Keychain Security

Application Interfaces (API)

Version 2.0
August 19, 1999

Keychain overview

A **keychain** is a secure repository for user-centric data, such as keys, passwords and certificates. The repository may be a file, a network database, a smart card, or other storage media..

The currently unlocked keychain in which items are added is also known as the **default keychain**. New items are always added to the default keychain. When an API call is made and the Keychain Manager detects there aren't any keychains available, the user is prompted to create one using the Keychain Manager UI.

The default keychain is automatically selected for the user by the Keychain Manager when an unspecified keychain is to be unlocked. The default keychain preference is stored in Internet Config and is modified by the user via the Keychain Access control panel. The default keychain is automatically configured by the Keychain Manager when a keychain is created for the first time on the user's machine. If the default keychain isn't configured (i.e. a non-existant Internet Preferences file), the user is prompted to chose among the locked keychains (first one chosen in the list).

Using the Keychain Manager

The following example demonstrates how a typical application might use the high-level keychain interfaces to store password data. Note that an explicit call to unlock the keychain is not required.

```
OSStatus StorePasswordInKeychain (ConstStr255Param password)
{
      OSStatus status;

      if (!KeychainManagerAvailable ())   // is it there?
            return ((OSStatus) MY_ERROR);

      KCItemRef   item;
      status = KCAddGenericPassword (
                        "\pMy_App_Pwd",   // service name
                        "\pBill Braskey", // account name
                        password[0],      // length of password
                        &password[1],     // pointer to password data
                        &item);
      return (status);
}
```

It is required that your application call `MaxApplZone()` to fully utilize the maximum memory available.

Keychain events

A **keychain event** is generated when one of the following occurs:

- a keychain's lock state changes
- an item is added to, updated in or removed from a keychain
- a keychain call takes a non-trivial amount of time to complete

Most applications do not need to handle keychain events. Applications which are interested in receiving keychain events must first call `KCAddCallback` to register a procedure that handles the event. What events the application is interested in can be registered as well. When an application is no longer interested in receiving keychain events, it calls `KCRemoveCallback` to unregister the callback procedure.

Idle Events

All keychain API calls are synchronous. They do not return until complete. If called within a thread, this is not normally an issue. If, however, your application is not threaded, you may want to register a callback procedure to receive keychain idle events. The callback procedure will be called periodically until the keychain call completes. The application must be sure to call `YieldToAnyThread()` or `WaitNextEvent()` when an idle event is generated. If the application does not register for the idle event, the Keychain Manager will call `YieldToAnyThread()` periodically instead.

Callback Restrictions

When a keychain event occurs, the callback routine may be called from within another process, making it unsafe to allocate memory or call toolbox routines that allocate memory. This is true for all events except the idle event. When an idle event occurs, the application is free to make any toolbox calls and perform memory allocation.

The following example demonstrates how an application might register and use a callback procedure.

```
// ---------------------------------------------------------------------------
OSStatus
RegisterMyCallbackProc (Ptr myDataPtr)
{
      OSStatus status = noErr;
      static KCCallbackUPP myCallbackUPP = nil;

      if (!myCallbackUPP)
      {
            // create a routine descriptor for the callback routine
            myCallbackUPP = NewKCCallbackProc(MyCallbackProc);
      }
      status = KCAddCallback(myCallbackUPP, kcEveryEvent, myDataPtr);
      return (status);
}

// ---------------------------------------------------------------------------
pascal void
MyCallbackProc (KCEvent keychainEvent, KCCallbackInfo* eventInfo,
                  void *userContext)
{
      MyDataPtrType myDataPtr = (MyDataPtrType) userContext;
      if (inEvent == kcIdleEvent)
      {
            YieldToAnyThread();
      }
      else if (myDataPtr != nil)
      {
            // it may not be safe to allocate or move memory here,
            // so you may want to queue the event for later processing.
            myDataPtr->event = eventInfo->keychainEvent;
            myDataPtr->keychain = eventInfo->keychain;
            myDataPtr->item  = eventInfo->item;
            myDataPtr->gotAnEvent = true;
      }
}
```

Keychain item icons

A keychain item can have an application-specific icon. To support this, the following must be true:

- `kTypeKCItemAttr` must be set to a file type for which there is a corresponding icon in the desktopDB.
- `kCreatorKCItemAttr` must be set to an appropriate application creator type.
- `kCustomIconKCItemAttr` must be set.

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.

Getting information about the Keychain Manager

```
Boolean KeychainManagerAvailable (void)
```

DESCRIPTION
Determines whether the Keychain Manager API routines are available.
(This routine is implemented as an inline function; all other Keychain Manager routines are implemented in the shared CFM library.)

```
OSStatus KCGetKeychainManagerVersion (UInt32* version)
```

DESCRIPTION
Returns the version of the Keychain Manager (an unsigned 32-bit integer) in `version`.

RESULT CODE

`noErr`	0	No error.
`errKCNotAvailable`	-25291	Keychain Manager was not loaded.

Locking and unlocking a keychain

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

`keychain`	A reference to a keychain that is to be unlocked. Pass NULL to specify the default keychain.
`password`	The password string for this keychain. Pass NULL if the password is unknown.

DESCRIPTION
Use this routine to unlock a keychain.

If `the password` is NULL, the Keychain Manager will display the unlock dialog. The authentication UI associated with the keychain about to be unlocked will be displayed as well.

If `keychain` specifies a keychain which is currently unlocked, the Unlock Keychain dialog is not displayed and `KCUnlock` returns `noErr`. If `keychain` specifies a keychain which is locked, that keychain will appear as the chosen menu item in the dialog's keychain popup menu.

If `keychain` is NIL and the default keychain is currently locked, that keychain will appear as the default choice. If the default keychain is currently unlocked, the Unlock Keychain dialog is not displayed and `KCUnlock` returns `noErr`. If at any time, an invalid password is passed to `KCUnlock,` you subsequently will not be able to unlock a keychain with a specified password until the machine is rebooted. Until then, `errKCInteractionRequired` is returned.

The memory that `keychain` occupies must be released by calling `KCReleaseKeychain` when finished with it.

RESULT CODE

`noErr`	0	No error (successfully unlocked).
`errUserCanceled`	-128	User pressed the Cancel button in the unlock dialog.
`errKCAuthFailed`	-25293	Authentication failed (too many unsuccessful retries).

errKCNoSuchKeychain	-25294	The specified keychain could not be found.
errKCInvalidKeychain	-25295	The keychain is invalid
errKCInteractionRequired	-25315	User interaction required to unlock this keychain

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

keychain A keychain reference.

DESCRIPTION
Locks the specified keychain if it is unlocked. (If keychain is NIL, the Keychain Manager locks all unlocked keychains.)

> NOTE: There is usually no need for an application to ever call KCLock. Unless your application is directly responding to a user's request for a keychain to be locked, it is recommended that you leave the keychain unlocked so that the user does not have to unlock it again in another application.

RESULT CODE

noErr	0	No error (successfully locked, or no unlocked keychain found).
errKCNoSuchKeychain	-25294	The specified keychain could not be found.
errKCInvalidKeychain	-25295	The keychain is invalid

Manually creating a reference to a keychain

NEW FOR V2.0

```
OSStatus KCMakeKCRefFromFSSpec (FSSpec* keychainFile, KCRef* keychain)
```

keychainFile	A pointer to a keychain file specification.
keychain	Returned keychain reference.

DESCRIPTION
Returns a referenece to the keychain specified by keychainFile.

The memory that keychain occupies must be released by calling KCReleaseKeychain when finished with it.

RESULT CODE

noErr	0	No error.
paramErr	-50	The keychain parameter is invalid (NULL).

NEW FOR V2.0

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

keychainAlias	A handle to an alias record of the keychain file
keychain	A returned keychain reference

DESCRIPTION
Returns a referenece to the keychain specified by the alias keychainAlias.

The memory that `keychain` occupies must be released by calling `KCReleaseKeychain` when finished with it.

RESULT CODE

`noErr`	0	No error.
`paramErr`	`-50`	Either the `keychainAlias` or `keychain` parameters are invalid (NULL).
`errKCNoSuchKeychain`	-25294	`keychainAlias` references in invalid keychain file

NEW FOR V2.0

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

`keychain`	A pointer to a keychain reference of a keychain file
`keychainAlias`	A returned alias handle to `keychain`

DESCRIPTION
Returns an alias handle to the keychain file specified by `keychain.`

The memory that `keychainAlias` occupies must be released by calling `DisposeHandle` when finished with it.

RESULT CODE

`noErr`	0	No error.
`paramErr`	`-50`	Either the `keychainAlias` or `keychain` parameters are invalid (NULL).

Working with the default keychain

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

`keychain`	A pointer to a reference of the default keychain.

DESCRIPTION
This routine returns a `KCRef` which specifies the default keychain. Your application might call this routine to obtain the name and location of the default keychain.

RESULT CODE

`noErr`	0	No error.
`errKCNoDefaultKeychain`	-25307	There is no currently default keychain.

The memory that `keychain` occupies must be released by calling `KCReleaseKeychain` when finished with it.

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

`keychain`	Specifies a reference to a keychain that will be made the default.

DESCRIPTION
This routine sets the default keychain to the keychain specified by `keychain`. If the specified keychain is locked, the user will be prompted to unlock it. Applications should normally have no need to call `KCSetDefaultKeychain`.

RESULT CODE

noErr	0	No error.
paramErr	-50	The input specification parameter was NULL.
errKCNoSuchKeychain	-25294	The specified keychain could not be found.
errKCInvalidKeychain	-25295	The specified keychain is invalid

Releasing a reference to a keychain

NEW FOR V2.0

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

keychain — Specifies the keychain reference to release.

DESCRIPTION
Releases the memory occupied by the reference that keychain points to and sets keychain to NIL.

RESULT CODE

noErr	0	No error.
paramErr	-50	The keychain parameter is invalid (NIL).

Creating a new keychain

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

password — The password string which will be used to protect the new keychain.
keychain — A keychain reference that specifies the keychain to create.
- NULL means create the keychain in the Keychains folder.
- a valid KCRef pointer specifies the reference of the keychain to create.
- a KCRef that points to NULL means have the Keychain Manager allocate the memory for the keychain reference of the newly created keychain and return it to the caller. The keychain will be created in the Keychains folder.

DESCRIPTION
Creates a new empty keychain optionally defined by keychain and password.
If keychain is NULL or points to NULL, the new keychain will be created in the "Keychains"folder: (SystemFolder:Preferences:Keychains). If the caller specifies a keychain and the location of that keychain is unspecified or invalid, the Keychain Manager will fill in the location to the Keychains folder. If the password is NULL, the Keychain Setup dialog will be displayed to obtain it. If the Keychain Manager uses UI to create a keychain, that keychain is automatically unlocked after creation.

RESULT CODE

noErr	0	No error (successfully created).
paramErr	-50	Invalid arguments were specified
errKCUserCanceled	-128	User pressed the Cancel button in the create keychain dialog.
errKCDuplicateKeychain	-25296	Tried to create a keychain which already exists.
errKCNoSuchKeychain	-25294	The specified keychain could not be created.
errKCInvalidKeychain	-25295	The specified keychain is invalid.

Additional errors may be returned if the keychain could not be created (for example, a file system or network error may be returned if there is no write access to the storage media).

The memory that keychain occupies must be released by calling KCReleaseKeychain when finished with it.

Getting information about a keychain

NEW FOR V2.0

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

`keychain`	Specifies a keychain whose name you want.
`keychainName`	Returned name of the keychain.

DESCRIPTION
Returns the name of `keychain` in `keychainName`.

RESULT CODE

`noErr`	0	No error.
`paramErr`	`-50`	`keychainName` must be valid (non-NULL).
`errKCInvalidKeychain`	`-25295`	Keychain is invalid

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

`keychain`	Pointer to a keychain reference (NULL specifies the default keychain).
`keychainStatus`	Returned status of the specified keychain.

DESCRIPTION
Returns status information for the specified keychain in the supplied parameter. If `keychain` is NULL, the status of the default keychain is returned.

The value returned in `keychainStatus` is a 32-bit field, the meaning of which must be determined by comparison with a list of predefined constants. Currently defined bitmask values are:

`kcUnlockState`	1	The specified keychain is unlocked if bit 0 is set.
`kcRdPerm`	2	The specified keychain is unlocked with read permission if bit 1 is set.
`kcWrPerm`	4	The specified keychain is unlocked with write permission if bit 2 is set.

RESULT CODE

`noErr`	0	No error.
`errKCNoSuchKeychain`	-25294	The specified keychain could not be found.
`errKCInvalidKeychain`	-25295	The specified keychain is invalid.

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

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

DESCRIPTION
Displays a dialog for changing the keychain name, password and settings associated with the specified keychain.

> NOTE: there is no way to supply authentication information directly to `KCChangeSettings`; for security reasons. The user must always change the keychain authentication information interactively.

RESULT CODE

`noErr`	0	No error.
`errUserCanceled`	-128	User pressed the Cancel button in the settings dialog.

errKCNoDefaultKeychain	-25307	No default keychain could be found.
errKCInvalidKeychain	-25295	The specified keychain is invalid.

Enumerating available keychains

```
UInt16 KCCountKeychains (void)
```

DESCRIPTION
This function returns the number of available keychains. This number includes all keychains within the "Keychains" folder, as well as any other keychains known to the Keychain Manager.

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

index	Index value that represents which keychain to acquire.
keychain	Returned keychain based on index.

DESCRIPTION
Given an index value (from 1 to the number returned by KCCountKeychains), returns a KCRef for the keychain corresponding to that index.

RESULT CODE

noErr	0	No error.
paramErr	-50	Input keychain specification pointer was NULL.
errKCNoSuchKeychain	-25294	No keychain found (index value out of range.)

The memory that keychain occupies must be released by calling KCReleaseKeychain when finished with it.

Registering a callback procedure

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

callbackProc — A universal procedure pointer to a routine with the following interface:

```
pascal void
MyCallbackProc (  KCEvent keychainEvent,
                  KCCallbackInfo* info,
                  void *userContext )
```

eventMask	A bit mask indicating which events are to be received.
userContext	A pointer to caller-defined storage. This value will be passed to your callback.

DESCRIPTION
The specified callback procedure will be called by the Keychain Manager whenever a keychain event occurs, or when a lengthy operation is occurring, as indicated by the event mask. The event type, an information structure about the event, and a reference to the application-defined context are passed as parameters to this function.

RESULT CODE

noErr	0	No error.
paramErr	-50	Callback procedure is NULL or a non-addressable location.
errKCDuplicateCallback	-25297	Callback procedure is already registered.

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

`callbackProc` A universal procedure pointer to a callback routine, previously added with `KCAddCallback`.

DESCRIPTION
Unregisters the specified callback procedure, which will no longer be called by the Keychain Manager.

RESULT CODE

`noErr`	0	No error.
`errKCInvalidCallback`	-25298	Callback procedure has not been previously registered.

Managing the Human Interface

NEW FOR V2.0

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

`state` Flag that toggles the UI state.

DESCRIPTION
Use this routine to notify that the Keychain Manager can or cannot bring up UI. Use this routine to turn on and off UI around calls that might bring up a user interface. Failing to re-enable user interaction will affect other clients of the Keychain Manager. The interaction allowed state is not persistent and is reset when the machine reboots.

RESULT CODE

`noErr`	0	No error.

NEW FOR V2.0

```
Boolean KCIsInteractionAllowed (void)
```

DESCRIPTION
This routine returns the state of the Keychain Manager UI. If true, the Keychain Manager is free to show a user interface when needed.

High-level interfaces

Storing and retrieving AppleShare passwords

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

`serverSignature`	A pointer to a 16-byte AFP server signature block [optional].
`serverAddress`	Pointer to a Pascal string containing the server address, which may be specified as an AppleTalk zone name, a DNS domain name, or an IP address. [optional]
`serverName`	Pointer to a Pascal string containing the server name.
`volumeName`	Pointer to a Pascal string containing the volume name.
`accountName`	Pointer to a Pascal string containing the account name.
`passwordLength`	Length of the password data to be stored.
`passwordData`	Pointer to a buffer containing the password data.
`item`	On exit, returns a reference to the added item (pass NULL if you don't need it.)

DESCRIPTION

Adds a new AppleShare server password to the default keychain. The password should be uniquely identified by the combination of `serverName`, `volumeName`, `accountName`, and a location specified either by `serverAddress` or `serverSignature`. A reference to the newly added keychain item is optionally returned in `item`.

> NOTE: Most applications do not need to store AppleShare password data, as this is handled transparently by the AppleShare client software. To be compatible with the AppleShare client, you should store a fully-specified AFPXVolMountInfo record as the password data. Refer to the File Manager documentation for the definition of AFPXVolMountInfo.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough parameters were supplied.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCDuplicateItem`	-25299	Tried to add a password that already exists in the keychain.
`errKCDataTooLarge`	-25302	Tried to add more data than is allowed for a record of this type.

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

`serverSignature`	Pointer to a 16-byte AFP server signature block [or NULL to match any].
`serverAddress`	Pointer to a Pascal string containing the server address, which may be specified as an AppleTalk zone name, a DNS domain name, or an IP address. If this parameter is NULL, match any server address.
`serverName`	Pointer to a Pascal string containing the server name [or NULL to match any].
`volumeName`	Pointer to a Pascal string containing the volume name [or NULL to match any].
`accountName`	Pointer to a Pascal string containing the account name [or NULL to match any].
`maxLength`	The length of the caller-supplied `passwordData` buffer.
`passwordData`	Pointer to a buffer which will hold the returned password data.
`actualLength`	On exit, returns the actual length of the password data that was retrieved.
`item`	On exit, returns a reference to the found item (pass NULL if you don't need it.)

DESCRIPTION

Finds the first AppleShare password in the default keychain which matches the supplied parameters. The password data is returned in `passwordData`. Optionally, an item reference is returned in `item`. If `actualLength` and `passwordData` aren't specified (NULL), only the item is returned without the data.

NOTE: If the actual length of the password is greater than the size of the buffer (as supplied in `maxLength`), an error is returned and it is up to the caller to allocate a new buffer of `actualLength` bytes before a subsequent call to `KCFindAppleSharePassword`.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough valid parameters were supplied.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCItemNotFound`	-25300	No matching password item was found.
`errKCBufferTooSmall`	-25301	The password data was too large for the supplied buffer.

Storing and retrieving Internet passwords

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

`serverName`	Pointer to a Pascal string containing the server name.
`securityDomain`	Pointer to a Pascal string containing the security domain [optional].
`accountName`	Pointer to a Pascal string containing the account name.
`port`	A TCP/IP port number (pass `kAnyPort` to specify any port).
`protocol`	A constant that identifies the protocol associated with this password (pass `kAnyProtocol` to specify any protocol).
`authType`	A constant that identifies the authentication scheme used (pass `kAnyAuthType` to specify any authentication scheme).
`passwordLength`	Length of the password data to be stored.
`passwordData`	Pointer to a buffer containing the password data.
`item`	On exit, returns a reference to the added item (Pass NULL if you don't need it.)

DESCRIPTION

Adds a new Internet server password to the default keychain. Required parameters to identify the password include `serverName`, `accountName`, `port`, `protocol` and `authType`. In addition, some protocols may require an optional `securityDomain` when authentication is requested. A reference to the newly added keychain item is optionally returned in `item`.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough parameters were supplied.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCDuplicateItem`	-25299	Tried to add a password that already exists in the keychain.
`errKCDataTooLarge`	-25302	Tried to add more data than is allowed for a record of this type.

NEW FOR V2.0

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

`serverName`	Pointer to a Pascal string containing the server name.
`securityDomain`	Pointer to a Pascal string containing the security domain [optional].
`accountName`	Pointer to a Pascal string containing the account name.
`path`	Pointer to a Pascal string containing additional information that specifies a file or directory on the server specified by `serverName` [optional].
`port`	A TCP/IP port number (pass `kAnyPort` to specify any port).
`protocol`	A constant that identifies the protocol associated with this password (pass `kAnyProtocol` to specify any protocol).
`authType`	A constant that identifies the authentication scheme used (pass `kAnyAuthType` to specify any authentication scheme).
`passwordLength`	Length of the password data to be stored.
`passwordData`	Pointer to a buffer containing the password data.
`item`	On exit, returns a reference to the added item (Pass NULL if you don't need it.)

DESCRIPTION

Adds a new Internet server password to the default keychain. You should pass as much information as possible about the password you are adding. While all attributes are optional, you must specify at least one to help distinguish the password. In addition, some protocols may require an optional `securityDomain` when authentication is requested. A reference to the newly added keychain item is optionally returned in `item`.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough parameters were supplied.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCDuplicateItem`	-25299	Tried to add a password that already exists in the keychain.
`errKCDataTooLarge`	-25302	Tried to add more data than is allowed for a record of this type.

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

`serverName`	Pointer to a Pascal string which has the server name [or NULL to match any].
`securityDomain`	Pointer to a Pascal string which has the security domain [or NULL to match any].
`accountName`	Pointer to a Pascal string containing the account name [or NULL to match any].
`port`	A TCP/IP port number [or `kAnyPort` to match any port number].
`protocol`	A constant that identifies the protocol associated with this password.
`authType`	A constant that identifies the authentication scheme used.
`maxLength`	The length of the caller-supplied `passwordData` buffer.
`passwordData`	Pointer to a buffer which will hold the returned password data.
`actualLength`	On exit, returns the actual length of the password data that was retrieved.
`item`	On exit, returns a reference to the found item. (Pass NULL if you don't need it.)

DESCRIPTION

Finds the first Internet password in the default keychain which matches the supplied parameters. The password is returned in `passwordData`. Optionally, an item reference is returned in `item`. If `actualLength` and `passwordData` aren't specified (NULL), only the item is returned without the data.

> NOTE: If the actual length of the password is greater than the size of the buffer (as supplied in `maxLength`), an error is returned and it is up to the caller to allocate a new buffer of `actualLength` bytes before a subsequent call to `KCFindInternetPassword`.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough valid parameters were supplied.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCItemNotFound`	-25300	No matching password object was found.
`errKCBufferTooSmall`	-25301	The password data was too large for the supplied buffer.

NEW FOR V2.0

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

`serverName`	Pointer to a Pascal string which has the server name [or NULL to match any].
`securityDomain`	Pointer to a Pascal string which has the security domain [or NULL to match any].
`accountName`	Pointer to a Pascal string which has the account name [or NULL to match any].
`path`	Pointer to a Pascal string containing additional information that specifies a file or directory on the server specified by `serverName`.
`port`	A TCP/IP port number, or `kAnyPort` to match any port number.
`protocol`	A constant that identifies the protocol associated with this password.
`authType`	A constant that identifies the authentication scheme used.
`maxLength`	The length of the caller-supplied `passwordData` buffer.
`passwordData`	Pointer to a buffer which will hold the returned password data.
`actualLength`	On exit, returns the actual length of the password data that was retrieved.
`item`	On exit, returns a reference to the found item. (Pass NULL if you don't need it.)

DESCRIPTION

Finds the first Internet password in the default keychain which matches the supplied parameters. The password is returned in `passwordData`. Optionally, an item reference is returned in `item`. If `actualLength` and `passwordData` aren't specified (NULL), only the item is returned without the data.

> NOTE: If the actual length of the password is greater than the size of the buffer (as supplied in `maxLength`), an error is returned and it is up to the caller to allocate a new buffer of `actualLength` bytes before a subsequent call to `KCFindInternetPasswordWithPath`.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough valid parameters were supplied.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCItemNotFound`	-25300	No matching password object was found.
`errKCBufferTooSmall`	-25301	The password data was too large for the supplied buffer.

Storing and retrieving other types of passwords

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

`serverName`	Pointer to a Pascal string containing an application-defined service name.
`accountName`	Pointer to a Pascal string containing the account name.
`passwordLength`	Length of the password data to be stored.
`passwordData`	Pointer to a buffer containing the password data.
`item`	On exit, returns a reference to the added item. (Pass NULL if you don't need it.)

DESCRIPTION

Adds a new generic password to the default keychain. Required parameters to identify the password are `serviceName` and `accountName`, which are application-defined strings. A reference to the newly added keychain item is optionally returned in `item`.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough parameters were supplied.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCDuplicateItem`	-25299	Tried to add a password that already exists in the keychain.
`errKCDataTooLarge`	-25302	Tried to add more data than is allowed for a record of this type.

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

`serviceName`	Pointer to a Pascal string containing an application-defined service name [or NULL to match any].
`accountName`	Pointer to a Pascal string containing the account name [or NULL to match any].
`maxLength`	The length of the caller-supplied `passwordData` buffer.
`passwordData`	Pointer to a buffer which will hold the returned password data.
`actualLength`	On exit, returns the actual length of the password data that was retrieved.
`item`	On exit, returns a reference to the found item (pass NULL if you don't need it.)

DESCRIPTION

Finds the first generic password in the default keychain which matches the supplied parameters. The password is returned in `passwordData`. Optionally, an item reference is returned in `item`. If `actualLength` and `passwordData` aren't specified (NULL), only the item is returned without the data.

> NOTE: If the actual length of the password is greater than the size of the buffer (as supplied in `maxLength`), an error is returned and it is up to the caller to allocate a new buffer of `actualLength` bytes before a subsequent call to `KCFindGenericPassword`.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough valid parameters were supplied.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCItemNotFound`	-25300	No matching password object was found.
`errKCBufferTooSmall`	-25301	The password data was too large for the supplied buffer.

Creating and editing a keychain item

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

`itemClass`	A constant identifying the class of item to be created.
`itemCreator`	The creator code of the application which owns this item.
`length`	Length of the data to be stored in this item.
`data`	Pointer to a buffer containing the data to be stored in this item.
`item`	A reference to the new keychain item created by `KCNewItem`.

DESCRIPTION

Creates a new keychain item from the supplied parameters. A reference to the newly-created item is returned in `item`. A copy of the data buffer pointed to by `data` is stored in the item.

To permanently store this item, a subsequent call must be made to `KCAddItem`. When the item reference is no longer required, call `KCReleaseItem` to deallocate memory occupied by the item.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough valid parameters were supplied.
`memFullErr`	-108	Not enough memory in current heap zone to create the object.

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

`item`	A reference to a keychain item.
`attr`	Pointer to a `KCAttribute` record which specifies the attribute and its value.

DESCRIPTION

Use this function to modify attribute data for a keychain item. Standard item attributes which can be modified include:

`kDescriptionKCItemAttr`	A user-visible string describing this item. (StringPtr)
`kCommentKCItemAttr`	A user-editable string containing comments for this item. (StringPtr)
`kLabelKCItemAttr`	A user-editable string containing the label for this item. (StringPtr)
`kCreatorKCItemAttr`	The item's creator. (OSType)
`kTypeKCItemAttr`	The item's type. (OSType)
`kCustomIconKCItemAttr`	Flag indicating this item has an icon. (Boolean)

Each class of item has additional attributes specific to that class which may be modified. Refer to the definition of `KCItemAttr` for a complete list.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough valid parameters were supplied.
`memFullErr`	-108	Not enough memory in current heap zone to set attribute data.
`errKCInvalidItemRef`	-25304	The specified keychain item reference was invalid.
`errKCNoSuchAttr`	-25303	Tried to set an attribute which is undefined for this item class.
`errKCDataTooLarge`	-25302	More data was supplied than is allowed for this attribute type.

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

`item`	A reference to a keychain item.
`attr`	Pointer to a `KCAttribute` record with the following fields set on entry:
`tag`	A `KCAttrType` constant which specifies the attribute to get.
`length`	The maximum length of the data buffer pointed to by `data`.
`data`	Pointer to a buffer of sufficient length for the type of data to be returned.
`actualLength`	The actual length of the attribute data.

DESCRIPTION

Use this function to retrieve attribute data from a keychain item. Standard item attributes which can be retrieved include:

`kClassKCItemAttr`	The class of this item. (OSType)
`kCreationDateKCItemAttr`	The creation date of this item. (UInt32)
`kModDateKCItemAttr`	The last modified date of this item. (UInt32)
`kDescriptionKCItemAttr`	A user-visible string describing this item. (StringPtr)
`kLabelKCItemAttr`	A user-visible string for this item's label. (StringPtr)
`kCommentKCItemAttr`	A user-editable string containing comments for this item. (StringPtr)
`kCreatorKCItemAttr`	The item's creator. (OSType)
`kTypeKCItemAttr`	The item's type. (OSType)
`kCustomIconKCItemAttr`	Flag indicating this item has an icon. (Boolean)
`kScriptCodeKCItemAttr`	The script code for all strings in this item. (UInt32)

Each class of object has additional attribute values specific to that class which may be retrieved. Refer to the definition of `KCItemAttr` for a complete list.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough valid parameters were supplied.
`errKCInvalidItemRef`	-25304	The specified keychain item reference was invalid.
`errKCNoSuchAttr`	-25303	Tried to set an attribute which is undefined for this item class.
`errKCBufferTooSmall`	-25301	The attribute data was too large for the supplied buffer.

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

`item`	A reference to a keychain item.
`length`	The length of the caller-supplied `data` buffer.
`data`	Pointer to a buffer containing the data to be stored in this item.

DESCRIPTION

Use this function to set or change the data stored in the given keychain item.

RESULT CODE

`noErr`	0	No error.
`paramErr`	-50	Not enough valid parameters were supplied.
`errKCInvalidItemRef`	-25304	The specified keychain item reference was invalid.
`errKCDataTooLarge`	-25302	The data was too large for the supplied buffer.
`errKCDataNotModifiable`	-25317	The data cannot be set for this item.

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

item	A reference to a keychain item.
maxLength	The length of the caller-supplied data buffer.
data	Pointer to a buffer which will hold the returned data (required).
actualLength	On exit, returns the actual length of the data being retrieved (required).

DESCRIPTION

Use this function to retrieve the data stored in the given keychain item. You will not be able to call KCGetData for a private key. A paramErr is returned if you pass in NULL for data.

> NOTE: If the actual length of the data is greater than the size of the buffer (as supplied in maxLength), an error is returned and it is up to the caller to allocate a new buffer of actualLength bytes before a subsequent call to KCGetData.

RESULT CODE

noErr	0	No error.
paramErr	-50	Not enough valid parameters were supplied.
errKCInvalidItemRef	-25304	The specified keychain item reference was invalid.
errKCBufferTooSmall	-25301	The data was too large for the supplied buffer.
errKCDataNotAvailable	-25316	The data is not available for this item.

Managing keychain items

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

item	A reference to a keychain item.

DESCRIPTION

Use this routine to add a keychain item to the default keychain's permanent data store. If item references an existing item in the keychain, the existing item is updated. However, if item has not yet been added to the keychain and an identical item already exists in the keychain, an error is returned. If you want to add it to a specific keychain, bracket this call with KCGetDefaultKeychain and KCSetDefaultKeychain

RESULT CODE

noErr	0	No error.
errKCNoDefaultKeychain	-25307	No default keychain could be found.
errKCInvalidItemRef	-25306	The specified keychain item reference was invalid.
errKCDuplicateItem	-25299	Tried to add a new item that already exists in the keychain.

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

item	A reference to a keychain item.

DESCRIPTION

Use this routine to delete a keychain item from the default keychain's permanent data store. If itemRef has not previously been added to the keychain, KCDeleteItem does nothing and returns noErr.

IMPORTANT: KCDeleteItem does not dispose the memory occupied by the item reference itself; use KCReleaseItem when you are completely finished with an item.

RESULT CODE

noErr	0	No error.
errKCNoDefaultKeychain	-25307	No default keychain could be found.
errKCInvalidItemRef	-25304	The specified keychain item reference was invalid.

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

item A reference to a keychain item.

DESCRIPTION

Use this routine to update an existing keychain item after changing its attributes or data. The item is written to the keychain's permanent data store. If item has not previously been added to a keychain, KCUpdateItem does nothing and returns noErr.

RESULT CODE

noErr	0	No error.
errKCNoDefaultKeychain	-25307	No default keychain could be found.
errKCInvalidItemRef	-25304	The specified keychain item reference was invalid.

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

item A pointer to a keychain item reference.

DESCRIPTION

Use this routine to dispose the memory occupied by a keychain item reference.
The reference pointed to by itemRef is set to NULL on exit from this routine, and should not be used again.

RESULT CODE

noErr	0	No error.
errKCInvalidItemRef	-25304	The specified keychain item reference was invalid.

NEW FOR V2.0

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

item A keychain item reference.
keychain The keychain where item is stored.

DESCRIPTION

Use this routine to find out the location of keychain items. The search is performed on unlocked keychains only. If you have a reference to a keychain item whose keychain is locked, errKCInvalidItemRef will be returned.

RESULT CODE

noErr	0	No error.
errKCInvalidItemRef	-25304	The specified keychain item reference was invalid.

Note that the keychain reference returned by KCGetKeychain should be released by calling KCReleaseKeychain.

NEW FOR V2.0

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

`item`	A keychain item reference to copy
`destKeychain`	The keychain where `item` will be copied to
`copy`	The new copied item that is copied to `destKeychain`

DESCRIPTION

Use this routine to copy a keychain item from one keychain to another. The copy will be returned in `copy`.

RESULT CODE

`noErr`	0	No error.
`errKCInvalidKeychain`	`-25295`	The specified `destKeychain` was invalid.
`errKCReadOnly`	`-25292`	The `destKeychain` is read only.
`errKCNoSuchClass`	`-25306`	`item` has an invalid keychain item class.

Searching and enumerating keychain items

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

`keychain`	The keychain to search (NULL means search all unlocked keychains)
`attrList`	A list of zero or more `KCAttribute` records to be matched (NULL matches any keychain item).
`search`	A reference to the current search is returned here.
`item`	A reference to the first matching keychain item is returned here.

DESCRIPTION

Finds the first keychain item matching a list of zero or more specified attributes in the specified `keychain` and returns a reference to the item. Pass NULL for `keychain` if you wish to search all unlocked keychains. The caller is responsible for calling `KCReleaseItem` to release this reference when finished with it. A reference to the current search criteria is also returned, for subsequent calls to `KCFindNextItem`. This reference must be released by the caller when completely finished with a search (by calling `KCReleaseSearch`).

RESULT CODE

`noErr`	0	No error.
`errKCNoDefaultKeychain`	-25307	No default keychain could be found.
`errKCItemNotFound`	-25300	No matching keychain item was found.
`errKCNoSuchAttr`	-25303	Specified an attribute which is undefined for this item class.

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

`search`	A reference to the current search criteria.
`item`	A reference to the next matching keychain item, if any, is returned here.

DESCRIPTION

Finds the next keychain item matching the given search criteria, as previously specified by a call to `KCFindFirstItem`, and returns a reference to the item. The caller is responsible for releasing this reference when finished with it.

RESULT CODE

noErr	0	No error.
errKCNoDefaultKeychain	-25307	No default keychain could be found.
errKCInvalidSearchRef	-25305	The specified search reference was invalid.
errKCItemNotFound	-25300	No more matching keychain items were found.

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

`search` A pointer to the current search reference.

DESCRIPTION
Use this routine to release the memory occupied by a search reference.
The reference pointed to by `search` is set to NULL on exit, and should not be used again.

RESULT CODE

noErr	0	No error.
errKCInvalidSearchRef	-25305	The specified search reference was invalid.

Working With Certificates

NEW FOR V2.0

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

keychain	The keychain in which to search
name	A "C" string pointer to the certificate owner's common name
emailAddress	A "C" string pointer to the certificate ownder's e-mail address
options	The search criteria used when retrieving certificates
certificateItems	On exit, returns a array reference of certificate items that are found (pass NULL if you don't need it.)

DESCRIPTION
Use this routine to find certificates in a given keychain with the provided search critiera.

RESULT CODE

noErr	0	No error.
paramErr	-50	One of the specified arguments is invalid
errKCNoDefaultKeychain	-25307	No default keychain could be found.
errKCItemNotFound	-25300	No matching certificate object was found.
errKCBufferTooSmall	-25301	The certificate data was too large for the supplied buffer.

NEW FOR V2.0

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

`items`	A Core Foundation array of certificate keychain item references
`certificate`	The returned certificate keychain item
policyOIDs	An array of policy OIDs that determine the trust policy. To get a pointer to an array of policy OIDs for Macintosh file signing, call SecMacGetDefaultPolicyOIDs.
stopOn	Use kPolicyKCStopOn for the default value. This will leave stop decisions up to the Trust Policy module.

DESCRIPTION

This routine presents UI of a list of certificates the user can chose from. If the array contains at least 2 keychain items, the user can choose from that list. Otherwise, it returns the single keychain item in `certificate` with not UI.

RESULT CODE

`noErr`	0	No error.
`userCanceledErr`	`-128`	The user canceled out from the UI presented.
`paramErr`		

Private Keychain Manager API (not exported):

```
OSStatus KCDispatch (UInt16 commandID, void* value);
```

For a description of of these calls, consult the KeychainPriv.h file for the list of commandIDs.

Keychain Manager calls that use "C" Strings

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

OSStatus kcgetkeychainname (KCRef keychain, char* keychainName);

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


OSStatus kcfindapplesharepassword (AFPServerSignature*      serverSignature,
                                  const char*       serverAddress,
                                  const char*       serverName,
                                  const char*       volumeName,
                                  const char*       accountName,
                                  UInt32            maxLength,
                                  void*             passwordData,
```

```
                                    UInt32*           actualLength,
                                    KCItemRef*        item);

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

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

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

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

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

OSStatus kcfindgenericpassword (const char*     serviceName,
                                const char* accountName,
                                UInt32      maxLength,
```

```
                                    void*       passwordData,
                                    UInt32*     actualLength,
                                    KCItemRef*  item);
```

Data structures and types

```
typedef struct OpaqueKCRef*                 KCRef;
typedef struct OpaqueKCItemRef*             KCItemRef;
typedef struct OpaqueKCSearchRef*           KCSearchRef;
typedef CFDataRef                           SecOIDRef;

enum
{
      kIdleKCEvent            = 0,  /* null event */
      kLockKCEvent            = 1,  /* a keychain was locked */
      kUnlockKCEvent          = 2,  /* a keychain was unlocked */
      kAddKCEvent             = 3,  /* an item was added to a keychain */
      kDeleteKCEvent          = 4,  /* an item was deleted from a keychain */
      kUpdateKCEvent          = 5,  /* an item was updated */
      kChangeIdentityKCEvent  = 6,  /* the keychain identity was changed */
      kFindKCEvent            = 7,  /* an item was found */
      kSystemKCEvent          = 8   /* keychain client can process events */
      kDefaultChangedKCEvent  = 9,  /* the default keychain was changed */
      kDataAccessKCEvent      = 10  /* some process called KCGetData() */
};
typedef UInt16    KCEvent;

enum
{
      kIdleKCEventMask              = 1 << kIdleKCEvent,
      kLockKCEventMask              = 1 << kLockKCEvent,
      kUnlockKCEventMask            = 1 << kUnlockKCEvent,
      kAddKCEventMask               = 1 << kAddKCEvent,
      kDeleteKCEventMask            = 1 << kDeleteKCEvent,
      kUpdateKCEventMask            = 1 << kUpdateKCEvent,
      kChangeIdentityKCEventMask    = 1 << kChangeIdentityKCEvent,
      kFindKCEventMask              = 1 << kFindKCEvent,
      kSystemEventKCEventMask       = 1 << kSystemKCEvent,
      kDefaultChangedKCEventMask    = 1 << kDefaultChangedKCEvent,
      kDataAccessKCEventMask        = 1 << kDataAccessKCEvent,
      kEveryKCEventMask             = 0xFFFF    /* all of the above*/
};
typedef UInt16    KCEventMask;

typedef UInt8     AFPServerSignature[16];
typedef OSType    KCAttrType;
struct KCCallbackInfo
{
      UInt32                version;
      KCItemRef             item;
      ProcessSerialNumber   processID;
      EventRecord           event;
      KCRef                 keychain;
};
typedef struct KCCallbackInfo       KCCallbackInfo;
```

```
enum
{
      kUnlockStateKCStatus    = 1,
      kRdPermKCStatus         = 2,
      kWrPermKCStatus         = 4
};
typedef UInt32    KCStatus;

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

enum
{
      /* Common attributes */
      kClassKCItemAttr        = FOUR_CHAR_CODE('clas'),
      kCreationDateKCItemAttr = FOUR_CHAR_CODE('cdat'),
      kModDateKCItemAttr      = FOUR_CHAR_CODE('mdat'),
      kDescriptionKCItemAttr  = FOUR_CHAR_CODE('desc'),
      kCommentKCItemAttr      = FOUR_CHAR_CODE('icmt'),
      kCreatorKCItemAttr      = FOUR_CHAR_CODE('crtr'),
      kTypeKCItemAttr         = FOUR_CHAR_CODE('type'),
      kScriptCodeKCItemAttr   = FOUR_CHAR_CODE('scrp'),
      kLabelKCItemAttr        = FOUR_CHAR_CODE('labl'),
      kInvisibleKCItemAttr    = FOUR_CHAR_CODE('invi'),
      kNegativeKCItemAttr     = FOUR_CHAR_CODE('nega'),
      kCustomIconKCItemAttr   = FOUR_CHAR_CODE('cusi'),

      /* Unique Generic password attributes */
      kAccountKCItemAttr      = FOUR_CHAR_CODE('acct'),
      kServiceKCItemAttr      = FOUR_CHAR_CODE('svce'),

      /* Unique Internet password attributes */
      kSecurityDomainKCItemAttr     = FOUR_CHAR_CODE('sdmn'),
      kServerKCItemAttr             = FOUR_CHAR_CODE('srvr'),
      kAuthTypeKCItemAttr           = FOUR_CHAR_CODE('atyp'),
      kPortKCItemAttr               = FOUR_CHAR_CODE('port'),
      kPathKCItemAttr               = FOUR_CHAR_CODE('path'),

      /* Unique Appleshare password attributes */
      kVolumeKCItemAttr             = FOUR_CHAR_CODE('vlme'),
      kAddressKCItemAttr            = FOUR_CHAR_CODE('addr'),
      kSignatureKCItemAttr          = FOUR_CHAR_CODE('ssig'),
      /* Unique AppleShare and Internet attributes */
      kProtocolKCItemAttr           = FOUR_CHAR_CODE('ptcl'),

      /* Certificate attributes */
      kCommonNameKCItemAttr         = FOUR_CHAR_CODE('cn  '),
      kGivenNameKCItemAttr          = FOUR_CHAR_CODE('gnam'),
      kLastNameKCItemAttr           = FOUR_CHAR_CODE('sn  '),
      kIssuerKCItemAttr             = FOUR_CHAR_CODE('issu'),
```

```
      kSerialNumberKCItemAttr       = FOUR_CHAR_CODE('snbr'),
      kEMailKCItemAttr              = FOUR_CHAR_CODE('mail'),
      kIssuerURLKCItemAttr          = FOUR_CHAR_CODE('iurl'),

      /* Attributes shared by keys and certificates*/
      kEncryptKCItemAttr            = FOUR_CHAR_CODE('encr'),
      kDecryptKCItemAttr            = FOUR_CHAR_CODE('decr'),
      kSignKCItemAttr               = FOUR_CHAR_CODE('sign'),
      kVerifyKCItemAttr             = FOUR_CHAR_CODE('veri'),
      kWrapKCItemAttr               = FOUR_CHAR_CODE('wrap'),
      kUnwrapKCItemAttr             = FOUR_CHAR_CODE('unwr'),
      kStartDateKCItemAttr          = FOUR_CHAR_CODE('sdat'),
      kEndDateKCItemAttr            = FOUR_CHAR_CODE('edat')
};
typedef FourCharCode          KCItemAttr;

struct KCAttribute
{
      KCAttrType        tag;        /* 4-byte attribute tag */
      UInt32            length;     /* Length of attribute data */
      void*             data;       /* Pointer to attribute data */
};
typedef struct KCAttribute    KCAttribute;
typedef KCAttribute *         KCAttributePtr;

struct KCAttributeList
{
      UInt32       count;     /* How many attributes in the array */
      KCAttribute* attr;      /* Pointer to first attribute in array */
};
typedef struct KCAttributeList      KCAttributeList;

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

enum
{
      kKCProtocolTypeFTP            = FOUR_CHAR_CODE('ftp '),
      kKCProtocolTypeFTPAccount     = FOUR_CHAR_CODE('ftpa'),
      kKCProtocolTypeHTTP           = FOUR_CHAR_CODE('http'),
      kKCProtocolTypeIRC            = FOUR_CHAR_CODE('irc '),
      kKCProtocolTypeNNTP           = FOUR_CHAR_CODE('nntp'),
      kKCProtocolTypePOP3           = FOUR_CHAR_CODE('pop3'),
      kKCProtocolTypeSMTP           = FOUR_CHAR_CODE('smtp'),
      kKCProtocolTypeSOCKS          = FOUR_CHAR_CODE('sox '),
      kKCProtocolTypeIMAP           = FOUR_CHAR_CODE('imap'),
      kKCProtocolTypeLDAP           = FOUR_CHAR_CODE('ldap'),
      kKCProtocolTypeAppleTalk      = FOUR_CHAR_CODE('atlk'),
```

```
      kKCProtocolTypeAFP            = FOUR_CHAR_CODE('afp '),
      kKCProtocolTypeTelnet         = FOUR_CHAR_CODE('teln')
};
typedef FourCharCode          KCProtocolType;

enum {
      kSecOptionReserved            = 0x000000FF,
      kCertUsageShift               = 8,
      kCertUsageSigningAdd          = 1 << (kCertUsageShift + 0),
      kCertUsageSigningAskAndAdd    = 1 << (kCertUsageShift + 1),
      kCertUsageVerifyAdd           = 1 << (kCertUsageShift + 2),
      kCertUsageVerifyAskAndAdd     = 1 << (kCertUsageShift + 3),
      kCertUsageEncryptAdd          = 1 << (kCertUsageShift + 4),
      kCertUsageEncryptAskAndAdd    = 1 << (kCertUsageShift + 5),
      kCertUsageDecryptAdd          = 1 << (kCertUsageShift + 6),
      kCertUsageDecryptAskAndAdd    = 1 << (kCertUsageShift + 7),
      kCertUsageKeyExchAdd          = 1 << (kCertUsageShift + 8),
      kCertUsageKeyExchAskAndAdd    = 1 << (kCertUsageShift + 9),
      kCertUsageRootAdd             = 1 << (kCertUsageShift + 10),
      kCertUsageRootAskAndAdd       = 1 << (kCertUsageShift + 11),
      kCertUsageSSLAdd              = 1 << (kCertUsageShift + 12),
      kCertUsageSSLAskAndAdd        = 1 << (kCertUsageShift + 13),
      kCertUsageAllAdd              = 0x7FFFFF00
};
typedef UInt32    KCCertAddOptions;

/* Other constants */
const UInt16      kAnyPort          = 0;
const OSType      kAnyProtocol      = ((OSType) 0L);
const OSType      kAnyAuthType      = ((OSType) 0L);


/* Keychain Manager error codes */
enum {
      errKCNotAvailable             = -25291,
      errKCReadOnly                 = -25292,
      errKCAuthFailed               = -25293,
      errKCNoSuchKeychain           = -25294,
      errKCInvalidKeychain          = -25295,
      errKCDuplicateKeychain        = -25296,
      errKCDuplicateCallback        = -25297,
      errKCInvalidCallback          = -25298,
      errKCDuplicateItem            = -25299,
      errKCItemNotFound             = -25300,
      errKCBufferTooSmall           = -25301,
      errKCDataTooLarge             = -25302,
      errKCNoSuchAttr               = -25303,
      errKCInvalidItemRef           = -25304,
      errKCInvalidSearchRef         = -25305,
      errKCNoSuchClass              = -25306,
      errKCNoDefaultKeychain        = -25307,
      errKCInteractionNotAllowed    = -25308,
      errKCReadOnlyAttr             = -25309,
      errKCWrongKCVersion           = -25310,
      errKCKeySizeNotAllowed        = -25311,
      errKCNoStorageModule          = -25312,
```

```
      errKCNoCertificateModule      = -25313,
      errKCNoPolicyModule           = -25314,
      errKCInteractionRequired      = -25315,
      errKCDataNotAvailable         = -25316,
      errKCDataNotModifiable        = -25317
};
```

Summary of Keychain Manager routines

Opening and getting information about the Keychain Manager

```
OSStatus KCGetKeychainManagerVersion (UInt32* returnVers);

Boolean KeychainManagerAvailable (void);
```

Creating references to keychains

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

OSStatus KCMakeKCRefFromAlias (AliasHandle keychainAlias, KCRef* keychain);

OSStatus KCMakeAliasFromKCRef (KCRef  keychain, AliasHandle* keychainAlias);

OSStatus KCReleaseKeychain (KCRef* keychain);
```

Locking and unlocking a keychain

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

OSStatus KCLock (KCRef keychain);
```

Specifying the default keychain

```
OSStatus KCGetDefaultKeychain (KCRef* keychain);

OSStatus KCSetDefaultKeychain (KCRef keychain);
```

Creating a new keychain

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

Getting information about a keychain

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

OSStatus KCGetKeychainName (KCRef keychain, StringPtr keychainName);

OSStatus KCChangeSettings (KCRef keychain);
```

Enumerating available keychains

```
UInt16 KCCountKeychains (void);

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

Registering a callback procedure

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

OSStatus KCRemoveCallback (KCCallbackUPP  callbackProc);
```

Application-defined callback procedure

```
pascal void MyCallbackProc (  KCEvent keychainEvent,
                              KCCallbackInfo* info,
                              void* userContext);
```

Managing the Human Interface

```
OSStatus KCSetInteractionAllowed (Boolean state);

Boolean KCIsInteractionAllowed (void);
```

Storing and retrieving AppleShare passwords

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

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

Storing and retrieving Internet passwords

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

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

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

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

Storing and retrieving other types of passwords

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

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

Creating and editing a keychain item

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

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

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

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

Managing keychain items

```
OSStatus KCAddItem ( KCItemRef item );

OSStatus KCDeleteItem ( KCItemRef item );

OSStatus KCUpdateItem ( KCItemRef item );

OSStatus KCReleaseItem ( KCItemRef* item );

OSStatus KCGetKeychain ( KCItemRef  item, KCRef* keychain );

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

Searching and enumerating keychain items

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

OSStatus KCFindNextItem ( KCSearchRef search, KCItemRef* item );

OSStatus KCReleaseSearch ( KCSearchRef *search );
```

Working With Certificates

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

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

Routines that use "C" strings

```
OSStatus kcunlock (KCRef keychain, const char* password);
OSStatus kccreatekeychain (const char* password, KCRef* keychain);
OSStatus kcgetkeychainname (KCRef keychain, char* keychainName);
OSStatus kcaddapplesharepassword (AFPServerSignature* serverSignature,
                                  const char*       serverAddress,
                                  const char*       serverName,
                                  const char*       volumeName,
                                  const char*       accountName,
                                  UInt32            passwordLength,
                                  const void*       passwordData,
                                  KCItemRef*        item);


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

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

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

OSStatus kcfindinternetpassword (const char*    serverName,
                                  const char* securityDomain,
                                  const char* accountName,
                                  UInt16      port,
                                  OSType      protocol,
                                  OSType      authType,
```

```
                                    UInt32      maxLength,
                                    void*       passwordData,
                                    UInt32*     actualLength,
                                    KCItemRef*  item);

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

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

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

OBSOLETE IN V2.0

```
OSStatus KCGetStartupKeychain (KCRef* keychain);
OSStatus KCSetStartupKeychain (KCRef keychain);
OSStatus KCGetUserName (Str255 userName);
```