    kSecAddressItemAttr             = 'addr',
    kSecSignatureItemAttr           = 'ssig',
    kSecProtocolItemAttr            = 'ptcl',
    kSecCertificateType             = 'ctyp',
    kSecCertificateEncoding         = 'cenc',
    kSecCrlType                     = 'crtp',
    kSecCrlEncoding                 = 'crnc',
    kSecAlias                       = 'alis'
};
```

Constants

`kSecCreationDateItemAttr`

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 v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecModDateItemAttr`

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 v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecDescriptionItemAttr`

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 particular kind of item, for example “disk image password”.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecCommentItemAttr`
: 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 v10.2 and later.

  Declared in `SecKeychainItem.h`.

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

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychainItem.h`.

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

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychainItem.h`.

`kSecScriptCodeItemAttr`
: 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. Use of this attribute is deprecated; string attributes should be stored in UTF-8 encoding.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychainItem.h`.

`kSecLabelItemAttr`
: 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 v10.2 and later.

  Declared in `SecKeychainItem.h`.

`kSecInvisibleItemAttr`
: 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 v10.2 and later.

  Declared in `SecKeychainItem.h`.

`kSecNegativeItemAttr`
: 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, which is typically invisible and with zero-length data, acts as a placeholder.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychainItem.h`.

`kSecCustomIconItemAttr`

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 `kSecTypeItemAttr` to a file type for which there is a corresponding icon in the desktop database, and set the attribute value identified by the tag `kSecCreatorItemAttr` 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 v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecAccountItemAttr`

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

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecServiceItemAttr`

Identifies the service attribute. You use this tag to set or get a string that represents the service associated with this item, for example, "iTools". This is unique to generic password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecGenericItemAttr`

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

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecSecurityDomainItemAttr`

Identifies the security domain attribute. You use this tag to set or get a value that represents the Internet security domain. This is unique to Internet password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecServerItemAttr`

Identifies the server attribute. You use this tag to set or get a string that represents the Internet server's domain name or IP address. This is unique to Internet password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecAuthenticationTypeItemAttr`

Identifies the authentication type attribute. You use this tag to set or get a value of type `SecAuthenticationType` that represents the Internet authentication scheme. For possible authentication values, see "Keychain Authentication Type Constants" (page 74). This is unique to Internet password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecPortItemAttr`
Identifies the port attribute. You use this tag to set or get a value of type `UInt32` that represents the Internet port number. This is unique to Internet password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecPathItemAttr`
Identifies the path attribute. You use this tag to set or get a value that represents the path. This is unique to Internet password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecVolumeItemAttr`
Identifies the volume attribute. You use this tag to set or get a value that represents the AppleShare volume. This is unique to AppleShare password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecAddressItemAttr`
Identifies the address attribute. You use this tag to set or get a value of type `string` that represents the AppleTalk zone name, or the IP or domain name that represents the server address. This is unique to AppleShare password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecSignatureItemAttr`
Identifies the server signature attribute. You use this tag to set or get a value of type `SecAFPServerSignature` (page 65) that represents the server signature block. This is unique to AppleShare password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecProtocolItemAttr`
Identifies the protocol attribute. You use this tag to set or get a value of type `SecProtocolType` that represents the Internet protocol. For possible protocol type values, see "Keychain Protocol Type Constants" (page 94). This is unique to AppleShare and Internet password attributes.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecCertificateType`
Indicates a `CSSM_CERT_TYPE` type.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecCertificateEncoding`
Indicates a `CSSM_CERT_ENCODING` type.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecCrlType`
: Indicates a `CSSM_CRL_TYPE` type.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychainItem.h`.

`kSecCrlEncoding`
: Indicates a `CSSM_CRL_ENCODING` type.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychainItem.h`.

`kSecAlias`
: Indicates an alias.

  Available in Mac OS X v10.2 and later.

  Declared in `SecKeychainItem.h`.

Discussion
Not all of these attributes are used for all types of items. Which set of attributes exist for each type of item is documented in the "Data Storage Library Services" chapter of *Common Security: CDSA and CSSM, version 2 (with corrigenda)* from The Open Group (http://www.opengroup.org/security/cdsa.htm) for standard items and in the DL section of the *Security Release Notes* for Apple-defined item types (if any).

To obtain information about a certificate, use the CDSA Certificate Library (CL) API. To obtain information about a key, use the `SecKeyGetCSSMKey` function and the CDSA Cryptographic Service Provider (CSP) API.

For attributes for keys, see "Keychain Item Attribute Constants For Keys" (page 83).

Keychain Item Attribute Constants For Keys

Specifies the attributes for a key item in a keychain.

```
enum
{
    kSecKeyKeyClass         =0,
    kSecKeyPrintName        =1,
    kSecKeyAlias            =2,
    kSecKeyPermanent        =3,
    kSecKeyPrivate          =4,
    kSecKeyModifiable       =5,
    kSecKeyLabel            =6,
    kSecKeyApplicationTag   =7,
    kSecKeyKeyCreator       =8,
    kSecKeyKeyType          =9,
    kSecKeyKeySizeInBits    =10,
    kSecKeyEffectiveKeySize =11,
    kSecKeyStartDate        =12,
    kSecKeyEndDate          =13,
    kSecKeySensitive        =14,
    kSecKeyAlwaysSensitive  =15,
    kSecKeyExtractable      =16,
    kSecKeyNeverExtractable =17,
    kSecKeyEncrypt          =18,
    kSecKeyDecrypt          =19,
    kSecKeyDerive           =20,
    kSecKeySign             =21,
    kSecKeyVerify           =22,
    kSecKeySignRecover      =23,
    kSecKeyVerifyRecover    =24,
    kSecKeyWrap             =25,
    kSecKeyUnwrap           =26
};
```

Constants

`kSecKeyKeyClass`

Type uint32 (`CSSM_KEYCLASS`); value is one of `CSSM_KEYCLASS_PUBLIC_KEY`, `CSSM_KEYCLASS_PRIVATE_KEY` or `CSSM_KEYCLASS_SESSION_KEY`.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyPrintName`

Type blob; human readable name of the key. Same as `kSecLabelItemAttr` for normal keychain items.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyAlias`

Type blob; currently unused.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyPermanent`

Type uint32; value is nonzero. This key is permanent (stored in some keychain) and is always `1`.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyPrivate`
Type uint32; value is nonzero. This key is protected by a user login, a password, or both.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyModifiable`
Type uint32; value is nonzero. Attributes of this key can be modified.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyLabel`
Type blob; for private and public keys this contains the hash of the public key. This is used to associate certificates and keys. Its value matches the value of the `kSecPublicKeyHashItemAttr` attribute of a certificate and it's used to construct an identity from a certificate and a key. For symmetric keys this is whatever the creator of the key passed in when they generated the key.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyApplicationTag`
Type blob; currently unused.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyKeyCreator`
Type data. The data points to a `CSSM_GUID` structure representing the module ID of the CSP owning this key.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyKeyType`
Type uint32; value is a CSSM algorithm (`CSSM_ALGORITHMS`) representing the algorithm associated with this key.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyKeySizeInBits`
Type uint32; value is the number of bits in this key.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyEffectiveKeySize`
Type uint32; value is the effective number of bits in this key. For example, a DES key has a key size in bits (`kSecKeyKeySizeInBits`) of 64 but a value for `kSecKeyEffectiveKeySize` of 56.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyStartDate`
Type `CSSM_DATE`. Earliest date at which this key may be used. If the value is all zeros or not present, no restriction applies.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyEndDate`
Type `CSSM_DATE`. Latest date at which this key may be used. If the value is all zeros or not present, no restriction applies.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeySensitive`
Type uint32; value is nonzero. This key cannot be wrapped with `CSSM_ALGID_NONE`.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyAlwaysSensitive`
Type uint32; value is nonzero. This key has always been marked sensitive.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyExtractable`
Type uint32; value is nonzero. This key can be wrapped.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyNeverExtractable`
Type uint32; value is nonzero. This key was never marked extractable.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyEncrypt`
Type uint32; value is nonzero. This key can be used in an encrypt operation.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyDecrypt`
Type uint32; value is nonzero. This key can be used in a decrypt operation.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyDerive`
Type uint32; value is nonzero. This key can be used in a key derivation operation.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeySign`
Type uint32, value is nonzero. This key can be used in a sign operation.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyVerify`
Type uint32, value is nonzero. This key can be used in a verify operation.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeySignRecover`
Type uint32.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyVerifyRecover`
Type uint32. This key can unwrap other keys.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyWrap`
Type uint32; value is nonzero. This key can wrap other keys.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

`kSecKeyUnwrap`
Type uint32; value is nonzero. This key can unwrap other keys.

Available in Mac OS X v10.4 and later.

Declared in `SecKey.h`.

Discussion
For attributes for items other than keys, see "Keychain Item Attribute Constants" (page 78).

Keychain Item Class Constants

Specifies a keychain item's class code.

```
typedef FourCharCode SecItemClass;
enum
{
    /* SecKeychainItem.h */
    kSecInternetPasswordItemClass   = 'inet',
    kSecGenericPasswordItemClass    = 'genp',
    kSecAppleSharePasswordItemClass = 'ashp',
    kSecCertificateItemClass        =
                            CSSM_DL_DB_RECORD_X509_CERTIFICATE,
};
enum
{
    /* Record Types defined in The Open Group Application Name Space  */
    /* cssmtype.h */
    CSSM_DL_DB_RECORD_PUBLIC_KEY =
                            CSSM_DB_RECORDTYPE_OPEN_GROUP_START  + 5,
    CSSM_DL_DB_RECORD_PRIVATE_KEY =
                            CSSM_DB_RECORDTYPE_OPEN_GROUP_START  + 6,
    CSSM_DL_DB_RECORD_SYMMETRIC_KEY =
                            CSSM_DB_RECORDTYPE_OPEN_GROUP_START  + 7,
    CSSM_DL_DB_RECORD_ALL_KEYS =
                            CSSM_DB_RECORDTYPE_OPEN_GROUP_START  + 8
};
```

Constants

`kSecInternetPasswordItemClass`
Indicates that the item is an Internet password.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecGenericPasswordItemClass`
Indicates that the item is a generic password.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecAppleSharePasswordItemClass`
Indicates that the item is an AppleShare password.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`kSecCertificateItemClass`
Indicates that the item is an X509 certificate.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychainItem.h`.

`CSSM_DL_DB_RECORD_PUBLIC_KEY`
Indicates that the item is a public key of a public-private pair.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_DL_DB_RECORD_PRIVATE_KEY`
Indicates that the item is a private key of a public-private pair.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_DL_DB_RECORD_SYMMETRIC_KEY`

Indicates that the item is a private key used for symmetric-key encryption.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

`CSSM_DL_DB_RECORD_ALL_KEYS`

The item can be any type of key; used for searches only.

Available in Mac OS X v10.0 and later.

Declared in `cssmtype.h`.

Discussion

These enumerations define 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 can also use these constants with the tag constant `SecItemAttr`.

Declared In

`SecKeychainItem.h, cssmtype.h.`

Keychain Item Import/Export Flags

Defines values for import and export flags.

```
enum
{
    kSecItemPemArmour           = 0x00000001,
};
typedef uint32_t SecItemImportExportFlags;
```

Constants

`kSecItemPemArmour`

The exported data should have PEM armour.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

Discussion

This enumeration lists values used by the `flags` parameter of the functions `SecKeychainItemExport` (page 44) and `SecKeychainItemImport` (page 48).

PEM armour refers to a way of expressing binary data as an ASCII string so that it can be transferred over text-only channels such as email. (PEM stands for an Internet standard, Privacy Enhanced Mail.)

Keychain Item Import/Export Parameter Flags

Defines values for the `flags` field of the import/export parameters.

```
enum
{
    kSecKeyImportOnlyOne        = 0x00000001,
    kSecKeySecurePassphrase     = 0x00000002,
    kSecKeyNoAccessControl      = 0x00000004
};
typedef uint32_t SecKeyImportExportFlags;
```

Constants

`kSecKeyImportOnlyOne`

Prevents the importing of more than one private key by the `SecKeychainItemImport` (page 48) function. If the `importKeychain` parameter is `NULL`, this bit is ignored. Otherwise, if this bit is set and there is more than one key in the incoming external representation, no items are imported to the specified keychain and the error `errSecMultipleKeys` is returned.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecKeySecurePassphrase`

When set, the password for import or export is obtained by user prompt. (A password is sometimes referred to as a passphrase to emphasize the fact that a longer string that includes non-letter characters, such as numbers, punctuation, and spaces, is more secure than a simple word.) Otherwise, you must provide the password in the `passphrase` field of the `SecKeyImportExportParameters` structure. A user-supplied password is preferred, because it avoids having the cleartext password appear in the application's address space at any time.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecKeyNoAccessControl`

When set, imported private keys have no access object attached to them. In the absence of both this bit and the `accessRef` field in `SecKeyImportExportParameters`, imported private keys are given default access controls.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

Discussion

These flags are used as input to the import/export parameters structure (`SecKeyImportExportParameters` (page 70), which in turn is used as input to the functions `SecKeychainItemExport` (page 44) and `SecKeychainItemImport` (page 48).

Keychain Item Import/Export Formats

Specifies the format of an item after export from or before import to the keychain.

```
enum
{
    kSecFormatUnknown = 0,

    /* Asymmetric Key Formats */
    kSecFormatOpenSSL,
    kSecFormatSSH,      //not supported
    kSecFormatBSAFE,
```

```
    /* Symmetric Key Formats */
    kSecFormatRawKey,

    /* Formats for wrapped symmetric and private keys */
    kSecFormatWrappedPKCS8,
    kSecFormatWrappedOpenSSL,
    kSecFormatWrappedSSH,   //not supported
    kSecFormatWrappedLSH,   //not supported

    /* Formats for certificates */
    kSecFormatX509Cert,

    /* Aggregate Types */
    kSecFormatPEMSequence,
    kSecFormatPKCS7,
    kSecFormatPKCS12,
    kSecFormatNetscapeCertSequence
};
typedef uint32_t SecExternalFormat;
```

Constants

`kSecFormatUnknown`

When importing, indicates the format is unknown. When exporting, use the default format for the item. For asymmetric keys, the default is `kSecFormatOpenSSL`. For symmetric keys, the default is `kSecFormatRawKey`. For certificates, the default is `kSecFormatX509Cert`. For multiple items, the default is `kSecFormatPEMSequence`.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatOpenSSL`

Format for asymmetric (public/private) keys. OpenSSL is an open source toolkit for Secure Sockets Layer (SSL) and Transport Layer Security (TLS). Also known as X.509 for public keys.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatSSH`

Not supported.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatBSAFE`

Format for asymmetric keys. BSAFE is a standard from RSA Security for encryption, digital signatures, and privacy.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatRawKey`

Format for symmetric keys. Raw, unformatted key bits. This is the default for symmetric keys.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatWrappedPKCS8`
Format for wrapped symmetric and private keys. PKCS8 is the Private-Key Information Syntax Standard from RSA Security.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatWrappedOpenSSL`
Format for wrapped symmetric and private keys. OpenSSL is an open-source toolkit for Secure Sockets Layer (SSL) and Transport Layer Security (TLS).

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatWrappedSSH`
Not supported.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatWrappedLSH`
Not supported.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatX509Cert`
Format for certificates. DER (distinguished encoding rules) encoded. X.509 is a standard for digital certificates from the International Telecommunication Union (ITU). This is the default for certificates.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatPEMSequence`
Sequence of certificates and keys with PEM armour. PEM armour refers to a way of expressing binary data as an ASCII string so that it can be transferred over text-only channels such as email. This is the default format for multiple items.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatPKCS7`
Sequence of certificates, no PEM armour. PKCS7 is the Cryptographic Message Syntax Standard from RSA Security, Inc.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatPKCS12`
Set of certificates and private keys. PKCS12 is the Personal Information Exchange Syntax from RSA Security, Inc.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecFormatNetscapeCertSequence`
Set of certificates in the Netscape Certificate Sequence format.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

Keychain Item Type When Importing

Specifies the type of keychain item being imported.

```
enum {
    kSecItemTypeUnknown,            /* caller doesn't know what  this is */
    kSecItemTypePrivateKey,
    kSecItemTypePublicKey,
    kSecItemTypeSessionKey,
    kSecItemTypeCertificate,
    kSecItemTypeAggregate
};
typedef uint32_t SecExternalItemType;
```

Constants

`kSecItemTypePrivateKey`
Indicates a private key.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecItemTypePublicKey`
Indicates a public key.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecItemTypeSessionKey`
Indicates a session key.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecItemTypeCertificate`
Indicates a certificate.

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

`kSecItemTypeAggregate`
Indicates a set of certificates or certificates and private keys, such as `PKCS7`, `PKCS12`, or `kSecFormatPEMSequence` formats (see "Keychain Item Import/Export Formats" (page 90)).

Available in Mac OS X v10.4 and later.

Declared in `SecImportExport.h`.

Keychain Preference Domain Constants

Defines constants for the keychain preference domains.

```
typedef enum {
    kSecPreferencesDomainUser,
    kSecPreferencesDomainSystem,
    kSecPreferencesDomainCommon,
    kSecPreferencesDomainAlternate } SecPreferencesDomain;
```

Constants

`kSecPreferencesDomainUser`

Indicates the user preference domain preferences.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecPreferencesDomainSystem`

Indicates the system or daemon preference domain preferences.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecPreferencesDomainCommon`

Indicates the preferences are common to everyone.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecPreferencesDomainAlternate`

Indicates an alternate preference domain preferences.

Available in Mac OS X v10.3 through Mac OS X v10.3.

Declared in `SecKeychain.h`.

Discussion

A preference domain is a set of security-related preferences, such as the default keychain and the current keychain search list. The default preference domain for system daemons (that is, for daemons running in the root session) is the system domain. The default preference domain for all other programs is the user domain. A common preference appears for all users and the system; for example, if you add a keychain to the keychain search list using `kSecPreferencesDomainCommon` for the preference domain, the keychain is added to the search list for all users and the system.

Keychain Protocol Type Constants

Defines the protocol type associated with an AppleShare or Internet password.

```
typedef FourCharCode SecProtocolType;
enum
{
    kSecProtocolTypeFTP         = 'ftp ',
    kSecProtocolTypeFTPAccount  = 'ftpa',
    kSecProtocolTypeHTTP        = 'http',
    kSecProtocolTypeIRC         = 'irc ',
    kSecProtocolTypeNNTP        = 'nntp',
    kSecProtocolTypePOP3        = 'pop3',
    kSecProtocolTypeSMTP        = 'smtp',
    kSecProtocolTypeSOCKS       = 'sox ',
    kSecProtocolTypeIMAP        = 'imap',
    kSecProtocolTypeLDAP        = 'ldap',
    kSecProtocolTypeAppleTalk   = 'atlk',
    kSecProtocolTypeAFP         = 'afp ',
    kSecProtocolTypeTelnet      = 'teln',
    kSecProtocolTypeSSH         = 'ssh ',
    kSecProtocolTypeFTPS        = 'ftps',
    kSecProtocolTypeHTTPS       = 'htps',
    kSecProtocolTypeHTTPProxy   = 'htpx',
    kSecProtocolTypeHTTPSProx   = 'htsx',
    kSecProtocolTypeFTPProxy    = 'ftpx',
    kSecProtocolTypeSMB         = 'smb ',
    kSecProtocolTypeRTSP        = 'rtsp',
    kSecProtocolTypeRTSPProxy   = 'rtsx',
    kSecProtocolTypeDAAP        = 'daap',
    kSecProtocolTypeEPPC        = 'eppc',
    kSecProtocolTypeIPP         = 'ipp ',
    kSecProtocolTypeNNTPS       = 'ntps',
    kSecProtocolTypeLDAPS       = 'ldps',
    kSecProtocolTypeTelnetS     = 'tels',
    kSecProtocolTypeIMAPS       = 'imps',
    kSecProtocolTypeIRCS        = 'ircs',
    kSecProtocolTypePOP3S       = 'pops'
};
```

Constants

`kSecProtocolTypeFTP`

Indicates FTP.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeFTPAccount`

Indicates a client side FTP account. The usage of this constant is deprecated as of Mac OS X v10.3.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeHTTP`

Indicates HTTP.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeIRC`

Indicates IRC.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeNNTP`
Indicates NNTP.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypePOP3`
Indicates POP3.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeSMTP`
Indicates SMTP.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeSOCKS`
Indicates SOCKS.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeIMAP`
Indicates IMAP.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeLDAP`
Indicates LDAP.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeAppleTalk`
Indicates AFP over AppleTalk.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeAFP`
Indicates AFP over TCP.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeTelnet`
Indicates Telnet.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeSSH`
Indicates SSH.
Available in Mac OS X v10.2 and later.
Declared in `SecKeychain.h`.

`kSecProtocolTypeFTPS`
Indicates FTP over TLS/SSL. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeHTTPS`
Indicates HTTP over TLS/SSL. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeHTTPProxy`
Indicates HTTP proxy. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeHTTPSProxy`
Indicates HTTPS proxy. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeFTPProxy`
Indicates FTP proxy. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeSMB`
Indicates SMB. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeRTSP`
Indicates RTSP. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeRTSPProxy`
Indicates RTSP proxy. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeDAAP`
Indicates DAAP. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeEPPC`
Indicates Remote Apple Events. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeIPP`
Indicates IPP. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeNNTPS`
Indicates NNTP over TLS/SSL. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeLDAPS`
Indicates LDAP over TLS/SSL. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeTelnetS`
Indicates Telnet over TLS/SSL. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeIMAPS`
Indicates IMAP4 over TLS/SSL. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypeIRCS`
Indicates IRC over TLS/SSL. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

`kSecProtocolTypePOP3S`
Indicates POP3 over TLS/SSL. This constant is available in Mac OS X v10.3 and later.

Available in Mac OS X v10.3 and later.

Declared in `SecKeychain.h`.

Keychain Settings Version

Defines the keychain settings version.

```
#define SEC_KEYCHAIN_SETTINGS_VERS1 1
```

Constants

`SEC_KEYCHAIN_SETTINGS_VERS1`
Defines the keychain settings version.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

Keychain Status Masks

Defines the current status of a keychain.

```
typedef UInt32 SecKeychainStatus;
enum
{
    kSecUnlockStateStatus        = 1,
    kSecReadPermStatus           = 2,
    kSecWritePermStatus          = 4
};
```

Constants

`kSecUnlockStateStatus`

Indicates the keychain is unlocked.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecReadPermStatus`

Indicates the keychain is readable.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

`kSecWritePermStatus`

Indicates the keychain is writable.

Available in Mac OS X v10.2 and later.

Declared in `SecKeychain.h`.

Discussion

You can use these masks in combination. For example, a keychain may be both readable and writable.

Result Codes

The most common result codes returned by Keychain Services are listed in the table below. The assigned error space for Keychain Services is discontinuous: –25240 through –25279 and –25290 through –25329. Keychain Item Services may also return `noErr` (0) or `paramErr` (–50), or CSSM result codes (see *Common Security: CDSA and CSSM, version 2 (with corrigenda)* from The Open Group (http://www.opengroup.org/security/cdsa.htm)).

Result Code	Value	Description
`errSecNotAvailable`	–25291	No trust results are available. Available in Mac OS X v10.2 and later.
`errSecReadOnly`	–25292	Read only error. Available in Mac OS X v10.2 and later.
`errSecAuthFailed`	–25293	Authorization/Authentication failed. Available in Mac OS X v10.2 and later.
`errSecNoSuchKeychain`	–25294	The keychain does not exist. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
errSecInvalidKeychain	–25295	The keychain is not valid. Available in Mac OS X v10.2 and later.
errSecDuplicateKeychain	–25296	A keychain with the same name already exists. Available in Mac OS X v10.2 and later.
errSecDuplicateCallback	–25297	More than one callback of the same name exists. Available in Mac OS X v10.2 and later.
errSecInvalidCallback	–25298	The callback is not valid. Available in Mac OS X v10.2 and later.
errSecDuplicateItem	–25299	The item already exists. Available in Mac OS X v10.2 and later.
errSecItemNotFound	–25300	The item cannot be found. Available in Mac OS X v10.2 and later.
errSecBufferTooSmall	–25301	The buffer is too small. Available in Mac OS X v10.2 and later.
errSecDataTooLarge	–25302	The data is too large for the particular data type. Available in Mac OS X v10.2 and later.
errSecNoSuchAttr	–25303	The attribute does not exist. Available in Mac OS X v10.2 and later.
errSecInvalidItemRef	–25304	The item reference is invalid. Available in Mac OS X v10.2 and later.
errSecInvalidSearchRef	–25305	The search reference is invalid. Available in Mac OS X v10.2 and later.
errSecNoSuchClass	–25306	The keychain item class does not exist. Available in Mac OS X v10.2 and later.
errSecNoDefaultKeychain	–25307	A default keychain does not exist. Available in Mac OS X v10.2 and later.
errSecInteractionNotAllowed	–25308	Interaction with the Security Server is not allowed. Available in Mac OS X v10.2 and later.
errSecReadOnlyAttr	–25309	The attribute is read only. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
errSecWrongSecVersion	–25310	The version is incorrect. Available in Mac OS X v10.2 and later.
errSecKeySizeNotAllowed	–25311	The key size is not allowed. Available in Mac OS X v10.2 and later.
errSecNoStorageModule	–25312	There is no storage module available. Available in Mac OS X v10.2 and later.
errSecNoCertificateModule	–25313	There is no certificate module available. Available in Mac OS X v10.2 and later.
errSecNoPolicyModule	–25314	There is no policy module available. Available in Mac OS X v10.2 and later.
errSecInteractionRequired	–25315	User interaction is required. Available in Mac OS X v10.2 and later.
errSecDataNotAvailable	–25316	The data is not available. Available in Mac OS X v10.2 and later.
errSecDataNotModifiable	–25317	The data is not modifiable. Available in Mac OS X v10.2 and later.
errSecCreateChainFailed	–25318	The attempt to create a certificate chain failed. Available in Mac OS X v10.2 and later.
errSecInvalidPrefsDomain	–25319	The preference domain specified is invalid. This error is available in Mac OS X v10.3 and later. Available in Mac OS X v10.3 and later.
errSecACLNotSimple	–25240	The access control list is not in standard simple form. Available in Mac OS X v10.2 and later.
errSecPolicyNotFound	–25241	The policy specified cannot be found. Available in Mac OS X v10.2 and later.
errSecInvalidTrustSetting	–25242	The trust setting is invalid. Available in Mac OS X v10.2 and later.
errSecNoAccessForItem	–25243	The specified item has no access control. Available in Mac OS X v10.2 and later.
errSecInvalidOwnerEdit	–25244	An invalid attempt to change the owner of an item. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
errSecTrustNotAvailable	–25245	No trust results are available. Available in Mac OS X v10.3 and later.
errSecUnsupportedFormat	–25256	The specified import or export format is not supported. Available in Mac OS X v10.4 and later.
errSecUnknownFormat	–25257	The item you are trying to import has an unknown format. Available in Mac OS X v10.4 and later.
errSecKeyIsSensitive	–25258	The key must be wrapped to be exported. Available in Mac OS X v10.4 and later.
errSecMultiplePrivKeys	–25259	An attempt was made to import multiple private keys. Available in Mac OS X v10.4 and later.
errSecPassphraseRequired	–25260	A password is required for import or export. Available in Mac OS X v10.4 and later.

Document Revision History

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

Date	Notes
2008-11-19	Added Keychain Services API for iPhone OS.
2005-04-29	Added attribute constants for key items and made minor editing corrections.
2004-08-20	Minor editing corrections.
2004-06-28	Added functions, constants, and data types for exporting and importing keychain items..
2004-05-27	Added information about access controls.
	Added section "Managing Trusted Applications" (page 11)
	Added information about access controls to other functions as appropriate.
	Added "Authorization Tag Type Constants" (page 72) for keychain items.
2003-10-08	Added keychain item class constants for keys.
2003-07-30	Added Mac OS X v10.3 API.
2003-06-09	First version of this document.

Document Revision History

Index

K

S