Keychain Services Reference

Security

2010-09-01

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

Contents

Keychain Services Reference

Framework: Security/Security.h

Declared in SecItem.h

Overview

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

Functions by Task

Using Keychain Item Search Dictionaries

For this interface, keychain items are found or defined by a CFDictionary of key-value pairs. Each key in the dictionary identifies one attribute of the keychain item, or a search option. For example, you can use the `kSecClass` key to specify that the keychain item is an Internet password, that it has a specific creation date, that it is for the HTTPS protocol, and that only the first match found should be returned. The keys that can be used for this purpose and the possible values for each key are listed in the "Keychain Services Constants" (page 11) section.

See the discussion section of the `SecItemCopyMatching` (page 7) function for information about how to construct a keychain-item search dictionary.

`SecItemCopyMatching` (page 7)
: Returns one or more keychain items that match a search query.

`SecItemAdd` (page 6)
: Adds one or more items to a keychain.

`SecItemUpdate` (page 10)
: Modifies items that match a search query.

`SecItemDelete` (page 9)
: Deletes items that match a search query.

Functions

SecItemAdd

Adds one or more items to a keychain.

```
OSStatus SecItemAdd (
   CFDictionaryRef attributes,
   CFTypeRef *result
);
```

Parameters

attributes

A dictionary containing an item class key-value pair ("Keychain Item Class Keys and Values" (page 11)) and optional attribute key-value pairs ("Attribute Item Keys and Values" (page 14)) specifying the item's attribute values.

result

On return, a reference to the newly added items. The exact type of the result is based on the values supplied in attributes, as discussed below. Pass `NULL` if this result is not required.

Return Value

A result code. See "Keychain Services Result Codes" (page 33). Call `SecCopyErrorMessageString` (Mac OS X only) to get a human-readable string explaining the result.

Discussion

You specify attributes defining an item by adding key-value pairs to the attributes dictionary. To add multiple items to a keychain at once use the `kSecUseItemList` key (see section "Item List Key" (page 31)) with an array of items as its value. This is currently only supported for non-password items.

If you want the new keychain item to be shared among multiple applications, include the `kSecAttrAccessGroup` (page 21) key in the attributes dictionary. The value of this key must be the name of a keychain access group to which all of the programs that will share this item belong.

When you use Xcode to create an application, Xcode adds an application-identifier entitlement to the application bundle. Keychain Services uses this entitlement to grant the application access to its own keychain items. You can also add a keychain-access-groups entitlement to the application and, in the entitlement property list file, specify an array of keychain access groups to which the application belongs. The property list file can have any name you like (for example, `keychain-access-groups.plist`). The Xcode build variable `CODE_SIGN_ENTITLEMENTS` should contain the `SRCROOT` relative path to the entitlement property list file. The property list file itself should be a dictionary with a top-level key called `keychain-access-groups` whose value is an array of strings. If you add such a property-list file to the application bundle, then the access group corresponding to the application-identifier entitlement is treated as the last element in the access groups array. If you do not include the `kSecAttrAccessGroup` (page 21) key in the attributes dictionary when you call the `SecItemAdd` function to add an item to the keychain, the function uses the first access group in the array by default. If there is no `kSecAttrAccessGroup` key in the attributes dictionary and there is no keychain-access-groups entitlement in the application bundle, then the access group of a newly created item is the value of the application-identifier entitlement.

For example, a development group in Apple might have the ID:

```
659823F3DC53.com.apple
```

and the application identifiers of their two applications might be:

`659823F3DC53.com.apple.oneappleapp` and

`659823F3DC53.com.apple.twoappleapp`

If both applications add a keychain-access-groups entitlement with one value in the array of access groups:

`659823F3DC53.com.apple.netaccount`

then both applications would add new keychain items to the `659823F3DC53.com.apple.netaccount` access group by default and both applications would have access to keychain items in that group. In addition, each application would still have access to its own private keychain items: OneAppleApp would have access to items in keychain access group `659823F3DC53.com.apple.oneappleapp` and TwoAppleApp would have access to items in `659823F3DC53.com.apple.twoappleapp`.

Return types ("Search Results Constants" (page 31)) are specified as follows:

- To obtain the data of the added item as an object of type `CFDataRef`, specify the return type key `kSecReturnData` with a value of `kCFBooleanTrue`.
- To obtain all the attributes of the added item as objects of type `CFDictionaryRef`, specify `kSecReturnAttributes` with a value of `kCFBooleanTrue`.
- To obtain a reference to the added item of type `SecKeychainItemRef`, `SecKeyRef`, `SecCertificateRef`, or `SecIdentityRef`), specify `kSecReturnRef` with a value of `kCFBooleanTrue`. This is the default behavior if a return type is not explicitly specified.
- To obtain a persistent reference to the added item (an object of type `CFDataRef`), specify `kSecReturnPersistentRef` with a value of `kCFBooleanTrue`. Note that unlike normal references, a persistent reference may be stored on disk or passed between processes.
- If more than one of these return types is specified, the result is returned as an object of type `CFDictionaryRef` containing all the requested data.

Availability
Available in iOS 2.0 and later.

Related Sample Code
CryptoExercise
GenericKeychain

Declared In
`SecItem.h`

SecItemCopyMatching

Returns one or more keychain items that match a search query.

```
OSStatus SecItemCopyMatching (
   CFDictionaryRef query,
   CFTypeRef *result
);
```

Parameters

query

A dictionary containing an item class specification ("Keychain Item Class Keys and Values" (page 11)) and optional attributes for controlling the search. See "Keychain Services Constants" (page 11) for a description of currently defined search attributes.

result

On return, a reference to the found items. The exact type of the result is based on the search attributes supplied in the query, as discussed below.

Return Value

A result code. See "Keychain Services Result Codes" (page 33). Call `SecCopyErrorMessageString` (Mac OS X only) to get a human-readable string explaining the result.

Discussion

You specify attributes defining a search by adding key-value pairs to the query dictionary.

A typical query consists of:

- The class key ("Item Class Key Constant" (page 11)) and a class value constant ("Item Class Value Constants" (page 11)), which specifies the class of items for which to search.
- One or more attribute key-value pairs ("Attribute Item Keys and Values" (page 14)), which specify the attribute data to be matched.
- One or more search key-value pairs ("Search Keys" (page 29)), which specify values that further refine the search.
- A return-type key-value pair ("Search Results Constants" (page 31)), specifying the type of results you desire.

Return types ("Search Results Constants" (page 31)) are specified as follows:

- To obtain a reference (of type `CFDataRef`) to the data of a matching item, specify `kSecReturnData` with a value of `kCFBooleanTrue`.
- To obtain a dictionary (of type `CFDictionaryRef`) containing the attributes of a matching item, specify `kSecReturnAttributes` with a value of `kCFBooleanTrue`.
- To obtain a reference (of type `SecKeychainItemRef`, `SecKeyRef`, `SecCertificateRef`, or `SecIdentityRef`) to a matching item, specify `kSecReturnRef` with a value of `kCFBooleanTrue`.
- To obtain a persistent reference (of type `CFDataRef`) to a matching item, specify `kSecReturnPersistentRef` with a value of `kCFBooleanTrue`. Note that unlike normal references, a persistent reference may be stored on disk or passed between processes.
- If more than one of these return types is specified, the result is returned as a dictionary (that is, an object of type `CFDictionaryRef`) containing all the requested data.

By default, this function returns only the first match found. To obtain more than one matching item at a time, specify the search key `kSecMatchLimit` with a value greater than `1`. The result will be an object of type `CFArrayRef` containing up to that number of matching items.

By default, this function searches for items in the keychain. To instead provide your own set of items to be filtered by this search query, specify the search key `kSecMatchItemList` with a value that consists of an object of type `CFArrayRef` referencing an array that contains items of type either `SecKeychainItemRef`, `SecKeyRef`, `SecCertificateRef`, or `SecIdentityRef`. The objects in the provided array must all be of the same type.

To convert from persistent item references to normal item references, specify the search key `kSecMatchItemList` with a value that consists of an object of type `CFArrayRef` referencing an array containing one or more elements of type `CFDataRef` (the persistent references), and a return-type key of `kSecReturnRef` whose value is `kCFBooleanTrue`. The objects in the provided array must all be of the same type.

When you use Xcode to create an application, Xcode adds an application-identifier entitlement to the application bundle. Keychain Services uses this entitlement to grant the application access to its own keychain items. You can also add a keychain-access-groups entitlement to the application and, in the entitlement property list file, specify an array of keychain access groups to which the application belongs. The property list file can have any name you like (for example, `keychain-access-groups.plist`). The Xcode build variable `CODE_SIGN_ENTITLEMENTS` should contain the `SRCROOT` relative path to the entitlement property list file. The property list file itself should be a dictionary with a top-level key called `keychain-access-groups` whose value is an array of strings. When you call the `SecItemAdd` (page 6) function to add an item to the keychain, you can specify the access group to which that item should belong. By default, the `SecItemCopyMatching` function searches all the access groups to which the application belongs. However, you can add the `kSecAttrAccessGroup` (page 21) key to the search dictionary to specify which access group to search for keychain items.

Availability
Available in iOS 2.0 and later.

Related Sample Code
CryptoExercise
GenericKeychain

Declared In
`SecItem.h`

SecItemDelete

Deletes items that match a search query.

```
OSStatus SecItemDelete (
   CFDictionaryRef query
);
```

Parameters

query

A dictionary containing an item class specification and optional attributes for controlling the search. See "Search Keys" (page 29) for a description of currently defined search attributes.

Return Value
A result code. See "Keychain Services Result Codes" (page 33). Call `SecCopyErrorMessageString` (Mac OS X only) to get a human-readable string explaining the result.

Discussion
See the discussion section of the `SecItemCopyMatching` (page 7) function for information about how to construct a search dictionary.

By default, this function deletes all items matching the specified query. You can change this behavior by specifying a key, as follows:

- To delete an item identified by a transient reference, specify the `kSecMatchItemList` search key with a reference returned by using the `kSecReturnRef` return type key in a previous call to the `SecItemCopyMatching` (page 7) or `SecItemAdd` (page 6) functions.
- To delete an item identified by a persistent reference, specify the `kSecMatchItemList` search key with a persistent reference returned by using the `kSecReturnPersistentRef` return type key to the `SecItemCopyMatching` (page 7) or `SecItemAdd` (page 6) functions.
- If more than one of these return keys is specified, the behavior is undefined.

Availability
Available in iOS 2.0 and later.

Related Sample Code
CryptoExercise
GenericKeychain

Declared In
`SecItem.h`

SecItemUpdate

Modifies items that match a search query.

```
OSStatus SecItemUpdate (
   CFDictionaryRef query,
   CFDictionaryRef attributesToUpdate
);
```

Parameters

query

A dictionary containing an item class specification and optional attributes for controlling the search. Specify the items whose values you wish to change. See "Search Keys" (page 29) for a description of currently defined search attributes.

attributesToUpdate

A dictionary containing the attributes whose values should be changed, along with the new values. Only real keychain attributes are permitted in this dictionary (no "meta" attributes are allowed.) See "Attribute Item Keys and Values" (page 14) for a description of currently defined value attributes.

Return Value
A result code. See "Keychain Services Result Codes" (page 33). Call `SecCopyErrorMessageString` (Mac OS X only) to get a human-readable string explaining the result.

Discussion
See the discussion section of the `SecItemCopyMatching` (page 7) function for information about how to construct a search dictionary.

Availability
Available in iOS 2.0 and later.

Related Sample Code
GenericKeychain

Declared In
`SecItem.h`

Constants

Keychain Item Class Keys and Values

Constants used in a search dictionary to specify the class of items in the keychain. See `SecItemCopyMatching` (page 7) for a description of a search dictionary.

Item Class Key Constant

Key constant used to set the item class value in a search dictionary.

```
CFTypeRef kSecClass;
```

Constants
`kSecClass`

Dictionary key whose value is the item's class code.

Possible values for this key are listed in "Item Class Value Constants" (page 11).

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

Item Class Value Constants

Values used with the `kSecClass` key in a search dictionary.

```
CFTypeRef kSecClassGenericPassword;
CFTypeRef kSecClassInternetPassword;
CFTypeRef kSecClassCertificate;
CFTypeRef kSecClassKey;
CFTypeRef kSecClassIdentity;
```

Constants

`kSecClassGenericPassword`

Generic password item.

The following attribute types ("Attribute Item Keys and Values" (page 14)) can be used with an item of this type:

`kSecAttrAccessible`
`kSecAttrAccessGroup`
`kSecAttrCreationDate`
`kSecAttrModificationDate`
`kSecAttrDescription`
`kSecAttrComment`
`kSecAttrCreator`
`kSecAttrType`
`kSecAttrLabel`
`kSecAttrIsInvisible`
`kSecAttrIsNegative`
`kSecAttrAccount`
`kSecAttrService`
`kSecAttrGeneric`

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecClassInternetPassword`

Internet password item.

The following attribute types ("Attribute Item Keys and Values" (page 14)) can be used with an item of this type:

`kSecAttrAccessible`
`kSecAttrAccessGroup`
`kSecAttrCreationDate`
`kSecAttrModificationDate`
`kSecAttrDescription`
`kSecAttrComment`
`kSecAttrCreator`
`kSecAttrType`
`kSecAttrLabel`
`kSecAttrIsInvisible`
`kSecAttrIsNegative`
`kSecAttrAccount`
`kSecAttrSecurityDomain`
`kSecAttrServer`
`kSecAttrProtocol`
`kSecAttrAuthenticationType`
`kSecAttrPort`
`kSecAttrPath`

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecClassCertificate`

Certificate item.

The following attribute types ("Attribute Item Keys and Values" (page 14)) can be used with an item of this type:

`kSecAttrAccessible`
`kSecAttrAccessGroup`
`kSecAttrCertificateType`
`kSecAttrCertificateEncoding`
`kSecAttrLabel`
`kSecAttrSubject`
`kSecAttrIssuer`
`kSecAttrSerialNumber`
`kSecAttrSubjectKeyID`
`kSecAttrPublicKeyHash`

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecClassKey`

Cryptographic key item.

The following attribute types ("Attribute Item Keys and Values" (page 14)) can be used with an item of this type:

`kSecAttrAccessible`
`kSecAttrAccessGroup`
`kSecAttrKeyClass`
`kSecAttrLabel`
`kSecAttrApplicationLabel`
`kSecAttrIsPermanent`
`kSecAttrApplicationTag`
`kSecAttrKeyType`
`kSecAttrKeySizeInBits`
`kSecAttrEffectiveKeySize`
`kSecAttrCanEncrypt`
`kSecAttrCanDecrypt`
`kSecAttrCanDerive`
`kSecAttrCanSign`
`kSecAttrCanVerify`
`kSecAttrCanWrap`
`kSecAttrCanUnwrap`

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecClassIdentity`

Identity item.

An identity is a certificate together with its associated private key. Because an identity is the combination of a private key and a certificate, this class shares attributes of both `kSecClassKey` and `kSecClassCertificate`.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

Attribute Item Keys and Values

You use keys in a search dictionary to specify the keychain items for which to search. You can specify a combination of item attributes and search attributes (see "Search Keys" (page 29)) when looking for matching items with the `SecItemCopyMatching` (page 7) function. This section lists all the keys that specify keychain item attributes. The description of each item indicates what the possible values are for that key. In a few cases, the programming interface provides a set of constants that you can use as values for a specific key. Those value constants are also in this section, following the descriptions of the keys.

Attribute Item Keys

Each type of keychain item can have a number of attributes describing that item. For the possible types of keychain item and the attributes that can be specified for each, see "Keychain Item Class Keys and Values" (page 11).

```
CFTypeRef kSecAttrAccessible;
CFTypeRef kSecAttrCreationDate;
CFTypeRef kSecAttrModificationDate;
CFTypeRef kSecAttrDescription;
CFTypeRef kSecAttrComment;
CFTypeRef kSecAttrCreator;
CFTypeRef kSecAttrType;
CFTypeRef kSecAttrLabel;
CFTypeRef kSecAttrIsInvisible;
CFTypeRef kSecAttrIsNegative;
CFTypeRef kSecAttrAccount;
CFTypeRef kSecAttrService;
CFTypeRef kSecAttrGeneric;
CFTypeRef kSecAttrSecurityDomain;
CFTypeRef kSecAttrServer;
CFTypeRef kSecAttrProtocol;
CFTypeRef kSecAttrAuthenticationType;
CFTypeRef kSecAttrPort;
CFTypeRef kSecAttrPath;
CFTypeRef kSecAttrSubject;
CFTypeRef kSecAttrIssuer;
CFTypeRef kSecAttrSerialNumber;
CFTypeRef kSecAttrSubjectKeyID;
CFTypeRef kSecAttrPublicKeyHash;
CFTypeRef kSecAttrCertificateType;
CFTypeRef kSecAttrCertificateEncoding;
CFTypeRef kSecAttrKeyClass;
CFTypeRef kSecAttrApplicationLabel;
CFTypeRef kSecAttrIsPermanent;
CFTypeRef kSecAttrApplicationTag;
CFTypeRef kSecAttrKeyType;
CFTypeRef kSecAttrKeySizeInBits;
CFTypeRef kSecAttrEffectiveKeySize;
CFTypeRef kSecAttrCanEncrypt;
CFTypeRef kSecAttrCanDecrypt;
CFTypeRef kSecAttrCanDerive;
CFTypeRef kSecAttrCanSign;
CFTypeRef kSecAttrCanVerify;
CFTypeRef kSecAttrCanWrap;
CFTypeRef kSecAttrCanUnwrap;
CFTypeRef kSecAttrAccessGroup;
```

Constants

`kSecAttrAccessible`

A `CFTypeRef` (opaque) value that indicates when your application needs access to the data in a keychain item. You should choose the most restrictive option that meets your application's needs so that iOS can protect that item to the greatest extent possible. For a list of possible values, see "Keychain Item Accessibility Constants" (page 27).

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrCreationDate`
Creation date key.

The corresponding value is of type `CFDateRef` and represents the date the item was created. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrModificationDate`
Modification date key.

The corresponding value is of type `CFDateRef` and represents the last time the item was updated. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrDescription`
Description attribute key.

The corresponding value is of type `CFStringRef` and specifies a user-visible string describing this kind of item (for example, "Disk image password").

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrComment`
Comment attribute key.

The corresponding value is of type `CFStringRef` and contains the user-editable comment for this item.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCreator`
Creator attribute key.

The corresponding value is of type `CFNumberRef` and represents the item's creator. This number is the unsigned integer representation of a four-character code (for example, `'aCrt'`).

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrType`
Type attribute key.

The corresponding value is of type `CFNumberRef` and represents the item's type. This number is the unsigned integer representation of a four-character code (for example, 'aTyp').

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrLabel`
Label attribute key.

The corresponding value is of type `CFStringRef` and contains the user-visible label for this item.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrIsInvisible`
Invisible attribute key.

The corresponding value is of type `CFBooleanRef` and is `kCFBooleanTrue` if the item is invisible (that is, should not be displayed).

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrIsNegative`
Negative attribute key.

The corresponding value is of type `CFBooleanRef` and 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.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccount`
Account attribute key.

The corresponding value is of type `CFStringRef` and contains an account name. Items of class `kSecClassGenericPassword` and `kSecClassInternetPassword` have this attribute.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrService`
Service attribute key.

The corresponding value is a string of type `CFStringRef` that represents the service associated with this item. Items of class `kSecClassGenericPassword` have this attribute.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrGeneric`
Generic attribute key.

The corresponding value is of type `CFDataRef` and contains a user-defined attribute. Items of class `kSecClassGenericPassword` have this attribute.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrSecurityDomain`
Security domain attribute key.

The corresponding value is of type `CFStringRef` and represents the Internet security domain. Items of class `kSecClassInternetPassword` have this attribute.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrServer`
Server attribute key.

The corresponding value is of type `CFStringRef` and contains the server's domain name or IP address. Items of class `kSecClassInternetPassword` have this attribute.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocol`
Protocol attribute key.

The corresponding value is of type `CFNumberRef` and denotes the protocol for this item (see "Protocol Values" (page 22)). Items of class `kSecClassInternetPassword` have this attribute.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrAuthenticationType`
Authentication type attribute key.

The corresponding value is of type `CFNumberRef` and denotes the authentication scheme for this item (see "Authentication Type Values" (page 25)).

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrPort`
Port attribute key.

The corresponding value is of type `CFNumberRef` and represents an Internet port number. Items of class `kSecClassInternetPassword` have this attribute.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrPath`
Path attribute key.

The corresponding value is of type `CFStringRef` and represents a path, typically the path component of the URL. Items of class `kSecClassInternetPassword` have this attribute.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrSubject`
Subject attribute key.

The corresponding value is of type `CFDataRef` and contains the X.500 subject name of a certificate. Items of class `kSecClassCertificate` have this attribute. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrIssuer`
Issuer attribute key.

The corresponding value is of type `CFDataRef` and contains the X.500 issuer name of a certificate. Items of class `kSecClassCertificate` have this attribute. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrSerialNumber`
Serial number attribute key.

The corresponding value is of type `CFDataRef` and contains the serial number data of a certificate. Items of class `kSecClassCertificate` have this attribute. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrSubjectKeyID`
Subject key ID attribute key.

The corresponding value is of type `CFDataRef` and contains the subject key ID of a certificate. Items of class `kSecClassCertificate` have this attribute. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrPublicKeyHash`
Public key hash attribute key.

The corresponding value is of type `CFDataRef` and contains the hash of a certificate's public key. Items of class `kSecClassCertificate` have this attribute. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCertificateType`
Certificate type attribute key.

The corresponding value is of type `CFNumberRef` and denotes the certificate type (see the `CSSM_CERT_TYPE` enumeration in cssmtype.h). Items of class `kSecClassCertificate` have this attribute. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCertificateEncoding`
Certificate encoding attribute key.

The corresponding value is of type `CFNumberRef` and denotes the certificate encoding (see the `CSSM_CERT_ENCODING` enumeration in cssmtype.h). Items of class `kSecClassCertificate` have this attribute. Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrKeyClass`
Key class attribute key.

The corresponding value is of type `CFTypeRef` and specifies a type of cryptographic key. Possible values are listed in "Key Class Values" (page 27). Read only.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrApplicationLabel`
Application label attribute key.

The corresponding value is of type `CFStringRef` and contains a label for this item. This attribute is different from the `kSecAttrLabel` attribute, which is intended to be human-readable. This attribute is used to look up a key programmatically; in particular, for keys of class `kSecAttrKeyClassPublic` and `kSecAttrKeyClassPrivate`, the value of this attribute is the hash of the public key.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrIsPermanent`
Permanence attribute key.

The corresponding value is of type `CFBooleanRef` and indicates whether this cryptographic key is to be stored permanently.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrApplicationTag`
Private tag attribute key.

The corresponding value is of type `CFDataRef` and contains private tag data.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrKeyType`
Algorithm attribute key.

The corresponding value is of type `CFNumberRef` and indicates the algorithm associated with this cryptographic key (see the `CSSM_ALGORITHMS` enumeration in cssmtype.h and "Key Type Value" (page 27)).

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrKeySizeInBits`
Number of bits attribute key.

The corresponding value is of type `CFNumberRef` and indicates the total number of bits in this cryptographic key. Compare with `kSecAttrEffectiveKeySize`.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrEffectiveKeySize`
Effective number of bits attribute key.

The corresponding value is of type `CFNumberRef` and indicates the effective number of bits in this cryptographic key. For example, a DES key has a `kSecAttrKeySizeInBits` of 64, but a `kSecAttrEffectiveKeySize` of 56 bits.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCanEncrypt`
Encryption attribute key.

The corresponding value is of type `CFBooleanRef` and indicates whether this cryptographic key can be used to encrypt data.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCanDecrypt`
Decryption attribute key.

The corresponding value is of type `CFBooleanRef` and indicates whether this cryptographic key can be used to decrypt data.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCanDerive`

Derivation attribute key.

The corresponding value is of type `CFBooleanRef` and indicates whether this cryptographic key can be used to derive another key.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCanSign`

Signature attribute key.

The corresponding value is of type `CFBooleanRef` and indicates whether this cryptographic key can be used to create a digital signature.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCanVerify`

Signature verification attribute key.

The corresponding value is of type `CFBooleanRef` and indicates whether this cryptographic key can be used to verify a digital signature.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCanWrap`

Wrap attribute key.

The corresponding value is of type `CFBooleanRef` and indicates whether this cryptographic key can be used to wrap another key.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrCanUnwrap`

Unwrap attribute key.

The corresponding value is of type `CFBooleanRef` and indicates whether this cryptographic key can be used to unwrap another key.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessGroup`

Access group key.

The corresponding value is of type `CFStringRef` and indicates which access group an item is in. Access groups can be used to share keychain items among two or more applications. For applications to share a keychain item, the applications must have a common access group listed in their keychain-access-groups entitlement, and the application adding the shared item to the keychain must specify this shared access-group name as the value for this key in the dictionary passed to the `SecItemAdd` (page 6) function.

An application can be a member of any number of access groups. By default, the `SecItemUpdate` (page 10), `SecItemDelete` (page 9), and `SecItemCopyMatching` (page 7) functions search all the access groups an application is a member of. Include this key in the search dictionary for these functions to specify which access group is searched.

A keychain item can be in only a single access group.

Available in iOS 3.0 and later.

Declared in `SecItem.h`.

Discussion
These predefined item attribute keys are used to get or set values in a dictionary. Not all attributes apply to each item class.

Protocol Values

Values that can be used with the `kSecAttrProtocol` attribute key.

```
CFTypeRef kSecAttrProtocolFTP;
CFTypeRef kSecAttrProtocolFTPAccount;
CFTypeRef kSecAttrProtocolHTTP;
CFTypeRef kSecAttrProtocolIRC;
CFTypeRef kSecAttrProtocolNNTP;
CFTypeRef kSecAttrProtocolPOP3;
CFTypeRef kSecAttrProtocolSMTP;
CFTypeRef kSecAttrProtocolSOCKS;
CFTypeRef kSecAttrProtocolIMAP;
CFTypeRef kSecAttrProtocolLDAP;
CFTypeRef kSecAttrProtocolAppleTalk;
CFTypeRef kSecAttrProtocolAFP;
CFTypeRef kSecAttrProtocolTelnet;
CFTypeRef kSecAttrProtocolSSH;
CFTypeRef kSecAttrProtocolFTPS;
CFTypeRef kSecAttrProtocolHTTPS;
CFTypeRef kSecAttrProtocolHTTPProxy;
CFTypeRef kSecAttrProtocolHTTPSProxy;
CFTypeRef kSecAttrProtocolFTPProxy;
CFTypeRef kSecAttrProtocolSMB;
CFTypeRef kSecAttrProtocolRTSP;
CFTypeRef kSecAttrProtocolRTSPProxy;
CFTypeRef kSecAttrProtocolDAAP;
CFTypeRef kSecAttrProtocolEPPC;
CFTypeRef kSecAttrProtocolIPP;
CFTypeRef kSecAttrProtocolNNTPS;
CFTypeRef kSecAttrProtocolLDAPS;
CFTypeRef kSecAttrProtocolTelnetS;
CFTypeRef kSecAttrProtocolIMAPS;
CFTypeRef kSecAttrProtocolIRCS;
CFTypeRef kSecAttrProtocolPOP3S;
```

Constants

`kSecAttrProtocolFTP`

FTP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolFTPAccount`

A client side FTP account.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolHTTP`
HTTP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolIRC`
IRC protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolNNTP`
NNTP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolPOP3`
POP3 protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolSMTP`
SMTP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolSOCKS`
SOCKS protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolIMAP`
IMAP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolLDAP`
LDAP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolAppleTalk`
AFP over AppleTalk.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolAFP`
AFP over TCP.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolTelnet`
Telnet protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolSSH`
SSH protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolFTPS`
FTP over TLS/SSL.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolHTTPS`
HTTP over TLS/SSL.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolHTTPProxy`
HTTP proxy.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolHTTPSProxy`
HTTPS proxy.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolFTPProxy`
FTP proxy.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolSMB`
SMB protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolRTSP`
RTSP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolRTSPProxy`
RTSP proxy.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolDAAP`
DAAP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolEPPC`
Remote Apple Events.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolIPP`
IPP protocol.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolNNTPS`
NNTP over TLS/SSL.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolLDAPS`
LDAP over TLS/SSL.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolTelnetS`
Telnet over TLS/SSL.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolIMAPS`
IMAP over TLS/SSL.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolIRCS`
IRC over TLS/SSL.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecAttrProtocolPOP3S`
POP3 over TLS/SSL.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

Authentication Type Values

Values that can be used with the `kSecAttrAuthenticationType` attribute key.

```
CFTypeRef kSecAttrAuthenticationTypeNTLM;
CFTypeRef kSecAttrAuthenticationTypeMSN;
CFTypeRef kSecAttrAuthenticationTypeDPA;
CFTypeRef kSecAttrAuthenticationTypeRPA;
CFTypeRef kSecAttrAuthenticationTypeHTTPBasic;
CFTypeRef kSecAttrAuthenticationTypeHTTPDigest;
CFTypeRef kSecAttrAuthenticationTypeHTMLForm;
CFTypeRef kSecAttrAuthenticationTypeDefault;
```

Constants

`kSecAttrAuthenticationTypeNTLM`
Windows NT LAN Manager authentication.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrAuthenticationTypeMSN`
Microsoft Network default authentication.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrAuthenticationTypeDPA`
Distributed Password authentication.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrAuthenticationTypeRPA`
Remote Password authentication.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrAuthenticationTypeHTTPBasic`
HTTP Basic authentication.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrAuthenticationTypeHTTPDigest`
HTTP Digest Access authentication.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrAuthenticationTypeHTMLForm`
HTML form based authentication.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrAuthenticationTypeDefault`
The default authentication type.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

Key Class Values

Values that can be used with the `kSecAttrKeyClass` attribute key.

```
CFTypeRef kSecAttrKeyClassPublic;
CFTypeRef kSecAttrKeyClassPrivate;
CFTypeRef kSecAttrKeyClassSymmetric;
```

Constants

`kSecAttrKeyClassPublic`
A public key of a public-private pair.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrKeyClassPrivate`
A private key of a public-private pair.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrKeyClassSymmetric`
A private key used for symmetric-key encryption and decryption.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

Key Type Value

A values that can be used with the `kSecAttrKeyType` attribute key.

```
CFTypeRef kSecAttrKeyTypeRSA;
```

Constants

`kSecAttrKeyTypeRSA`
RSA algorithm.
Available in iOS 2.0 and later.
Declared in `SecItem.h`.

`kSecAttrKeyTypeEC`
Elliptic curve algorithm.
Available in iOS 4.0 and later.
Declared in `SecItem.h`.

Keychain Item Accessibility Constants

These constants are legal values for `kSecAttrAccessible` (page 15) used for determining when a keychain item should be readable.

```
CFTypeRef kSecAttrAccessibleWhenUnlocked;
CFTypeRef kSecAttrAccessibleAfterFirstUnlock;
CFTypeRef kSecAttrAccessibleAlways;
CFTypeRef kSecAttrAccessibleWhenUnlockedThisDeviceOnly;
CFTypeRef kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly;
CFTypeRef kSecAttrAccessibleAlwaysThisDeviceOnly;
```

Constants

`kSecAttrAccessibleAfterFirstUnlock`

The data in the keychain item cannot be accessed after after a restart until the device has been unlocked once by the user. After the first unlock, the data remains accessible until the next restart. This is recommended for items that need to be accessed by background applications. Items with this attribute migrate to a new device when using encrypted backups.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly`

The data in the keychain item cannot be accessed after after a restart until the device has been unlocked once by the user. After the first unlock, the data remains accessible until the next restart. This is recommended for items that need to be accessed by background applications. Items with this attribute *do not* migrate to a new device or new installation. Thus, after restoring from a backup, these items will not be present.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleAlways`

The data in the keychain item can always be accessed regardless of whether the device is locked. This is not recommended for application use. Items with this attribute migrate to a new device when using encrypted backups.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleAlwaysThisDeviceOnly`

The data in the keychain item can always be accessed regardless of whether the device is locked. This is not recommended for application use. Items with this attribute *do not* migrate to a new device or new installation. Thus, after restoring from a backup, these items will not be present.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleWhenUnlocked`

The data in the keychain item can be accessed only while the device is unlocked by the user. This is recommended for items that need to be accesible only while the application is in the foreground. Items with this attribute migrate to a new device when using encrypted backups.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

`kSecAttrAccessibleWhenUnlockedThisDeviceOnly`

The data in the keychain item can be accessed only while the device is unlocked by the user. This is recommended for items that need to be accesible only while the application is in the foreground. Items with this attribute *do not* migrate to a new device or new installation. Thus, after restoring from a backup, these items will not be present.

Available in iOS 4.0 and later.

Declared in `SecItem.h`.

Search Keys

Search Attribute Keys

Keys used to set search attributes in a keychain search dictionary. You can specify a combination of search attributes and item attributes (see "Attribute Item Keys and Values" (page 14)) when looking for matching items with the `SecItemCopyMatching` (page 7) function.

```
CFTypeRef kSecMatchPolicy;
CFTypeRef kSecMatchItemList;
CFTypeRef kSecMatchSearchList;
CFTypeRef kSecMatchIssuers;
CFTypeRef kSecMatchEmailAddressIfPresent;
CFTypeRef kSecMatchSubjectContains;
CFTypeRef kSecMatchCaseInsensitive;
CFTypeRef kSecMatchTrustedOnly;
CFTypeRef kSecMatchValidOnDate;
CFTypeRef kSecMatchLimit;
CFTypeRef kSecMatchLimitOne;
CFTypeRef kSecMatchLimitAll;
```

Constants

`kSecMatchPolicy`

Match policy attribute key.

The corresponding value is of type `SecPolicyRef`. If provided, returned certificates or identities must verify with this policy.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchItemList`

Item list attribute key.

To provide your own set of items to be filtered by a search query rather than searching the keychain, specify this search key in a call to the `SecItemCopyMatching` (page 7) function with a value that consists of an object of type `CFArrayRef` where the array contains either `SecKeychainItemRef`, `SecKeyRef`, `SecCertificateRef`, `SecIdentityRef`, or `CFDataRef` items. The objects in the provided array must all be of the same type.

To convert from persistent item references to normal item references, specify this search key in a call to the `SecItemCopyMatching` (page 7) function with a value of type `CFArrayRef` where the array contains one or more `CFDataRef` elements (the persistent references), and a return-type key of `kSecReturnRef` whose value is `kCFBooleanTrue`.

To delete an item identified by a transient reference, specify the `kSecMatchItemList` search key in a call to the `SecItemDelete` (page 9) function with a reference returned by using the `kSecReturnRef` return type key in a previous call to the `SecItemCopyMatching` (page 7) or `SecItemAdd` (page 6) functions.

To delete an item identified by a persistent reference, specify the `kSecMatchItemList` search key in a call to the `SecItemDelete` (page 9) function with a persistent reference returned by using the `kSecReturnPersistentRef` return type key to the `SecItemCopyMatching` (page 7) or `SecItemAdd` (page 6) functions.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchSearchList`

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchIssuers`

Issuers attribute key.

The corresponding value is of type `CFArrayRef`, where the array consists of X.500 names of type `CFDataRef`. If provided, returned certificates or identities are limited to those whose certificate chain contains one of the issuers provided in this list.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchEmailAddressIfPresent`

Email address attribute key.

The corresponding value is of type `CFStringRef` and contains an RFC822 email address. If provided, returned certificates or identities are limited to those that either contain the address or do not contain any email address.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchSubjectContains`

Subject attribute key.

The corresponding value is of type `CFStringRef`. If provided, returned certificates or identities are limited to those whose subject contains this string.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchCaseInsensitive`

Case sensitivity attribute key.

The corresponding value is of type `CFBooleanRef`. If this value is `kCFBooleanFalse`, or if this attribute is not provided, then case-sensitive string matching is performed.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchTrustedOnly`

Trusted anchor attribute key.

The corresponding value is of type `CFBooleanRef`. If this attribute is provided with A value of `kCFBooleanTrue`, only certificates that can be verified back to a trusted anchor are returned. If this value is `kCFBooleanFalse` or the attribute is not provided, then both trusted and untrusted certificates may be returned.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchValidOnDate`

Valid-on-date attribute key.

The corresponding value is of type `CFDateRef`. If provided, returned keys, certificates or identities are limited to those that are valid for the given date. Pass a value of `kCFNull` to indicate the current date.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecMatchLimit`
: Match limit attribute key.

  The corresponding value is of type `CFNumberRef`. If provided, this value specifies the maximum number of results to return. If not provided, results are limited to the first item found. For a single item, specify `kSecMatchLimitOne`. To return all matching items, specify `kSecMatchLimitAll`.

  Available in iOS 2.0 and later.

  Declared in `SecItem.h`.

`kSecMatchLimitOne`
: Results are limited to the first item found; used as a value for the `kSecMatchLimit` attribute key.

  Available in iOS 2.0 and later.

  Declared in `SecItem.h`.

`kSecMatchLimitAll`
: An unlimited number of results may be returned; used as a value for the `kSecMatchLimit` attribute key.

  Available in iOS 2.0 and later.

  Declared in `SecItem.h`.

Item List Key

Key used to specify a list of items to search or add.

```
CFTypeRef kSecUseItemList;
```

Constants

`kSecUseItemList`
: Item list key.

  The corresponding value is of type `CFArrayRef`, where the array contains either `SecKeychainItemRef`, `SecKeyRef`, `SecCertificateRef`, `SecIdentityRef`, or (for persistent item references) `CFDataRef` items. If provided, this array is treated as the set of all possible items to search (or to add if the function being called is `SecItemAdd` (page 6)). The items in the array must all be of the same type.

  Available in iOS 2.0 and later.

  Declared in `SecItem.h`.

Discussion

When this attribute is provided, no keychains are searched.

Search Results Constants

Return Type Keys

Keys used to specify the type of results that should be returned by the `SecItemCopyMatching` (page 7) or `SecItemAdd` (page 6) function.

```
CFTypeRef kSecReturnData;
CFTypeRef kSecReturnAttributes;
CFTypeRef kSecReturnRef;
CFTypeRef kSecReturnPersistentRef;
```

Constants

`kSecReturnData`

Return data attribute key.

The corresponding value is of type `CFBooleanRef`. A value of `kCFBooleanTrue` indicates that the data of an item should be returned in the form of a `CFDataRef`. For keys and password items, data is secret (encrypted) and may require the user to enter a password for access.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecReturnAttributes`

Return attributes attribute key.

The corresponding value is of type `CFBooleanRef`. A value of `kCFBooleanTrue` indicates that a dictionary of the (nonencrypted) attributes of an item should be returned in the form of a `CFDictionaryRef`.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecReturnRef`

Return reference attribute key.

The corresponding value is of type `CFBooleanRef`. A value of `kCFBooleanTrue` indicates that a reference should be returned. Depending on the item class requested, the returned references may be of type `SecKeychainItemRef`, `SecKeyRef`, `SecCertificateRef`, `SecIdentityRef`, or `CFDataRef`.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecReturnPersistentRef`

Return persistent reference attribute key.

The corresponding value is of type `CFBooleanRef`. A value of `kCFBooleanTrue` indicates that a persistent reference to an item (`CFDataRef`) should be returned.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

Discussion

You can specify zero or more of these return types. If you specify more than one of these return types, Keychain Services returns the result as a `CFDictionaryRef` reference to a dictionary whose keys are the return types and whose alues are the requested data.

Value Type Keys

Keys used in the results dictionary for `SecItemCopyMatching` (page 7) or `SecItemAdd` (page 6), indicating the type of values returned. You can specify zero or more of these types depending on the function you are calling.

```
CFTypeRef kSecValueData;
CFTypeRef kSecValueRef;
CFTypeRef kSecValuePersistentRef;
```

Constants

`kSecValueData`

Data attribute key.

The corresponding value is of type `CFDataRef`. For keys and password items, the data is secret (encrypted) and may require the user to enter a password for access.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecValueRef`

Reference attribute key.

The corresponding value, depending on the item class requested, is of type `SecKeychainItemRef`, `SecKeyRef`, `SecCertificateRef`, or `SecIdentityRef`.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

`kSecValuePersistentRef`

Persistent reference attribute key.

The corresponding value is of type `CFDataRef`. The bytes in this `CFDataRef` can be stored by the caller and used on a subsequent invocation of the application (or even a different application) to retrieve the item referenced by it.

Available in iOS 2.0 and later.

Declared in `SecItem.h`.

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
`errSecSuccess`	0	No error. Available in iOS 2.0 and later.
`errSecUnimplemented`	-4	Function or operation not implemented. Available in iOS 2.0 and later.
`errSecParam`	-50	One or more parameters passed to the function were not valid. Available in iOS 2.0 and later.

Result Code	Value	Description
`errSecAllocate`	-108	Failed to allocate memory. Available in iOS 2.0 and later.
`errSecNotAvailable`	–25291	No trust results are available. Available in iOS 2.0 and later.
`errSecAuthFailed`	–25293	Authorization/Authentication failed. Available in iOS 4.2 and later.
`errSecDuplicateItem`	–25299	The item already exists. Available in iOS 2.0 and later.
`errSecItemNotFound`	–25300	The item cannot be found. Available in iOS 2.0 and later.
`errSecInteractionNotAllowed`	–25308	Interaction with the Security Server is not allowed. Available in iOS 2.0 and later.
`errSecDecode`	-26275	Unable to decode the provided data. Available in iOS 2.0 and later.

Document Revision History

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

Date	Notes
2010-09-01	Updated for iOS 4.0. Added Keychain accessibility attributes.
2010-04-06	Corrected typos.
2009-04-27	Updated for iOS v3.0.
	Corrected some mistakes and added some constants. Added functions to Mac OS X version of document that create and read persistent references and that return a human-readable error string.
2008-11-19	Added Keychain Services API for iOS.
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 ?)
	Added information about access controls to other functions as appropriate.
	Added "Authorization Tag Type Constants" 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