Certificate, Key, and Trust Services Reference

Security > Authentication

2008-11-19

Contents

Tables

Certificate, Key, and Trust Services Reference

Framework:	Security/Security.h
Declared in	SecBase.h SecCertificate.h SecIdentity.h SecIdentitySearch.h SecKey.h SecKeychainItem.h SecPolicy.h SecPolicySearch.h SecTrust.h SecTrustSettings.h cssmapple.h

Overview

Certificate, Key, and Trust Services provides a C API for managing certificates, public and private keys, and trust policies. You can use these services in your application to:

- Determine identity by matching a certificate with a private key
- Create and request certificate objects
- Import certificates, keys, and identities
- Create public-private key pairs
- Represent trust policies

Certificate, Key, and Trust Services can be used in applications running in Aspen.

Functions by Task

Getting Type Identifiers

`SecCertificateGetTypeID` (page 20)
: Returns the unique identifier of the opaque type to which a `SecCertificate` object belongs.

`SecIdentityGetTypeID` (page 25)
: Returns the unique identifier of the opaque type to which a `SecIdentity` object belongs.

SecIdentitySearchGetTypeID (page 27)
    Returns the unique identifier of the opaque type to which a SecIdentitySearch object belongs.

SecKeyGetTypeID (page 33)
    Returns the unique identifier of the opaque type to which a SecKey object belongs.

SecPolicyGetTypeID (page 35)
    Returns the unique identifier of the opaque type to which a SecPolicy object belongs.

SecPolicySearchGetTypeID (page 37)
    Returns the unique identifier of the opaque type to which a SecPolicySearch object belongs.

SecTrustGetTypeID (page 46)
    Returns the unique identifier of the opaque type to which a SecTrust object belongs.

Managing Certificates

SecCertificateAddToKeychain (page 13)
    Adds a certificate to a keychain.

SecCertificateCreateFromData (page 16)
    Creates a certificate object based on the specified data, type, and encoding.

SecCertificateCopyCommonName (page 14)
    Retrieves the common name of the subject of a certificate.

SecCertificateCopyEmailAddresses (page 14)
    Retrieves the email addresses for the subject of a certificate.

SecCertificateCopyPreference (page 15)
    Retrieves the preferred certificate for the specified name and key use.

SecCertificateCopyPublicKey (page 15)
    Retrieves the public key from a certificate.

SecCertificateGetAlgorithmID (page 17)
    Retrieves the algorithm identifier for a certificate.

SecCertificateGetCLHandle (page 17)
    Retrieves the certificate library handle from a certificate object.

SecCertificateGetData (page 18)
    Retrieves the data for a certificate.

SecCertificateGetIssuer (page 18)
    Unsupported.

SecCertificateGetItem (page 19)
    Unsupported.

SecCertificateGetSubject (page 19)
    Unsupported.

SecCertificateGetType (page 19)
    Retrieves the type of a specified certificate.

SecCertificateSetPreference (page 20)
    Sets the preferred certificate for a specified name, key use, and date.

Managing Identities

`SecIdentityCopyCertificate` (page 22)
Retrieves a certificate associated with an identity.

`SecIdentityCopyPreference` (page 22)
Returns the preferred identity for the specified name and key use.

`SecIdentityCopyPrivateKey` (page 23)
Retrieves the private key associated with an identity.

`SecIdentityCopySystemIdentity` (page 23)
Obtains the system-wide identity associated with a specified domain.

`SecIdentityCreateWithCertificate` (page 24)
Creates a new identity for a certificate and its associated private key.

`SecIdentitySearchCopyNext` (page 25)
Finds the next identity matching specified search criteria

`SecIdentitySearchCreate` (page 26)
Creates a search object for finding identities.

`SecIdentitySetPreference` (page 27)
Sets the preferred identity for the specified name and key use.

`SecIdentitySetSystemIdentity` (page 28)
Assigns the system-wide identity to be associated with a specified domain.

Cryptography and Digital Signatures

`SecKeyCreatePair` (page 28)
Creates an asymmetric key pair and stores it in a keychain.

`SecKeyGetCredentials` (page 31)
Returns an access credential for a key.

`SecKeyGetCSPHandle` (page 32)
Returns the CSSM CSP handle for a key.

`SecKeyGenerate` (page 30)
Creates a symmetric key and optionally stores it in a keychain.

`SecKeyGetCSSMKey` (page 33)
Retrieves a pointer to the `CSSM_KEY` structure containing the key stored in a keychain item.

Managing Policies

`SecPolicyGetOID` (page 34)
Retrieves a policy's object identifier.

`SecPolicyGetTPHandle` (page 34)
Retrieves the trust policy handle for a policy object.

`SecPolicyGetValue` (page 35)
Retrieves a policy's value.

`SecPolicySearchCopyNext` (page 36)
Retrieves a policy object for the next policy matching specified search criteria.

`SecPolicySearchCreate` (page 36)
Creates a search object for finding policies.

`SecPolicySetValue` (page 38)
Sets a policy's value.

Managing Trust

`SecTrustCopyAnchorCertificates` (page 38)
Retrieves the anchor (root) certificates stored by Mac OS X.

`SecTrustCopyCustomAnchorCertificates` (page 39)
Retrieves the custom anchor certificates, if any, used by a given trust.

`SecTrustCopyPolicies` (page 40)
Retrieves the policies used by a given trust management object.

`SecTrustCreateWithCertificates` (page 40)
Creates a trust management object based on certificates and policies.

`SecTrustEvaluate` (page 41)
Evaluates trust for the specified certificate and policies.

`SecTrustGetCssmResult` (page 43)
Retrieves the CSSM trust result.

`SecTrustGetCssmResultCode` (page 43)
Retrieves the CSSM result code from the most recent trust evaluation for a trust management object.

`SecTrustGetResult` (page 44)
Retrieves details on the outcome of a call to the function `SecTrustEvaluate`.

`SecTrustGetTPHandle` (page 45)
Retrieves the trust policy handle.

`SecTrustSetAnchorCertificates` (page 46)
Sets the anchor certificates used when evaluating a trust management object.

`SecTrustSetKeychains` (page 47)
Sets the keychains searched for intermediate certificates when evaluating a trust management object.

`SecTrustSetParameters` (page 48)
Sets the action and action data for a trust management object.

`SecTrustSetPolicies` (page 49)
Set the policies to use in an evaluation.

`SecTrustSetVerifyDate` (page 55)
Sets the date and time against which the certificates in a trust management object are verified.

`SecTrustGetCSSMAnchorCertificates` (page 73) Deprecated in Mac OS X v10.5
Retrieves the CSSM anchor certificates.

`SecTrustGetUserTrust` (page 73) Deprecated in Mac OS X v10.5
Retrieves the user-specified trust setting for a certificate and policy.

`SecTrustSetUserTrust` (page 74) Deprecated in Mac OS X v10.5
Sets the user-specified trust settings of a certificate and policy.

Managing Trust Settings

`SecTrustSettingsCopyCertificates` (page 49)
    Obtains an array of all certificates that have trust settings in a specific trust settings domain.

`SecTrustSettingsCopyModificationDate` (page 50)
    Obtains the date and time at which a certificate's trust settings were last modified.

`SecTrustSettingsCopyTrustSettings` (page 50)
    Obtains the trust settings for a certificate.

`SecTrustSettingsCreateExternalRepresentation` (page 53)
    Obtains an external, portable representation of the specified domain's trust settings.

`SecTrustSettingsImportExternalRepresentation` (page 53)
    Imports trust settings into a trust domain.

`SecTrustSettingsRemoveTrustSettings` (page 54)
    Deletes the trust settings for a certificate.

`SecTrustSettingsSetTrustSettings` (page 54)
    Specifies trust settings for a certificate.

Reporting Errors

`SecCopyErrorMessageString` (page 21)
    Returns a string describing an error.

Functions

SecCertificateAddToKeychain

Adds a certificate to a keychain.

```
OSStatus SecCertificateAddToKeychain (
   SecCertificateRef certificate,
   SecKeychainRef keychain
);
```

Parameters

`certificate`
    The certificate object for the certificate to add to the keychain.

`keychain`
    The keychain object for the keychain to which you want to add the certificate. Pass `NULL` to add the certificate to the default keychain.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
This function requires a certificate object, which can, for example, be created with the `SecCertificateCreateFromData` (page 16) function or obtained over a network (see *Secure Transport Reference*). If the certificate has already been added to the specified keychain, the function returns

`errSecDuplicateItem` and does not add another copy to the keychain. The function looks at the certificate data, not at the certificate object, to determine whether the certificate is a duplicate. It considers two certificates to be duplicates if they have the same primary key attributes.

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

Declared In
`SecCertificate.h`

SecCertificateCopyCommonName

Retrieves the common name of the subject of a certificate.

```
OSStatus SecCertificateCopyCommonName(
    SecCertificateRef certificate,
    CFStringRef *commonName
);
```

Parameters

`certificate`
The certificate object from which to retrieve the common name.

`commonName`
On return, points to the common name. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`SecCertificate.h`

SecCertificateCopyEmailAddresses

Retrieves the email addresses for the subject of a certificate.

```
OSStatus SecCertificateCopyEmailAddresses(
    SecCertificateRef certificate,
    CFArrayRef *emailAddresses
);
```

Parameters

`certificate`
The certificate object from which to retrieve the email addresses.

`emailAddresses`
On return, an array of zero or more `CFStringRef` elements, each containing one email address found in the certificate subject. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
Not every certificate subject includes an email address. If the function does not find any email addresses, it returns a `CFArrayRef` object with zero elements in the array.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`SecCertificate.h`

SecCertificateCopyPreference

Retrieves the preferred certificate for the specified name and key use.

```
OSStatus SecCertificateCopyPreference(
    CFStringRef name,
    CSSM_KEYUSE keyUsage,
    SecCertificateRef *certificate
);
```

Parameters

name
: A string containing an email address (RFC822) or other name for which a preferred certificate is requested.

keyUsage
: A key use value, as defined in `Security.framework/cssmtype.h`. Pass `0` to ignore this parameter.

certificate
: On return, a reference to the preferred certificate, or `NULL` if none was found. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
This function is typically used to obtain the preferred encryption certificate for an email recipient.

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecCertificateSetPreference` (page 20)

Declared In
`SecCertificate.h`

SecCertificateCopyPublicKey

Retrieves the public key from a certificate.

```
OSStatus SecCertificateCopyPublicKey(
    SecCertificateRef certificate,
    SecKeyRef *key
);
```

Parameters

certificate

The certificate object from which to retrieve the public key.

key

On return, points to the public key for the specified certificate. Call the `CFRelease` function to release this object when you are finished with it.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Availability

Available in Mac OS X v10.5 and later.

Declared In

`SecCertificate.h`

SecCertificateCreateFromData

Creates a certificate object based on the specified data, type, and encoding.

```
OSStatus SecCertificateCreateFromData (
   const CSSM_DATA *data,
   CSSM_CERT_TYPE type,
   CSSM_CERT_ENCODING encoding,
   SecCertificateRef *certificate
);
```

Parameters

data

A pointer to the certificate data. The data must be an X509 certificate in binary format.

type

The certificate type as defined in `Security.framework/cssmtype.h`. Permissible values are `CSSM_CERT_X_509v1`, `CSSM_CERT_X_509v2`, and `CSSM_CERT_X_509v3`. If you are unsure of the certificate type, use `CSSM_CERT_X_509v3`.

encoding

The certificate encoding as defined in `Security.framework/cssmtype.h`. Permissible values are `CSSM_CERT_ENCODING_BER` and `CSSM_CERT_ENCODING_DER`. If you are unsure of the encoding, use `CSSM_CERT_ENCODING_BER`.

certificate

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

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

The certificate object returned by this function is used as input to several other functions in the API.

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

Declared In
`SecCertificate.h`

SecCertificateGetAlgorithmID

Retrieves the algorithm identifier for a certificate.

```
OSStatus SecCertificateGetAlgorithmID(
    SecCertificateRef certificate,
    const CSSM_X509_ALGORITHM_IDENTIFIER **algid
);
```

Parameters

`certificate`
The certificate object from which to retrieve the algorithm identifier.

`algid`
On return, points to a struct that identifies the algorithm for this certificate. This pointer remains valid until the certificate reference is released. Do not attempt to free this pointer.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
The `CSSM_X509_ALGORITHM_IDENTIFIER` struct is defined in `Security.framework/x509defs.h` and discussed in *Common Security: CDSA and CSSM, version 2 (with corrigenda)* from The Open Group (http://www.opengroup.org/security/cdsa.htm). Possible algorithms are enumerated in `Security.framework/oidsalg.h`.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`SecCertificate.h`

SecCertificateGetCLHandle

Retrieves the certificate library handle from a certificate object.

```
OSStatus SecCertificateGetCLHandle (
   SecCertificateRef certificate,
   CSSM_CL_HANDLE *clHandle
);
```

Parameters

`certificate`
The certificate object from which to obtain the certificate library handle.

`clHandle`
On return, points to the certificate library handle of the specified certificate. This handle remains valid until the certificate object is released.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
The certificate library handle is the CSSM identifier of the certificate library module that is managing the certificate. The certificate library handle is used as an input to a number of CSSM functions.

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

Declared In
`SecCertificate.h`

SecCertificateGetData

Retrieves the data for a certificate.

```
OSStatus SecCertificateGetData (
   SecCertificateRef certificate,
   CSSM_DATA_PTR data
);
```

Parameters

`certificate`
A certificate object for the certificate from which to retrieve the data.

`data`
On return, points to the data for the certificate specified. You must allocate the space for a `CSSM_DATA` structure before calling this function. This data pointer is only guaranteed to remain valid as long as the certificate remains unchanged and valid.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
This function requires a certificate object, which can, for example, be created with the `SecCertificateCreateFromData` (page 16) function, obtained from an identity with the `SecIdentityCopyCertificate` (page 22) function, or obtained over a network (see *Secure Transport Reference*).

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

Related Sample Code
SSLSample

Declared In
`SecCertificate.h`

SecCertificateGetIssuer

Unsupported.

```
OSStatus SecCertificateGetIssuer (
   SecCertificateRef certificate,
   CSSM_X509_NAME *issuer
);
```

Availability
Unsupported.

Declared In
`SecCertificate.h`

SecCertificateGetItem

Unsupported.

```
OSStatus SecCertificateGetItem (
    SecCertificateRef certificate,
    SecKeychainItemRef *item
);
```

Availability
Unsupported.

Declared In
`SecCertificate.h`

SecCertificateGetSubject

Unsupported.

```
OSStatus SecCertificateGetSubject (
   SecCertificateRef certificate,
   CSSM_X509_NAME *subject
);
```

Availability
Unsupported.

Declared In
`SecCertificate.h`

SecCertificateGetType

Retrieves the type of a specified certificate.

```
OSStatus SecCertificateGetType (
   SecCertificateRef certificate,
   CSSM_CERT_TYPE *certificateType
);
```

Parameters

`certificate`
A certificate object for the certificate for which to obtain the type.

certificateType
: On return, points to the type of the specified certificate. Certificate types are defined in `Security.framework/cssmtype.h`. You must allocate the space for a `CSSM_CERT_TYPE` structure before calling this function.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

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

Declared In
`SecCertificate.h`

SecCertificateGetTypeID

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

```
CFTypeID SecCertificateGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecCertificateRef` (page 57) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecCertificateRef` (page 57) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

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

Declared In
`SecCertificate.h`

SecCertificateSetPreference

Sets the preferred certificate for a specified name, key use, and date.

```
OSStatus SecCertificateSetPreference(
    SecCertificateRef certificate,
    CFStringRef name,
    CSSM_KEYUSE keyUsage,
    CFDateRef date
);
```

Parameters

certificate
: The certificate object identifying the preferred certificate.

name
: A string containing an email address (RFC822) or other name with which the preferred certificate is to be associated.

keyUsage

A key use value, as defined in `Security.framework/cssmtype.h`. Pass `0` if you don't want to specify a particular key use.

date

The date after which this preference is no longer valid. If supplied, the preferred certificate is changed only if this date is later than the currently saved setting. Pass `NULL` if this preference should not be restricted by date.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

This function is typically used to set the preferred encryption certificate for an email recipient, either manually (when encrypting email to a recipient) or automatically upon receipt of encrypted email.

Availability

Available in Mac OS X v10.5 and later.

See Also

`SecCertificateCopyPreference` (page 15)

Declared In

`SecCertificate.h`

SecCopyErrorMessageString

Returns a string describing an error.

```
CFStringRef SecCopyErrorMessageString(
   OSStatus status,
   void *reserved
);
```

Parameters

status

An error result code of type `OSStatus` or `CSSM_RETURN`, as returned by a security or CSSM function.

reserved

Reserved for future use. Pass `NULL` in this parameter.

Return Value

A reference to an error string, or NULL if no error string is available for the specified result code. You must release this reference when you are finished with it by calling the `CFRelease` function.

Discussion

The error strings returned by this function are taken from the `SecBase.h` header file and are therefore not localizable.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`SecBase.h`

SecIdentityCopyCertificate

Retrieves a certificate associated with an identity.

```
OSStatus SecIdentityCopyCertificate (
   SecIdentityRef identityRef,
   SecCertificateRef *certificateRef
);
```

Parameters

identityRef

The identity object for the identity whose certificate you wish to retrieve.

certificateRef

On return, points to the certificate object associated with the specified identity.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

An identity is a digital certificate together with its associated private key.

For a certificate in a keychain, you can cast the `SecCertificateRef` data type to a `SecKeychainItemRef` for use with Keychain Services functions.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecIdentity.h`

SecIdentityCopyPreference

Returns the preferred identity for the specified name and key use.

```
OSStatus SecIdentityCopyPreference(
    CFStringRef name,
    CSSM_KEYUSE keyUsage,
    CFArrayRef validIssuers,
    SecIdentityRef *identity
);
```

Parameters

name

A string containing a URI, RFC822 email address, DNS hostname, or other name that uniquely identifies the service requiring an identity.

keyUsage

A key use value, as defined in `Security.framework/cssmtype.h`. Pass `0` if you don't want to specify a particular key use.

validIssuers

An array of `CFDataRef` instances whose contents are the subject names of allowable issuers, as returned by a call to `SSLCopyDistinguishedNames` (`Security.framework/SecureTransport.h`). Pass `NULL` if you don't want to limit the search to specific issuers.

`identity`

On return, a reference to the preferred identity, or `NULL` if none was found. Call the `CFRelease` function to release this object when you are finished with it.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

If a preferred identity has not been set for the specified name, the returned identity reference is `NULL`. You should then typically perform a search for possible identities, using `SecIdentitySearchCreate` (page 26) and `SecIdentitySearchCopyNext` (page 25) , allowing the user to choose from a list if more than one is found.

Availability

Available in Mac OS X v10.5 and later.

See Also

`SecIdentitySetPreference` (page 27)

Declared In

`SecIdentity.h`

SecIdentityCopyPrivateKey

Retrieves the private key associated with an identity.

```
OSStatus SecIdentityCopyPrivateKey (
   SecIdentityRef identityRef,
   SecKeyRef *privateKeyRef
);
```

Parameters

`identityRef`

The identity object for the identity whose private key you wish to retrieve.

`privateKeyRef`

On return, points to the private key object for the specified identity. The private key must be of class type `kSecAppleKeyItemClass`.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

An identity is a digital certificate together with its associated private key.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecIdentity.h`

SecIdentityCopySystemIdentity

Obtains the system-wide identity associated with a specified domain.

```
OSStatus SecIdentityCopySystemIdentity(
    CFStringRef domain,
    SecIdentityRef *idRef,
    CFStringRef *actualDomain
);
```

Parameters

domain

The domain for which you want to find an identity, typically in reverse DNS notation, such as `com.apple.security`. You may also pass the values defined in "System Identity Domains" (page 64).

idRef

On return, the identity object of the system-wide identity associated with the specified domain. Call the `CFRelease` function to release this object when you are finished with it.

actualDomain

On return, the actual domain name of the returned identity object is returned here. This may be different from the requested domain. Pass `NULL` if you do not want this information.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
If no system-wide identity exists for the specified domain, a domain-specific alternate may be returned instead, typically (but not exclusively) the system-wide default identity (`kSecIdentityDomainDefault`).

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecIdentitySetSystemIdentity` (page 28)

Declared In
`SecIdentity.h`

SecIdentityCreateWithCertificate

Creates a new identity for a certificate and its associated private key.

```
OSStatus SecIdentityCreateWithCertificate(
    CFTypeRef keychainOrArray,
    SecCertificateRef certificateRef,
    SecIdentityRef *identityRef
);
```

Parameters

keychainOrArray

A reference to a keychain or an array of keychains to search for the associated private key. Specify `NULL` to search the user's default keychain search list.

certificateRef

The certificate for which you want to create an identity.

identityRef

On return, an identity object for the certificate and its associated private key. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Availability
Available in Mac OS X v10.5 and later.

Declared In
`SecIdentity.h`

SecIdentityGetTypeID

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

```
CFTypeID SecIdentityGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecIdentityRef` (page 57) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecIdentityRef` (page 57) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

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

Related Sample Code
SSLSample

Declared In
`SecIdentity.h`

SecIdentitySearchCopyNext

Finds the next identity matching specified search criteria

```
OSStatus SecIdentitySearchCopyNext (
   SecIdentitySearchRef searchRef,
   SecIdentityRef *identity
);
```

Parameters

`searchRef`
An identity search object specifying the search criteria for this search. You create the identity search object by calling the `SecIdentitySearchCreate` (page 26) function.

`identity`
On return, points to the identity object of the next matching identity (if any). Call the `CFRelease` function to release this object when finished with it.

Return Value
A result code. When there are no more identities that match the parameters specified to `SecIdentitySearchCreate` (page 26), `errSecItemNotFound` is returned. See "Certificate, Key, and Trust Services Result Codes" (page 69).

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

Related Sample Code
SSLSample

Declared In
`SecIdentitySearch.h`

SecIdentitySearchCreate

Creates a search object for finding identities.

```
OSStatus SecIdentitySearchCreate (
   CFTypeRef keychainOrArray,
   CSSM_KEYUSE keyUsage,
   SecIdentitySearchRef *searchRef
);
```

Parameters

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

keyUsage
: ACSSM key use value as defined in `Security.framework/cssmtype.h`. (Note that, because key recovery is not implemented, the `SIGN_RECOVER` and `VERIFY_RECOVER` constants are not supported.) Use this parameter to filter the search by specifying the key use for the identity. Pass `0` if you want all identities returned by this search. Pass `CSSM_KEYUSE_ANY` to limit the identities returned to those that can be used for every operation.

searchRef
: On return, points to the identity search object. Call the `CFRelease` function to release this object when you are done with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
You can OR `CSSM_KEYUSE` values together to set more than one value for key use. Use the returned search object in calls to the `SecIdentitySearchCopyNext` (page 25) function to obtain identities that match the search criteria.

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

Related Sample Code
SSLSample

Declared In
`SecIdentitySearch.h`

SecIdentitySearchGetTypeID

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

```
CFTypeID SecIdentitySearchGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecIdentitySearchRef` (page 58) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecIdentitySearchRef` (page 58) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

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

Declared In
`SecIdentitySearch.h`

SecIdentitySetPreference

Sets the preferred identity for the specified name and key use.

```
OSStatus SecIdentitySetPreference(
    SecIdentityRef identity,
    CFStringRef name,
    CSSM_KEYUSE keyUsage
);
```

Parameters

`identity`
A reference to the preferred identity.

`name`
A string containing a URI, RFC822 email address, DNS host name, or other name that uniquely identifies a service requiring this identity.

`keyUsage`
A key use value, as defined in `Security.framework/cssmtype.h`. Pass `0` if you don't want to specify a particular key use.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecIdentityCopyPreference` (page 22)

Declared In
`SecIdentity.h`

SecIdentitySetSystemIdentity

Assigns the system-wide identity to be associated with a specified domain.

```
OSStatus SecIdentitySetSystemIdentity(
    CFStringRef domain,
    SecIdentityRef idRef
);
```

Parameters

`domain`

The domain to which the specified identity will be assigned, typically in reverse DNS notation, such as `com.apple.security`. You may also pass the values defined in "System Identity Domains" (page 64).

`idRef`

The identity to be assigned to the specified domain. Pass `NULL` to delete any currently-assigned identity for the specified domain; in this case, it is not an error if no identity exists for the specified domain.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

The caller must be running as root.

Availability

Available in Mac OS X v10.5 and later.

See Also

`SecIdentityCopySystemIdentity` (page 23)

Declared In

`SecIdentity.h`

SecKeyCreatePair

Creates an asymmetric key pair and stores it in a keychain.

```
OSStatus SecKeyCreatePair (
   SecKeychainRef keychainRef,
   CSSM_ALGORITHMS algorithm,
   uint32 keySizeInBits,
   CSSM_CC_HANDLE contextHandle,
   CSSM_KEYUSE publicKeyUsage,
   uint32 publicKeyAttr,
   CSSM_KEYUSE privateKeyUsage,
   uint32 privateKeyAttr,
   SecAccessRef initialAccess,
   SecKeyRef *publicKey,
   SecKeyRef *privateKey
);
```

Parameters

keychainRef
: The keychain object for the keychain in which to store the private and public key items. Specify `NULL` for the default keychain.

algorithm
: The algorithm to use to generate the key pair. Possible values are defined in `Security.framework/cssmtype.h`. Algorithms supported by the AppleCSP module are listed in *Security Release Notes*. This parameter is ignored if the `contextHandle` parameter is not `0`.

keySizeInBits
: A key size for the key pair. See *Security Release Notes* for permissible key sizes for each algorithm supported by the AppleCSP module.

contextHandle
: A CSSM CSP handle, or `0`. If this argument is not `0`, the `algorithm` and `keySizeInBits` parameters are ignored.

publicKeyUsage
: A bit mask indicating all permitted uses for the new public key. The possible values for the `CSSM_KEYUSE` data type are defined in `Security.framework/cssmtype.h`.

publicKeyAttr
: A bit mask defining attribute values for the new public key. The bit mask values are equivalent to those defined for `CSSM_KEYATTR_FLAGS` in `Security.framework/cssmtype.h`.

privateKeyUsage
: A bit mask indicating all permitted uses for the new private key. The possible values for the `CSSM_KEYUSE` data type are defined in `Security.framework/cssmtype.h`.

privateKeyAttr
: A bit mask defining attribute values for the new private key. The bit mask values are defined in `CSSM_KEYATTR_FLAGS` in `Security.framework/cssmtype.h`. Supported values are `CSSM_KETATTR_EXTRACTABLE` (the key can be taken out of the keychain) and `CSSM_KEYATTR_SENSITIVE` (an extractable key can be taken out of the keychain only in wrapped form—that is, encrypted). (Note that you must set *both* of these bits if you want the key to be extractable in wrapped form.) For any other value of this attribute, the key cannot be taken out of the keychain under any circumstances.

initialAccess
: An access object that sets the initial access control list for each of the keys returned. See "Creating an Access Object" in *Keychain Services Reference* for functions that create an access object. For default access, specify `NULL`. The default is free access to the tool or application that calls this function, with attempted access to sensitive information by any other application causing a confirmation dialog to be displayed.

publicKey

On return, points to the keychain item object of the new public key. Use this object as input to the `SecKeyGetCSSMKey` (page 33) function to obtain the `CSSM_KEY` structure containing the key. Call the `CFRelease` function to release this object when you are finished with it.

privateKey

On return, points to the keychain item object of the new private key. Use this object as input to the `SecKeyGetCSSMKey` (page 33) function to obtain the `CSSM_KEY` structure containing the key. Call the `CFRelease` function to release this object when you are finished with it.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

This function uses default values for any attributes required by specific key-generation algorithms. Algorithms supported by the AppleCSP module are listed in *Security Release Notes*. For details about algorithms and default values for key-generation parameters, download the CDSA security framework from the ADC website at http://developer.apple.com/darwin/projects/security/ and read the file `Supported_CSP_Algorithms.doc` in the Documentation folder.

If you need extra parameters to generate a key—as required by some algortihms—call `SecKeychainGetCSPHandle` to obtain a CSSM CSP handle and then call `CSSM_CSP_CreateKeyGenContext` to create a context. With this context, use `CSSM_UpdateContextAttributes` to add additional parameters. Finally, call `CSSM_DeleteContext` to dispose of the context after calling this function.

Availability

Available in Mac OS X v10.2 and later.

See Also

`SecKeyGenerate` (page 30)

Declared In

`SecKey.h`

SecKeyGenerate

Creates a symmetric key and optionally stores it in a keychain.

```
OSStatus SecKeyGenerate(
    SecKeychainRef keychainRef,
    CSSM_ALGORITHMS algorithm,
    uint32 keySizeInBits,
    CSSM_CC_HANDLE contextHandle,
    CSSM_KEYUSE keyUsage,
    uint32 keyAttr,
    SecAccessRef initialAccess,
    SecKeyRef* keyRef
);
```

Parameters

keychainRef

The keychain in which to store the generated key. Specify `NULL` to generate a transient key.

algorithm
: The algorithm to use in generating the symmetric key. Possible values are defined in `cssmtype.h`. Algorithms supported by the AppleCSP module are listed in *Security Release Notes*. This parameter is ignored if the `contextHandle` parameter is not `0`.

keySizeInBits
: A key size for the key pair. This parameter is ignored if the `contextHandle` parameter is not `0`.

contextHandle
: A CSSM CSP handle, or `0`. If this argument is not `0`, the `algorithm` and `keySizeInBits` parameters are ignored.

keyUsage
: A bit mask indicating all permitted uses for the new key. The possible values for the `CSSM_KEYUSE` data type are defined in `cssmtype.h`.

keyAttr
: A bit mask defining attribute values for the new key. The bit mask values are defined in `CSSM_KEYATTR_FLAGS` in `cssmtype.h`.

initialAccess
: An access object that sets the initial access control list for the key returned. See "Creating an Access Object" in *Keychain Services Reference* for functions that create an access object. This parameter is ignored if you specify `NULL` for the `keychainRef` parameter.

keyRef
: On return, points to the keychain item object of the new public key. Use this object as input to the `SecKeyGetCSSMKey` (page 33) function to obtain the `CSSM_KEY` structure containing the key. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
Key-generation algorithms supported by the AppleCSP module are listed in *Security Release Notes*. For details about algorithms and default values for key-generation parameters, download the CDSA security framework from the ADC website at http://developer.apple.com/darwin/projects/security/ and read the file `Supported_CSP_Algorithms.doc` in the Documentation folder.

If you need extra parameters to generate a key—as required by some algortihms—call `SecKeychainGetCSPHandle` to obtain a CSSM CSP handle and then call `CSSM_CSP_CreateKeyGenContext` to create a context. With this context, use `CSSM_UpdateContextAttributes` to add additional parameters. Finally, call `CSSM_DeleteContext` to dispose of the context after calling this function.

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecKeyCreatePair` (page 28)

Declared In
`SecKey.h`

SecKeyGetCredentials

Returns an access credential for a key.

```
OSStatus SecKeyGetCredentials(
    SecKeyRef keyRef,
    CSSM_ACL_AUTHORIZATION_TAG operation,
    SecCredentialType credentialType,
    const CSSM_ACCESS_CREDENTIALS **outCredentials
);
```

Parameters

`keyRef`
The key for which you want an access credential.

`operation`
The type of operation to be performed with this key. Possible values are listed under "Authorization tag types" in `Security.framework/cssmtype.h`.

`credentialType`
The type of credential requested. See "Key Credential Type Constants" (page 65) for possible values.

`outCredentials`
On return, points to an access credential for the specified key. This pointer remains valid until the key reference is released. Do not attempt to modify or free this data.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
An access credential is required as an input to a number of CSSM functions.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`SecKey.h`

SecKeyGetCSPHandle

Returns the CSSM CSP handle for a key.

```
OSStatus SecKeyGetCSPHandle(
    SecKeyRef keyRef,
    CSSM_CSP_HANDLE *cspHandle
);
```

Parameters

`keyRef`
The key for which you want a CSSM CSP handle.

`cspHandle`
On return, points to the CSSM CSP handle for the specified key. This pointer remains valid until the key reference is released. Do not attempt to modify or free this data.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
A CSSM CSP handle is required as an input to a number of CSSM functions.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`SecKey.h`

SecKeyGetCSSMKey

Retrieves a pointer to the `CSSM_KEY` structure containing the key stored in a keychain item.

```
OSStatus SecKeyGetCSSMKey (
   SecKeyRef key,
   const CSSM_KEY **cssmKey
);
```

Parameters

`key`
A keychain key item object.

`cssmKey`
A pointer to a `CSSM_KEY` structure for the specified key. You should not modify or free this data, because it is owned by the system.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
The `CSSM_KEY` structure is used to represent keys in CSSM and is used as an input value to several CSSM functions. The `CSSM_KEY` structure is valid until the keychain item object is released.

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

Declared In
`SecKey.h`

SecKeyGetTypeID

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

```
CFTypeID SecKeyGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecKeyRef` (page 58) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecKeyRef` (page 58) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

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

Declared In
`SecKey.h`

SecPolicyGetOID

Retrieves a policy's object identifier.

```
OSStatus SecPolicyGetOID (
   SecPolicyRef policyRef,
   CSSM_OID *oid
);
```

Parameters

policyRef
: The policy object for which to obtain the object identifier. You can obtain a policy object with the `SecPolicySearchCopyNext` (page 36) function.

oid
: On return, points to the policy's object identifier. This identifier is owned by the policy object and remains valid until that object is destroyed; do not release it separately.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
The policy's object identifier, in the form of a `CSSM_OID` structure, is used in the CSSM API together with the policy's value. Use the `SecPolicyGetValue` (page 35) function to obtain the value that corresponds to this object identifier.

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

Declared In
`SecPolicy.h`

SecPolicyGetTPHandle

Retrieves the trust policy handle for a policy object.

```
OSStatus SecPolicyGetTPHandle (
   SecPolicyRef policyRef,
   CSSM_TP_HANDLE *tpHandle
);
```

Parameters

policyRef
: The policy object from which to obtain the trust policy handle.

tpHandle
: On return, points to the policy object's trust policy handle. The handle remains valid until the policy object is released.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
The trust policy handle is the CSSM identifier of the trust policy module that is managing the certificate. The trust policy handle is uses as an input to a number of CSSM functions.

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

Declared In
`SecPolicy.h`

SecPolicyGetTypeID

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

```
CFTypeID SecPolicyGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecPolicyRef` (page 58) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecPolicyRef` (page 58) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

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

Declared In
`SecPolicy.h`

SecPolicyGetValue

Retrieves a policy's value.

```
OSStatus SecPolicyGetValue (
   SecPolicyRef policyRef,
   CSSM_DATA *value
);
```

Parameters

policyRef
The policy object for which to retrieve the value.

value
On return, points to the policy's value. This value is owned by the policy object and remains valid until that object is destroyed; do not release it separately.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

A policy's value is defined and interpreted by the policy. If you are using CSSM, you can specify object-identifier–policy-value pairs as input to the `CSSM_TP_POLICYINFO` function. Use the `SecPolicyGetOID` (page 34) function to obtain the object identifier (OID) for a policy.

Depending on how the policy uses the value, the value can be specific to a transaction. Because some other process might be using this policy object, it is best not to assign a new value to the policy using the same policy object. Instead, obtain a new policy object before assigning a new value to the policy.

Availability

Available in Mac OS X v10.2 and later.

See Also

`SecPolicySetValue` (page 38)

Declared In

`SecPolicy.h`

SecPolicySearchCopyNext

Retrieves a policy object for the next policy matching specified search criteria.

```
OSStatus SecPolicySearchCopyNext (
   SecPolicySearchRef searchRef,
   SecPolicyRef *policyRef
);
```

Parameters

`searchRef`

A policy search object specifying the search criteria for this search. You create the policy search object by calling the `SecPolicySearchCreate` (page 36) function.

`policyRef`

On return, points to the policy object for the next policy (if any) matching the specified search criteria. Call the `CFRelease` function to release this object when you are finished with it.

Return Value

A result code. When there are no more policies that match the parameters specified to `SecPolicySearchCreate` (page 36), `errSecPolicyNotFound` is returned. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecPolicySearch.h`

SecPolicySearchCreate

Creates a search object for finding policies.

```
OSStatus SecPolicySearchCreate (
   CSSM_CERT_TYPE certType,
   const CSSM_OID *policyOID,
   const CSSM_DATA *value,
   SecPolicySearchRef *searchRef
);
```

Parameters

certType

The type of certificates a policy uses, as defined in `Security.framework/cssmtype.h`. Permissible values are `CSSM_CERT_X_509v1`, `CSSM_CERT_X_509v2`, and `CSSM_CERT_X_509v3`. If you are unsure of the certificate type, use `CSSM_CERT_X_509v3`.

policyOID

A pointer to a BER-encoded policy object identifier that uniquely specifies the policy. See "AppleX509TP Trust Policies" (page 77) for a list of policies and object identifiers provided by the AppleX509TP module.

value

A pointer to an optional, policy-defined value. The contents of this value depend on the policy object identifier specified. (Note that this parameter refers to the value stored in MDS and is not related to the `value` parameter of the `SecPolicyGetValue` (page 35) function.) Currently the function does not use this parameter; pass `NULL` for this pointer.

searchRef

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

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

You use the search object created by this function in subsequent calls to the `SecPolicySearchCopyNext` (page 36) function to obtain trust policy objects. Policies are stored in the Module Directory Services (MDS) database. MDS is described in detail in "Part 8: Module Directory Service (MDS)" of *Common Security: CDSA and CSSM, version 2 (with corrigenda)* from The Open Group (http://www.opengroup.org/security/cdsa.htm).

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecPolicySearch.h`

SecPolicySearchGetTypeID

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

```
CFTypeID SecPolicySearchGetTypeID (
   void
);
```

Return Value

A value that identifies the opaque type of a `SecPolicySearchRef` (page 59) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecPolicySearchRef` (page 59) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

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

Declared In
`SecPolicySearch.h`

SecPolicySetValue

Sets a policy's value.

```
OSStatus SecPolicySetValue(
    SecPolicyRef policyRef,
    const CSSM_DATA *value
);
```

Parameters

`policyRef`
: The policy object whose value you wish to set.

`value`
: The value to be set into the policy object, replacing any previous value.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
A policy's value is defined and interpreted by the policy. If you are using CSSM, you can specify object-identifier–policy-value pairs as input to the `CSSM_TP_POLICYINFO` function. Use the `SecPolicyGetOID` (page 34) function to obtain the object identifier (OID) for a policy.

Depending on how the policy uses the value, the value can be specific to a transaction. Because some other process might be using this policy object, it is best not to assign a new value to the policy using the same policy object. Instead, obtain a new policy object before assigning a new value to the policy.

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecPolicyGetValue` (page 35)

Declared In
`SecPolicy.h`

SecTrustCopyAnchorCertificates

Retrieves the anchor (root) certificates stored by Mac OS X.

```
OSStatus SecTrustCopyAnchorCertificates (
   CFArrayRef *anchors
);
```

Parameters

anchors

On return, points to an array of certificate objects for trusted anchor (root) certificates, which is the default set of anchors for the caller. Call the `CFRelease` function to release the `CFArrayRef` object when you are finished with it.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

This function retrieves the certificates in the system's store of anchor certificates (see `SecTrustSetAnchorCertificates` (page 46)). You can use the `SecCertificateRef` objects retrieved by this function as input to other functions of this API, such as `SecTrustCreateWithCertificates` (page 40). If you want references to the anchor certificates in a form appropriate for calls to the CSSM API, use the `SecTrustGetCSSMAnchorCertificates` (page 73) function instead.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecTrust.h`

SecTrustCopyCustomAnchorCertificates

Retrieves the custom anchor certificates, if any, used by a given trust.

```
OSStatus SecTrustCopyCustomAnchorCertificates(
    SecTrustRef trust,
    CFArrayRef *anchors
);
```

Parameters

trust

The trust management object from which you wish to retrieve the custom anchor certificates.

anchors

On return, a reference to an array of `SecCertificateRef` objects representing the set of anchor certificates that are considered valid (trusted) anchors by the `SecTrustEvaluate` (page 41) function when verifying a certificate using the trust management object in the `trust` parameter. Returns `NULL` if no custom anchors have been specified. Call the `CFRelease` function to release this object when you are finished with it.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

You can use the `SecTrustSetAnchorCertificates` (page 46) function to set custom anchor certificates.

Availability

Available in Mac OS X v10.5 and later.

See Also
`SecTrustSetAnchorCertificates` (page 46)

Declared In
`SecTrust.h`

SecTrustCopyPolicies

Retrieves the policies used by a given trust management object.

```
OSStatus SecTrustCopyPolicies(
    SecTrustRef trust,
    CFArrayRef *policies
);
```

Parameters

`trust`
: The trust management object whose policies you wish to retrieve.

`policies`
: On return, an array of `SecPolicyRef` (page 58) objects for the policies used by this trust management object. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecTrustSetPolicies` (page 49)

Declared In
`SecTrust.h`

SecTrustCreateWithCertificates

Creates a trust management object based on certificates and policies.

```
OSStatus SecTrustCreateWithCertificates (
   CFArrayRef certificates,
   CFTypeRef policies,
   SecTrustRef *trustRef
);
```

Parameters

`certificates`
: The certificate to be verified, plus any other certificates you think might be useful for verifying the certificate. The certificate to be verified must be the first in the array. If you want to specify only one certificate, you can pass a `SecCertificateRef` object; otherwise, pass an array of `SecCertificateRef` objects.

policies

References to one or more policies to be evaluated. You can pass a single `SecPolicyRef` object, or an array of one or more `SecPolicyRef` objects. Use the `SecPolicySearchCopyNext` (page 36) function to obtain policy objects. If you pass in multiple policies, all policies must verify for the certificate chain to be considered valid.

trustRef

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

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

The trust management object includes a reference to the certificate to be verified, plus pointers to the policies to be evaluated for those certificates. You can optionally include references to other certificates, including anchor certificates, that you think might be in the certificate chain needed to verify the first (leaf) certificate. Any input certificates that turn out to be irrelevant are harmlessly ignored. Call the `SecTrustEvaluate` (page 41) function to evaluate the trust for the returned trust management object.

If not all the certificates needed to verify the leaf certificate are included in the `certificates` parameter, `SecTrustEvaluate` searches for certificates in the keychain search list (see `SecTrustSetKeychains` (page 47)) and in the system's store of anchor certificates (see `SecTrustSetAnchorCertificates` (page 46)). However, you should gain a significant performance benefit by passing in the entire certificate chain, in order, in the `certificates` parameter.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecTrust.h`

SecTrustEvaluate

Evaluates trust for the specified certificate and policies.

```
OSStatus SecTrustEvaluate (
   SecTrustRef trust,
   SecTrustResultType *result
);
```

Parameters

trust

The trust management object to evaluate. A trust management object includes the certificate to be verified plus the policy or policies to be used in evaluating trust. It can optionally also include other certificates to be used in verifying the first certificate. Use the `SecTrustCreateWithCertificates` (page 40) function to create a trust management object.

result

On return, points to a result type reflecting the result of this evaluation. See "Trust Result Type Constants" (page 62) for descriptions of possible values.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

This function evaluates a certificate's validity to establish trust for a particular use—for example, in creating a digital signature or to establish a Secure Sockets Layer connection. Before you call this function, you can optionally call any of the `SecTrustSet...` functions (such as `SecTrustSetParameters` (page 48) or `SecTrustSetVerifyDate` (page 55)) to set values for parameters and options.

The `SecTrustEvaluate` function validates a certificate by verifying its signature plus the signatures of the certificates in its certificate chain, up to the anchor certificate, according to the policy or policies included in the trust management object. For each policy, the function evaluates trust according to the user-specified trust setting (see `SecTrustSetUserTrust` (page 74) and `SecTrustGetUserTrust` (page 73)). For an example of user-specified trust settings, use the Keychain Access utility and look at any certificate.

For each policy, `SecTrustEvaluate` starts with the leaf certificate and checks each certificate in the chain, in turn, for a valid user-specified trust setting. It uses the first such value it finds for the trust evaluation. For example, if the user-specified trust for the leaf certificate is not set, the first intermediate certificate is set to "Always Trust," and one of the other intermediate certificates is set to "Never Trust," `SecTrustEvaluate` trusts the certificate. Thus, you can use a user-specified trust setting for a certificate closer to the leaf to override a setting closer to the anchor.

If there is no user-specified trust setting for the entire certificate chain, the `SecTrustEvaluate` function returns `kSecTrustResultUnspecified` as the result type. In that case, you should call the `SFCertificateTrustPanel` class in the *Security Interface Framework Reference* to let the user specify a trust setting for the certificate. Alternately, you can use a default value. If you use a default value, you should provide a preference setting so that the user can change the default.

If `SecTrustEvaluate` returns `kSecTrustResultRecoverableTrustFailure` as the result type, you can call the `SecTrustGetResult` (page 44) function for details of the problem. Then, as appropriate, you can call one or more of the `SecTrustSet...` functions to correct or bypass the problem, or you can inform the user of the problem and call the `SFCertificateTrustPanel` class to let the user change the trust setting for the certificate. When you think you have corrected the problem, call `SecTrustEvaluate` again. Each time you call `SecTrustEvaluate`, it discards the results of any previous evaluation and replaces them with the new results. If `SecTrustEvaluate` returns `kSecTrustResultFatalTrustFailure`, on the other hand, changing parameter values and calling `SecTrustEvaluate` again is unlikely to be successful.

If not all the certificates needed to verify the leaf certificate are included in the trust management object, then `SecTrustEvaluate` searches for certificates in the keychain search list (see `SecTrustSetKeychains` (page 47)) and in the system's store of anchor certificates (see `SecTrustSetAnchorCertificates` (page 46)).

By default, `SecTrustEvaluate` uses the current date and time when verifying a certificate. However, you can call the `SecTrustSetVerifyDate` (page 55) function before calling `SecTrustEvaluate` to set an other date and time to use when verifying the certificate.

Before you call `SecTrustEvaluate`, you can optionally use the `SecTrustSetParameters` (page 48) function to set one or more actions to modify the evaluation or to pass data required by an action.

The results of the trust evaluation are stored in the trust management object. Call the `SecTrustGetResult` (page 44) function to get more information about the results of the trust evaluation, or the `SecTrustGetCssmResult` (page 43) function to get information about the evaluation in a form that can be passed to CSSM functions.

Availability

Available in Mac OS X v10.2 and later.

Declared In
`SecTrust.h`

SecTrustGetCssmResult

Retrieves the CSSM trust result.

```
OSStatus SecTrustGetCssmResult (
   SecTrustRef trust,
   CSSM_TP_VERIFY_CONTEXT_RESULT_PTR *result
);
```

Parameters

`trust`

A trust management object that has previously been sent to the `SecTrustEvaluate` (page 41) function for evaluation.

`result`

On return, points to the CSSM trust result pointer. You should not modify or free this data, as it is owned by the system.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
After calling the `SecTrustEvaluate` (page 41) function, you can call the `SecTrustGetResult` (page 44) function or the `SecTrustGetCssmResult` function to get information about the certificates in the certificate chain and everything that might be wrong with each certificate. Whereas the `SecTrustGetResult` (page 44) function returns the information in a form that you can interpret without extensive knowledge of CSSM, the `SecTrustGetCssmResult` function returns information in a form that can be passed directly to CSSM functions. See *Common Security: CDSA and CSSM, version 2 (with corrigenda)* from The Open Group (http://www.opengroup.org/security/cdsa.htm for more information about the `CSSM_TP_VERIFY_CONTEXT_RESULT` structure pointed to by the `result` parameter.

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

Declared In
`SecTrust.h`

SecTrustGetCssmResultCode

Retrieves the CSSM result code from the most recent trust evaluation for a trust management object.

```
OSStatus SecTrustGetCssmResultCode(
    SecTrustRef trust,
    OSStatus *resultCode
);
```

Parameters

`trust`

The trust management object for which you wish to retrieve a result code.

resultCode

On return, the CSSM result code produced by the most recent call to the `SecTrustEvaluate` (page 41) function for the trust management object specified in the `trust` parameter. The value of this parameter is undefined if `SecTrustEvaluate` has not been called.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69). Returns `errSecTrustNotAvailable` if the `SecTrustEvaluate` function has not been called for the specified trust.

Discussion

Whereas the `SecTrustEvaluate` function returns one of the Security Framework result codes (see "Certificate, Key, and Trust Services Result Codes" (page 69)), the `SecTrustGetCssmResultCode` function returns the CSSM result code as enumerated in `Security.framework/cssmerr.h`. Other functions that might be of interest are the `SecTrustGetResult` (page 44) function, which returns detailed results for each certificate in the certificate chain, and the `SecTrustGetCssmResult` (page 43) function, which returns the results in a format that can be passed directly to CSSM functions.

Availability

Available in Mac OS X v10.5 and later.

See Also

`SecTrustEvaluate` (page 41)

Declared In

`SecTrust.h`

SecTrustGetResult

Retrieves details on the outcome of a call to the function `SecTrustEvaluate`.

```
OSStatus SecTrustGetResult (
   SecTrustRef trustRef,
   SecTrustResultType *result,
   CFArrayRef *certChain,
   CSSM_TP_APPLE_EVIDENCE_INFO **statusChain
);
```

Parameters

trustRef

A trust management object that has previously been sent to the `SecTrustEvaluate` (page 41) function for evaluation.

result

A pointer to the result type returned in the `result` parameter by the `SecTrustEvaluate` function.

certChain

On return, points to an array of certificates that constitute the certificate chain used to verify the input certificate. Call the `CFRelease` function to release this object when you are finished with it.

`statusChain`
: On return, points to an array of `CSSM_TP_APPLE_EVIDENCE_INFO` structures, one for each certificate in the certificate chain. The first item in the array corresponds to the leaf certificate, and the last item corresponds to the anchor (assuming that verification of the chain did not fail before reaching the anchor certificate). Each structure describes the status of one certificate in the chain. This structure is defined in `cssmapple.h`. Do not attempt to free this pointer; it remains valid until the trust management object is released or until the next call to the function `SecTrustEvaluate` that uses this trust management object.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
After calling the `SecTrustEvaluate` (page 41) function, you can call the `SecTrustGetResult` function or the `SecTrustGetCssmResult` (page 43) function to get detailed information about the results of the evaluation. Whereas the `SecTrustGetResult` function returns the information in a form that you can interpret without extensive knowledge of CSSM, the `SecTrustGetCssmResult` (page 43) function returns information in a form that can be passed directly to CSSM functions.

You can call the `SFCertificateTrustPanel` class in the *Security Interface Framework Reference* to display these results to the user.

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

Declared In
`SecTrust.h`

SecTrustGetTPHandle

Retrieves the trust policy handle.

```
OSStatus SecTrustGetTPHandle (
   SecTrustRef trust,
   CSSM_TP_HANDLE *handle
);
```

Parameters

`trust`
: The trust management object from which to obtain the trust policy handle. A trust management object includes one or more certificates plus the policy or policies to be used in evaluating trust. Use the `SecTrustCreateWithCertificates` (page 40) function to create a trust management object.

`handle`
: On return, points to a CSSM trust policy handle. This handle remains valid until the trust management object is released or until the next call to the function `SecTrustEvaluate` (page 41) that uses this trust management object.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
The trust policy handle is the CSSM identifier of the trust policy module that is managing the certificate. The trust policy handle is used as an input to a number of CSSM functions.

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

Declared In
`SecTrust.h`

SecTrustGetTypeID

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

```
CFTypeID SecTrustGetTypeID (
   void
);
```

Return Value
A value that identifies the opaque type of a `SecTrustRef` (page 59) object.

Discussion
This function returns a value that uniquely identifies the opaque type of a `SecTrustRef` (page 59) object. You can compare this value to the `CFTypeID` identifier obtained by calling the `CFGetTypeID` function on a specific object. These values might change from release to release or platform to platform.

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

Declared In
`SecTrust.h`

SecTrustSetAnchorCertificates

Sets the anchor certificates used when evaluating a trust management object.

```
OSStatus SecTrustSetAnchorCertificates (
   SecTrustRef trust,
   CFArrayRef anchorCertificates
);
```

Parameters

`trust`
The trust management object containing the certificate you want to evaluate. A trust management object includes the certificate to be verified plus the policy or policies to be used in evaluating trust. It can optionally also include other certificates to be used in verifying the first certificate. Use the `SecTrustCreateWithCertificates` (page 40) function to create a trust management object.

`anchorCertificates`
A reference to an array of `SecCertificateRef` objects representing the set of anchor certificates that are to be considered valid (trusted) anchors by the `SecTrustEvaluate` (page 41) function when verifying a certificate. Pass `NULL` to restore the default set of anchor certificates.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

The `SecTrustEvaluate` (page 41) function looks for an anchor certificate in the array of certificates specified by the `SecTrustSetAnchorCertificates` function, or uses a default set provided by the system. In Mac OS X v10.3, for example, the default set of anchors is in the keychain file /System/Library/Keychains/X509Anchors. If you want to create a set of anchor certificates by modifying the default set, call the `SecTrustCopyAnchorCertificates` (page 38) function to obtain the current set of anchor certificates, modify that set as you wish, and create a new array of certificates. Then call `SecTrustSetAnchorCertificates` with the modified array.

The list of custom anchor certificates is stored in the trust management object and can be retrieved with the `SecTrustCopyCustomAnchorCertificates` (page 39) function.

Use the `SecTrustSetKeychains` (page 47) function to set the keychains searched for intermediate certificates in the certificate chain.

Important: Calling this function without also calling `SecTrustSetAnchorCertificatesOnly` disables the trusting of any anchors other than the ones specified by this function call.

Availability

Available in Mac OS X v10.2 and later.

See Also

`SecTrustCopyCustomAnchorCertificates` (page 39)

Declared In

`SecTrust.h`

SecTrustSetKeychains

Sets the keychains searched for intermediate certificates when evaluating a trust management object.

```
OSStatus SecTrustSetKeychains (
   SecTrustRef trust,
   CFTypeRef keychainOrArray
);
```

Parameters

`trust`

The trust management object containing the certificate you want to evaluate. A trust management object includes the certificate to be verified plus the policy or policies to be used in evaluating trust. It can optionally also include other certificates to be used in verifying the first certificate. Use the `SecTrustCreateWithCertificates` (page 40) function to create a trust management object.

`keychainOrArray`

A keychain object for a single keychain to search, an array of keychain objects for a set of keychains to search, or `NULL` to search the user's default keychain search list. To prevent the `SecTrustEvaluate` (page 41) function from searching any keychains at all, pass a `CFArrayRef` array with no elements.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
By default, `SecTrustEvaluate` (page 41) uses the user's keychain search list to look for intermediate certificates in the certificate chain. Use the `SecTrustSetKeychains` function to change the set of keychains to be searched. If you want to modify the default set of keychains, first call the `SecKeychainCopySearchList` function (see *Keychain Services Reference*) to obtain the current keychain search list, modify that set as you wish, and create a new search list. Then you can call `SecTrustSetKeychains` with the modified list.

Use the `SecTrustSetAnchorCertificates` (page 46) function to set the array of anchor certificates searched.

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

Declared In
`SecTrust.h`

SecTrustSetParameters

Sets the action and action data for a trust management object.

```
OSStatus SecTrustSetParameters (
   SecTrustRef trustRef,
   CSSM_TP_ACTION action,
   CFDataRef actionData
);
```

Parameters

`trustRef`
The trust management object to which you want to add an action or set action data. A trust management object includes one or more certificates plus the policy or policies to be used in evaluating trust. Use the `SecTrustCreateWithCertificates` (page 40) function to create a trust management object.

`action`
A CSSM trust action. Pass `CSSM_TP_ACTION_DEFAULT` for the default action. Other actions available, if any, are described in the documentation for the trust policy module. For the AppleX509TP module, see the *Security Release Notes*.

`actionData`
A reference to action data. "Action Data Flags" (page 63) lists possible values for this parameter for the AppleX509TP trust policy module's default action. For other actions (if any), the possible values for the action data are specified in the *Security Release Notes*.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
Before you call `SecTrustEvaluate` (page 41), you can optionally use this function to set one or more action flags or to set action data. Actions, where available, affect the trust evaluation for all policies being evaluated. For example, if you set the action data for the default action to `CSSM_TP_ACTION_ALLOW_EXPIRED`, then the `SecTrustEvaluate` function ignores the certificate's expiration date and time.

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

Declared In
`SecTrust.h`

SecTrustSetPolicies

Set the policies to use in an evaluation.

```
OSStatus SecTrustSetPolicies(
    SecTrustRef trust,
    CFTypeRef policies
);
```

Parameters

`trust`
The trust management object whose policy list you wish to set.

`policies`
An array of one or more `SecPolicyRef` (page 58) objects for the policies to be used by this trust management object. A single policy object of type `SecPolicyRef` may also be passed, representing an array of one policy.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
The policies you set with this function replace any already in the trust management object.

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecTrustCopyPolicies` (page 40)

Declared In
`SecTrust.h`

SecTrustSettingsCopyCertificates

Obtains an array of all certificates that have trust settings in a specific trust settings domain.

```
OSStatus SecTrustSettingsCopyCertificates(
    SecTrustSettingsDomain domain,
    CFArrayRef *certArray
);
```

Parameters

`domain`
The trust settings domain for which you want a list of certificates. For possible values, see "Trust Settings Domain Constants" (page 65).

`certArray`
On return, an array of `SecCertificateRef` objects representing the certificates that have trust settings in the specified domain. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69). Returns `errSecNoTrustSettings` if no trust settings exist for the specified domain.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`SecTrustSettings.h`

SecTrustSettingsCopyModificationDate

Obtains the date and time at which a certificate's trust settings were last modified.

```
OSStatus SecTrustSettingsCopyModificationDate(
    SecCertificateRef certRef,
    SecTrustSettingsDomain domain,
    CFDateRef *modificationDate
);
```

Parameters

`certRef`
The certificate for which you wish to obtain the modification time. Pass the value `kSecTrustSettingsDefaultRootCertSetting` to obtain the modification time for the default root certificate trust settings for the domain.

`domain`
The trust settings domain of the trust settings for which you wish to obtain the modification time. For possible values, see "Trust Settings Domain Constants" (page 65).

`modificationDate`
On return, the date and time at which the certificate's trust settings were last modified. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69). Returns `errSecItemNotFound` if no trust settings exist for the specified certificate and domain.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`SecTrustSettings.h`

SecTrustSettingsCopyTrustSettings

Obtains the trust settings for a certificate.

```
OSStatus SecTrustSettingsCopyTrustSettings(
    SecCertificateRef certRef,
    SecTrustSettingsDomain domain,
    CFArrayRef *trustSettings
);
```

Parameters

certRef

The certificate for which you want the trust settings. Pass the value `kSecTrustSettingsDefaultRootCertSetting` to obtain the default root certificate trust settings for the domain.

domain

The trust settings domain of the trust settings that you wish to obtain. For possible values, see "Trust Settings Domain Constants" (page 65).

trustSettings

On return, an array of `CFDictionary` objects specifying the trust settings for the certificate. For the contents of the dictionaries, see the discussion below. Call the `CFRelease` function to release this object when you are finished with it.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69). Returns `errSecItemNotFound` if no trust settings exist for the specified certificate and domain.

Discussion

Each certificate's trust settings are expressed as a `CFArray` that includes any number (including zero) of dictionaries of type `CFDictionary`, each of which comprises one set of usage constraints. Each usage constraints dictionary contains zero or one of each of the following key-value pairs:

Key	Value
`kSecTrustSettings-Policy`	A policy object (`SecPolicyRef`) specifying the certificate verification policy; for example: SSL, SMIME. Use the `SecPolicySearchCopyNext` (page 36) function to obtain a policy object.
`kSecTrustSettings-Application`	A trusted application reference (`SecTrustedApplicationRef`) for the application checking the certificate's trust settings. Use the `SecTrusted-ApplicationCreateFromPath` function to get this reference.
`kSecTrustSettings-PolicyString`	A `CFString` containing policy-specific data. For the SMIME policy, this string contains an email address. For the SSL policy, it contains a host name.
`kSecTrustSettings-KeyUsage`	A `CFNumber` containing an `SInt32` value specifying the operations for which the encryption key in this certificate can be used. For possible values, see "Trust Settings Key Use Constants" (page 66).

Key	Value
`kSecTrustSettings-Result`	A `CFNumber` containing an `SInt32` value indicating the effective trust setting for this usage constraints dictionary. A given usage constraints dictionary is included in the evaluation of trust for a certificate only if the specified policy, application, and key use match the use for which the certificate is being evaluated. If this is the case, then the value of the `kSecTrustSettingsResult` key is ORed with the results from other dictionaries to determine the overall trust setting for the certificate. If this key is not present, a default value of `kSecTrustSettings-ResultTrustRoot` is assumed. Because only a root certificate can have this value, a usage constraints dictionary for a non-root certificate that is missing this key is not valid. Possible values for this key are listed in "Trust Settings Result Constants" (page 68).
`kSecTrustSettings-AllowedError`	A `CFNumber` containing an `SInt32` value indicating a `CSSM_RETURN` result code which, if encountered during certificate verification, is ignored for that certificate. These "allowed error" values are applied to the evaluation only if the usage constraints dictionary meets the criteria described with the `kSecTrustSettingsResult` key. A usage constraint dictionary with no constraints but with an allowed error value causes that error to always be allowed when the certificate is being evaluated.

The overall trust settings for a certificate are the sum of all the usage constraints dictionaries that match the use for which that certificate is being evaluated. Trust settings for a given use apply if *any* of the dictionaries in the certificate's trust settings array satisfies the specified use. Thus, when a certificate has multiple usage constraints dictionaries in its trust settings array, the overall trust settings for the certificate are:

((usage constraint dictionary 0 component 0) AND (usage constraint dictionary 0 component 1) AND (...)) OR ((usage constraint dictionary 1 component 0) AND (usage constraint dictionary 1 component 1) AND (...)) OR (...) ...

If the value of the `kSecTrustSettingsResult` component is *not* `kSecTrustSettingsResultUnspecified` for a usage constraints dictionary that has no constraints, the default value `kSecTrustSettingsResultTrustRoot` is assumed. To specify a value for the `kSecTrustSettingsAllowedError` component without explicitly trusting or distrusting the associated certificate, specify a value of `kSecTrustSettingsResultUnspecified` for the `kSecTrustSettingsResult` component. An empty trust settings array (that is, the `trustSettings` parameter returns a valid but empty `CFArray`) means "always trust this certificate" with an overall trust setting for the certificate of `kSecTrustSettingsResultTrustRoot`. Note that an empty trust settings array is not the same as no trust settings (the `trustSettings` parameter returns `NULL`), which means "this certificate must be verified to a known trusted certificate". Note the distinction between the results `kSecTrustSettingsResultTrustRoot` and `kSecTrustSettingsResultTrustAsRoot`: The former can only be applied to root (self-signed) certificates; the latter can only be applied to non-root certificates. Therefore, an empty trust settings array for a non-root certificate is invalid, because the default value of `kSecTrustSettingsResultTrustRoot` is not valid for a non-root certificate. When making changes to the per-user trust settings, the user is prompted with an alert panel asking for authentication (user name and password or other credentials normally used for login). Therefore, it is not possible to modify per-user trust settings when not running in a GUI environment (that is, when the user is not logged in via the login window). When making changes to the system-wide trust settings, the user is prompted with an alert panel asking for an administrator's name and password unless the calling process is running as root, in which case no futher authentication is needed.

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecTrustSettingsSetTrustSettings` (page 54)

Declared In
`SecTrustSettings.h`

SecTrustSettingsCreateExternalRepresentation

Obtains an external, portable representation of the specified domain's trust settings.

```
OSStatus SecTrustSettingsCreateExternalRepresentation(
    SecTrustSettingsDomain domain,
    CFDataRef *trustSettings
);
```

Parameters

`domain`
: The trust settings domain for which you want an external representation of trust settings. For possible values, see "Trust Settings Domain Constants" (page 65).

`trustSettings`
: An external representation of the domain's trust settings. Call the `CFRelease` function to release this object when you are finished with it.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69). Returns `errSecNoTrustSettings` if no trust settings exist for the specified domain.

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecTrustSettingsImportExternalRepresentation` (page 53)

Declared In
`SecTrustSettings.h`

SecTrustSettingsImportExternalRepresentation

Imports trust settings into a trust domain.

```
OSStatus SecTrustSettingsImportExternalRepresentation(
    SecTrustSettingsDomain domain,
    CFDataRef trustSettings
);
```

Parameters

`domain`
: The trust settings domain into which you want to import trust settings. For possible values, see "Trust Settings Domain Constants" (page 65).

trustSettings

An external representation of the trust settings (created by the `SecTrustSettingsCreateExternalRepresentation` (page 53) function) that you want to import.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Availability

Available in Mac OS X v10.5 and later.

See Also

`SecTrustSettingsCreateExternalRepresentation` (page 53)

Declared In

`SecTrustSettings.h`

SecTrustSettingsRemoveTrustSettings

Deletes the trust settings for a certificate.

```
OSStatus SecTrustSettingsRemoveTrustSettings(
    SecCertificateRef certRef,
    SecTrustSettingsDomain domain);
```

Parameters

certRef

The certificate whose trust settings you wish to remove. Pass the value `kSecTrustSettingsDefaultRootCertSetting` to remove the default root certificate trust settings for the domain.

domain

The trust settings domain for which you wish to remove the trust settings. For possible values, see "Trust Settings Domain Constants" (page 65).

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69). Returns `errSecItemNotFound` if no trust settings exist for the certificate.

Discussion

If a certificate has no trust settings, the certificate must be verified to a known, trusted certificate.

Availability

Available in Mac OS X v10.5 and later.

See Also

`SecTrustSettingsSetTrustSettings` (page 54)

Declared In

`SecTrustSettings.h`

SecTrustSettingsSetTrustSettings

Specifies trust settings for a certificate.

```
OSStatus SecTrustSettingsSetTrustSettings(
    SecCertificateRef certRef,
    SecTrustSettingsDomain domain,
    CFTypeRef trustSettingsDictOrArray);
```

Parameters

`certRef`
: The certificate for which you want to specify the trust settings. Pass the value `kSecTrustSettingsDefaultRootCertSetting` to set the default root certificate trust settings for the domain.

`domain`
: The trust settings domain of the trust settings that you wish to specify. For possible values, see "Trust Settings Domain Constants" (page 65).

`trustSettings`
: On return, an array of `CFDictionary` objects specifying the trust settings for the certificate. For the contents of the dictionaries, see the discussion below. Call the `CFRelease` function to release this object when you are finished with it.

`trustSettingsDictOrArray`
: The trust settings you wish to specify for this certificate, in the form of a `CFDictionary` object, a `CFArray` of `CFDictionary` objects, or `NULL`. The contents of `CFDictionary` objects used to specify trust settings are detailed in the `SecTrustSettingsCopyTrustSettings` (page 50) function description. Pass `NULL` if you want to specify an empty trust settings array.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
If you pass `NULL` for the `trustSettingsDictOrArray` parameter, then the trust settings for this certificate are stored as an empty trust settings array, indicating "always trust this root certificate regardless of use." This setting is valid only for a self-signed (root) certificate. To instead remove all trust settings for the certificate (interpreted as "this certificate must be verified to a known trusted certificate"), use the `SecTrustSettingsRemoveTrustSettings` (page 54) function.

If the specified certificate already has trust settings in the specified domain, this function replaces them.

Availability
Available in Mac OS X v10.5 and later.

See Also
`SecTrustSettingsCopyTrustSettings` (page 50)
`SecTrustSettingsRemoveTrustSettings` (page 54)

Declared In
`SecTrustSettings.h`

SecTrustSetVerifyDate

Sets the date and time against which the certificates in a trust management object are verified.

```
OSStatus SecTrustSetVerifyDate (
   SecTrustRef trust,
   CFDateRef verifyDate
);
```

Parameters

`trust`

The trust management object whose verification date you want to set. A trust management object includes one or more certificates plus the policy or policies to be used in evaluating trust. Use the `SecTrustCreateWithCertificates` (page 40) function to create a trust management object.

`verifyDate`

The date and time to use when verifying the certificate.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

By default, the `SecTrustEvaluate` (page 41) function uses the current date and time when verifying a certificate. However, you can use `SecTrustSetVerifyDate` to set another date and time to use when verifying a certificate. For example, you can determine whether the certificate was valid when the document was signed rather than whether it's valid at the present time.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`SecTrust.h`

Data Types

CSSM_TP_APPLE_EVIDENCE_INFO

Contains information about a certificate evaluation.

```
typedef struct {
    CSSM_TP_APPLE_CERT_STATUS   StatusBits;
    uint32                      NumStatusCodes
    CSSM_RETURN                 *StatusCodes;
    uint32                      Index;
    CSSM_DL_DB_HANDLE           DlDbHandle
    CSSM_DB_UNIQUE_RECORD_PTR   UniqueRecord;
} CSSM_TP_APPLE_EVIDENCE_INFO;
```

Fields

`StatusBits`

Indicates whether the certificate is valid and where it was found; see "Certificate Status Constants" (page 61).

`NumStatusCodes`

The number of `CSSM_RETURN` structures returned in the `StatusCodes` field.

`StatusCodes`
: An array of `CSSM_RETURN` values indicating what problems were found with the certificate. Apple-specific values are in `cssmapple.h`. Standard CSSM values are defined in `cssmerr.h` and are discussed in "Error Codes and Error Values" in the "Trust Policy Services API" chapter of *Common Security: CDSA and CSSM, version 2 (with corrigenda)* from The Open Group (http://www.open-group.org/security/cdsa.htm

`Index`
: An index into the standard set of certificates or anchor certificates if the certificate came from one of those sets.

`DlDbHandle`
: A CSSM object that identifies a particular database. This field is used if the certificate did not come from the standard set of certificates or anchor certificates. This value is useful only as input to functions in the CSSM API.

`UniqueRecord`
: A CSSM object that identifies a particular record in a database. This field is used if the certificate did not come from the standard set of certificates or anchor certificates. This value is useful only as in input to functions in the CSSM API.

Discussion
An array of these structures is returned by the `SecTrustGetResult` (page 44) function; each one describes a certificate in the certificate chain.

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

Declared In
`cssmapple.h`

SecCertificateRef

Abstract Core Foundation-type object representing an X.509 certificate.

```
typedef struct __SecCertificate *SecCertificateRef;
```

Discussion
A `SecCertificateRef` object for a certificate that is stored in a keychain can be safely cast to a `SecKeychainItemRef` for manipulation as a keychain item. On the other hand, if the `SecCertificateRef` is not stored in a keychain, casting the object to a `SecKeychainItemRef` and passing it to Keychain Services functions returns errors.

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

Declared In
`SecBase.h`

SecIdentityRef

Abstract Core Foundation-type object representing an identity.

```
typedef struct __SecIdentity *SecIdentityRef;
```

Discussion
A `SecIdentityRef` object contains a `SecKeyRef` object and an associated `SecCertificateRef` object.

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

Declared In
`SecBase.h`

SecIdentitySearchRef

Contains information about an identity search.

```
typedef struct OpaqueSecIdentitySearchRef *SecIdentitySearchRef;
```

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

Declared In
`SecIdentitySearch.h`

SecKeyRef

Abstract Core Foundation-type object representing an asymmetric key.

```
typedef struct __SecKey *SecKeyRef;
```

Discussion
A `SecKeyRef` object for a key that is stored in a keychain can be safely cast to a `SecKeychainItemRef` for manipulation as a keychain item. On the other hand, if the `SecKeyRef` is not stored in a keychain, casting the object to a `SecKeychainItemRef` and passing it to Keychain Services functions returns errors.

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

Declared In
`SecBase.h`

SecPolicyRef

Contains information about a policy.

```
typedef struct OpaqueSecPolicyRef *SecPolicyRef;
```

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

Declared In
`SecBase.h`

SecPolicySearchRef

Contains information about a policy search.

```
typedef struct OpaquePolicySearchRef *SecPolicySearchRef;
```

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

Declared In
`SecPolicySearch.h`

SecPublicKeyHash

Represents a 20-byte public key hash.

```
typedef UInt8 SecPublicKeyHash[20];
```

Discussion
The `SecPublicKeyHash` type represents a hash of a public key. You can use the constant `kSecPublicKeyHashItemAttr` as input to functions in the Keychain Services API to set or retrieve a certificate attribute value of this type. See *Keychain Services Reference* for information about getting and setting attribute values.

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

Declared In
`SecKeychainItem.h`

SecTrustRef

Contains information about trust management.

```
typedef struct __SecTrust *SecTrustRef;
```

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

Declared In
`SecTrust.h`

SecTrustUserSetting

Represents user-specified trust settings.

```
typedef SecTrustResultType SecTrustUserSetting;
```

Discussion
See "Trust Result Type Constants" (page 62) for possible values.

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

Declared In
`SecTrust.h`

Constants

Certificate Item Attribute Constants

Indicates certificate item attributes.

```
enum
{
    kSecSubjectItemAttr              = 'subj',
    kSecIssuerItemAttr               = 'issu',
    kSecSerialNumberItemAttr         = 'snbr',
    kSecPublicKeyHashItemAttr        = 'hpky',
    kSecSubjectKeyIdentifierItemAttr = 'skid',
    kSecCertTypeItemAttr             = 'ctyp',
    kSecCertEncodingItemAttr         = 'cenc'
};
```

Constants

`kSecSubjectItemAttr`
DER-encoded subject distinguished name.

Available in Mac OS X v10.2 and later.

Declared in `SecCertificate.h`.

`kSecIssuerItemAttr`
DER-encoded issuer distinguished name.

Available in Mac OS X v10.2 and later.

Declared in `SecCertificate.h`.

`kSecSerialNumberItemAttr`
DER-encoded certificate serial number.

Available in Mac OS X v10.2 and later.

Declared in `SecCertificate.h`.

`kSecPublicKeyHashItemAttr`
Public key hash.

Available in Mac OS X v10.2 and later.

Declared in `SecCertificate.h`.

`kSecSubjectKeyIdentifierItemAttr`
Subject key identifier.

Available in Mac OS X v10.2 and later.

Declared in `SecCertificate.h`.

`kSecCertTypeItemAttr`
Certificate type.

Available in Mac OS X v10.2 and later.

Declared in `SecCertificate.h`.

`kSecCertEncodingItemAttr`
: Certificate encoding.

  Available in Mac OS X v10.2 and later.

  Declared in `SecCertificate.h`.

Certificate Status Constants

Specifies the status of a certificate.

```
typedef uint32 CSSM_TP_APPLE_CERT_STATUS;
enum
{
    CSSM_CERT_STATUS_EXPIRED            = 0x00000001,
    CSSM_CERT_STATUS_NOT_VALID_YET      = 0x00000002,
    CSSM_CERT_STATUS_IS_IN_INPUT_CERTS  = 0x00000004,
    CSSM_CERT_STATUS_IS_IN_ANCHORS      = 0x00000008,
    CSSM_CERT_STATUS_IS_ROOT            = 0x00000010,
    CSSM_CERT_STATUS_IS_FROM_NET        = 0x00000020
};
```

Constants

`CSSM_CERT_STATUS_EXPIRED`
: The certificate has expired.

  Available in Mac OS X v10.2 and later.

  Declared in `cssmapple.h`.

`CSSM_CERT_STATUS_NOT_VALID_YET`
: The certificate is not yet valid. In addition to the expiration, or "Not Valid After," date and time, each certificate has a "Not Valid Before" date and time.

  Available in Mac OS X v10.2 and later.

  Declared in `cssmapple.h`.

`CSSM_CERT_STATUS_IS_IN_INPUT_CERTS`
: This is one of the certificates included in the array of certificates passed to the `SecTrustCreateWithCertificates` (page 40) function.

  Available in Mac OS X v10.2 and later.

  Declared in `cssmapple.h`.

`CSSM_CERT_STATUS_IS_IN_ANCHORS`
: This certificate was found in the system's store of anchor certificates (see `SecTrustSetAnchorCertificates` (page 46)).

  Available in Mac OS X v10.2 and later.

  Declared in `cssmapple.h`.

`CSSM_CERT_STATUS_IS_ROOT`
: The certificate is a root certificate. If this bit is set but the `CSSM_CERT_STATUS_IS_IN_ANCHORS` bit is not, then this is an untrusted anchor.

  Available in Mac OS X v10.2 and later.

  Declared in `cssmapple.h`.

`CSSM_CERT_STATUS_IS_FROM_NET`
: The certificate was obtained through some mechanism other than the certificates stored by the operating system and those passed into the `SecTrustCreateWithCertificates` (page 40) function. For example, the certificate might have been fetched over a network.

  Available in Mac OS X v10.3 and later.

  Declared in `cssmapple.h`.

Discussion

If none of these bits are set, the certificate came from a standard certificate search; see the description of the `SecTrustSetKeychains` (page 47) function.

Trust Result Type Constants

Specifies the trust result type.

```
typedef enum {
    kSecTrustResultInvalid,
    kSecTrustResultProceed,
    kSecTrustResultConfirm,
    kSecTrustResultDeny,
    kSecTrustResultUnspecified,
    kSecTrustResultRecoverableTrustFailure,
    kSecTrustResultFatalTrustFailure,
    kSecTrustResultOtherError
} SecTrustResultType;
```

Constants

`kSecTrustResultInvalid`
: Invalid setting or result. Usually, this result indicates that the `SecTrustEvaluate` (page 41) function did not complete successfully.

  Available in Mac OS X v10.2 and later.

  Declared in `SecTrust.h`.

`kSecTrustResultProceed`
: The user indicated that you may trust the certificate for the purposes designated in the specified policies. This value may be returned by the `SecTrustEvaluate` (page 41) function or stored as part of the user trust settings. In the Keychain Access utility, this value is termed “Always Trust.”

  Available in Mac OS X v10.2 and later.

  Declared in `SecTrust.h`.

`kSecTrustResultConfirm`
: Confirmation from the user is required before proceeding. This value may be returned by the `SecTrustEvaluate` (page 41) function or stored as part of the user trust settings. In the Keychain Access utility, this value is termed “Ask Permission.”

  Available in Mac OS X v10.2 and later.

  Declared in `SecTrust.h`.

`kSecTrustResultDeny`
: The user specified that the certificate should not be trusted. This value may be returned by the `SecTrustEvaluate` (page 41) function or stored as part of the user trust settings. In the Keychain Access utility, this value is termed “Never Trust.”

  Available in Mac OS X v10.2 and later.

  Declared in `SecTrust.h`.

`kSecTrustResultUnspecified`
: The user did not specify a trust setting. This value may be returned by the `SecTrustEvaluate` (page 41) function or stored as part of the user trust settings. In the Keychain Access utility, this value is termed "Use System Policy." This is the default user setting.

  Available in Mac OS X v10.2 and later.

  Declared in `SecTrust.h`.

`kSecTrustResultRecoverableTrustFailure`
: Trust denied; retry after changing settings. For example, if trust is denied because the certificate has expired, you can ask the user whether to trust the certificate anyway. If the user answers yes, then use the `SecTrustSetUserTrust` (page 74) function to set the user trust setting to `kSecTrustResultProceed` and call `SecTrustEvaluate` (page 41) again. This value may be returned by the `SecTrustEvaluate` (page 41) function but not stored as part of the user trust settings.

  Available in Mac OS X v10.2 and later.

  Declared in `SecTrust.h`.

`kSecTrustResultFatalTrustFailure`
: Trust denied; no simple fix is available. For example, if a certificate cannot be verified because it is corrupted, trust cannot be established without replacing the certificate. This value may be returned by the `SecTrustEvaluate` (page 41) function but not stored as part of the user trust settings.

  Available in Mac OS X v10.2 and later.

  Declared in `SecTrust.h`.

`kSecTrustResultOtherError`
: A failure other than that of trust evaluation; for example, an internal failure of the `SecTrustEvaluate` (page 41) function. This value may be returned by the `SecTrustEvaluate` (page 41) function but not stored as part of the user trust settings.

  Available in Mac OS X v10.2 and later.

  Declared in `SecTrust.h`.

Discussion
These constants may be returned by the `SecTrustEvaluate` (page 41) function or stored as one of the user trust settings (see `SecTrustSetUserTrust` (page 74)), as noted. When evaluating user trust, both `SecTrustGetUserTrust` (page 73) and `SecTrustEvaluate` start with the leaf certificate and work through the chain down to the anchor. The `SecTrustGetUserTrust` function returns the user trust setting of the first certificate for which the setting is other than `kSecTrustResultUnspecified`. Similarly, the function uses the user trust setting of the first certificate for which the setting is other than `kSecTrustResultUnspecified`, regardless of the user trust settings of other certificates in the chain.

Action Data Flags

Specifies options for the AppleX509TP trust policy module's default action.

```
typedef uint32 CSSM_APPLE_TP_ACTION_FLAGS;
enum {
    CSSM_TP_ACTION_ALLOW_EXPIRED        = 0x00000001,
    CSSM_TP_ACTION_LEAF_IS_CA           = 0x00000002,
    CSSM_TP_ACTION_FETCH_CERT_FROM_NET  = 0x00000004,
    CSSM_TP_ACTION_ALLOW_EXPIRED_ROOT   = 0x00000008
};
```

Constants

`CSSM_TP_ACTION_ALLOW_EXPIRED`

Ignore the expiration date and time for all certificates.

Available in Mac OS X v10.2 and later.

Declared in `cssmapple.h`.

`CSSM_TP_ACTION_LEAF_IS_CA`

First certificate is that of a certification authority (CA). By formal definition, a valid certificate chain must begin with a certificate that is not a CA. Set this bit if you want to validate a partial chain, starting with a CA and working toward the anchor, or if you want to evaluate a single self-signed certificate as a one-certificate "chain" for testing purposes.

Available in Mac OS X v10.3 and later.

Declared in `cssmapple.h`.

`CSSM_TP_ACTION_FETCH_CERT_FROM_NET`

Enable fetching intermediate certificates over the network using http or LDAP.

Available in Mac OS X v10.3 and later.

Declared in `cssmapple.h`.

`CSSM_TP_ACTION_ALLOW_EXPIRED_ROOT`

Ignore the expiration date and time for root certificates only.

Available in Mac OS X v10.2 and later.

Declared in `cssmapple.h`.

Discussion

See `SecTrustSetParameters` (page 48) for more information about actions.

System Identity Domains

Domains for which you can set or obtain a system-wide identity.

```
const CFStringRef kSecIdentityDomainDefault;
const CFStringRef kSecIdentityDomainKerberosKDC;
```

Constants

`kSecIdentityDomainDefault`

The system-wide default identity.

Available in Mac OS X v10.5 and later.

Declared in `SecIdentity.h`.

`kSecIdentityDomainKerberosKDC`

Kerberos Key Distribution Center (KDC) identity.

Available in Mac OS X v10.5 and later.

Declared in `SecIdentity.h`.

Discussion
These constants can be used with the `SecIdentitySetSystemIdentity` (page 28) and `SecIdentityCopySystemIdentity` (page 23) functions.

Key Credential Type Constants

The credential type to be returned by `SecKeyGetCredentials` (page 31).

```
typedef uint32 SecCredentialType;

enum
{
    kSecCredentialTypeDefault = 0,
    kSecCredentialTypeWithUI,
    kSecCredentialTypeNoUI
};
```

Constants

`kSecCredentialTypeDefault`
: The default setting for determining whether to present UI is used.

  The default setting can be changed with a call to `SecKeychainSetUserInteractionAllowed`.

  Available in Mac OS X v10.5 and later.

  Declared in `SecKey.h`.

`kSecCredentialTypeWithUI`
: Keychain operations on keys that have this credential are allowed to present UI if required.

  Available in Mac OS X v10.5 and later.

  Declared in `SecKey.h`.

`kSecCredentialTypeNoUI`
: Keychain operations on keys that have this credential are not allowed to present UI, and will fail if UI is required.

  Available in Mac OS X v10.5 and later.

  Declared in `SecKey.h`.

Discussion
See the section "Servers and the Keychain" in the Keychain Services Tasks chapter of *Keychain Services Programming Guide* for information on the use of UI with keychain tasks.

Trust Settings Domain Constants

The trust settings domains used by the trust settings API.

```
enum {     kSecTrustSettingsDomainUser = 0,     kSecTrustSettingsDomainAdmin,
 kSecTrustSettingsDomainSystem }; typedef uint32 SecTrustSettingsDomain;
```

Constants

`kSecTrustSettingsDomainUser`
: Per-user trust settings.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsDomainAdmin`
Locally administered, system-wide trust settings.

Administrator privileges are required to make changes to this domain.

Available in Mac OS X v10.5 and later.

Declared in `SecTrustSettings.h`.

`kSecTrustSettingsDomainSystem`
System trust settings.

These trust settings are immutable and comprise the set of trusted root certificates supplied in Mac OS X. These settings are read-only, even by root.

Available in Mac OS X v10.5 and later.

Declared in `SecTrustSettings.h`.

Trust Settings Key Use Constants

Allowed uses for the encryption key in a certificate.

```
enum {
    kSecTrustSettingsKeyUseSignature        = 0x00000001,
    kSecTrustSettingsKeyUseEnDecryptData    = 0x00000002,
    kSecTrustSettingsKeyUseEnDecryptKey     = 0x00000004,
    kSecTrustSettingsKeyUseSignCert         = 0x00000008,
    kSecTrustSettingsKeyUseSignRevocation   = 0x00000010,
    kSecTrustSettingsKeyUseKeyExchange      = 0x00000020,
    kSecTrustSettingsKeyUseAny              = 0xffffffff
};
typedef uint32 SecTrustSettingsKeyUsage;
```

Constants

`kSecTrustSettingsKeyUseSignature`
The key can be used to sign data or verify a signature.

Available in Mac OS X v10.5 and later.

Declared in `SecTrustSettings.h`.

`kSecTrustSettingsKeyUseEnDecryptData`
The key can be used to encrypt or decrypt data.

Available in Mac OS X v10.5 and later.

Declared in `SecTrustSettings.h`.

`kSecTrustSettingsKeyUseEnDecryptKey`
The key can be used to encrypt or decrypt (wrap or unwrap) a key.

Private keys must be wrapped before they can be exported from a keychain.

Available in Mac OS X v10.5 and later.

Declared in `SecTrustSettings.h`.

`kSecTrustSettingsKeyUseSignCert`
The key can be used to sign a certificate or verify a signature.

Available in Mac OS X v10.5 and later.

Declared in `SecTrustSettings.h`.

`kSecTrustSettingsKeyUseSignRevocation`
: The key can be used to sign an OCSP (online certificate status protocol) message or CRL (certificate verification list), or to verify a signature.

  OCSP messages and CRLs are used to revoke certificates.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsKeyUseKeyExchange`
: The key is a private key that has been shared using a key exchange protocol, such as Diffie-Hellman key exchange.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsKeyUseAny`
: The key can be used for any purpose.

  This is the default key-use setting if no other key use is specified.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

Trust Settings Usage Constraints Dictionary Keys

The keys in one usage constraints dictionary.

```
#define kSecTrustSettingsPolicy          CFSTR("kSecTrustSettingsPolicy")
#define kSecTrustSettingsApplication     CFSTR("kSecTrustSettingsApplication")
#define kSecTrustSettingsPolicyString    CFSTR("kSecTrustSettingsPolicyString")
#define kSecTrustSettingsKeyUsage        CFSTR("kSecTrustSettingsKeyUsage")
#define kSecTrustSettingsAllowedError    CFSTR("kSecTrustSettingsAllowedError")
#define kSecTrustSettingsResult          CFSTR("kSecTrustSettingsResult")
```

Constants

`kSecTrustSettingsPolicy`
: A policy object (`SecPolicyRef`) specifying the certificate verification policy.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsApplication`
: A trusted application reference (`SecTrustedApplicationRef`) for the application checking the certificate's trust settings.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsPolicyString`
: A`CFString` containing policy-specific data.

  For the SMIME policy, this string contains an email address. For the SSL policy, it contains a host name.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsKeyUsage`
: A `CFNumber` containing an `SInt32` value specifying the operations for which the encryption key in this certificate can be used.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsAllowedError`
: A `CFNumber` containing an `SInt32` value indicating a `CSSM_RETURN` result code which, if encountered during certificate verification, is ignored for that certificate.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsResult`
: A `CFNumber` containing an `SInt32` value indicating the effective trust setting for this usage constraints dictionary.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

Trust Settings Result Constants

Effective trust settings for usage constraints dictionaries used by the `SecTrustSettingsCopyTrustSettings` (page 50) and `SecTrustSettingsSetTrustSettings` (page 54) functions.

```
enum {
    kSecTrustSettingsResultInvalid = 0,
    kSecTrustSettingsResultTrustRoot,
    kSecTrustSettingsResultTrustAsRoot,
    kSecTrustSettingsResultDeny,
    kSecTrustSettingsResultUnspecified
};
typedef uint32 SecTrustSettingsResult;
```

Constants

`kSecTrustSettingsResultInvalid`
: Never valid in a trust settings array or in an API call.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsResultTrustRoot`
: This root certificate is explicitly trusted.

  If the certificate is not a root (self-signed) certificate, the usage constraints dictionary is invalid.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsResultTrustAsRoot`
: This non-root certificate is explicitly trusted as if it were a trusted root.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsResultDeny`
: This certificate is explicitly distrusted.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

`kSecTrustSettingsResultUnspecified`
: This certificate is neither trusted nor distrusted. This value can be used to specify an "allowed error" without assigning trust to a specific certificate.

  This value can be used to specify an allowed error without assigning trust to the certificate.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

Default Root Certificate Trust Settings

A value indicating the default root certificate trust settings when used for a `SecCertificateRef` object in a trust settings API function.

```
#define kSecTrustSettingsDefaultRootCertSetting        ((SecCertificateRef)-1)
```

Constants

`kSecTrustSettingsDefaultRootCertSetting`
: Default trust settings for root certificates.

  Available in Mac OS X v10.5 and later.

  Declared in `SecTrustSettings.h`.

Discussion

Use this value with the `SecTrustSettingsSetTrustSettings` (page 54) function to set the default trust settings for root certificates. When evaluating trust settings for a root certificate in a given domain, if no matching explicit trust settings exist for that certificate, then the default value for the effective trust setting is returned (assuming that a default has been set and that the result is not `kSecTrustSettingsResultUnspecified`).

Result Codes

The most common result codes returned by Certificate, Key, and Trust Services are listed in the table below. The assigned error space is discontinuous: –25240..–25279 and –25290..–25329.

Result Code	Value	Description
`errSecNotAvailable`	–25291	No keychain is available. Available in Mac OS X v10.2 and later.
`errSecReadOnly`	–25292	A read-only error occurred. Available in Mac OS X v10.2 and later.
`errSecAuthFailed`	–25293	Authorization or authentication failed. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
`errSecNoSuchKeychain`	–25294	The keychain does not exist. Available in Mac OS X v10.2 and later.
`errSecInvalidKeychain`	–25295	The keychain is not valid. Available in Mac OS X v10.2 and later.
`errSecDuplicateKeychain`	–25296	A keychain with the same name already exists. Available in Mac OS X v10.2 and later.
`errSecDuplicateItem`	–25299	An item with the same primary key attributes already exists. Available in Mac OS X v10.2 and later.
`errSecItemNotFound`	–25300	The item cannot be found. Available in Mac OS X v10.2 and later.
`errSecBufferTooSmall`	–25301	The buffer is too small. Available in Mac OS X v10.2 and later.
`errSecDataTooLarge`	–25302	The data is too large for the particular data type. Available in Mac OS X v10.2 and later.
`errSecNoSuchAttr`	–25303	The attribute does not exist. Available in Mac OS X v10.2 and later.
`errSecInvalidItemRef`	–25304	The item object is invalid. Available in Mac OS X v10.2 and later.
`errSecInvalidSearchRef`	–25305	The search object is invalid. Available in Mac OS X v10.2 and later.
`errSecNoSuchClass`	–25306	The specified item does not appear to be a valid keychain item. Available in Mac OS X v10.2 and later.
`errSecNoDefaultKeychain`	–25307	A default keychain does not exist. Available in Mac OS X v10.2 and later.
`errSecInteractionNotAllowed`	–25308	Interaction with the user is required in order to grant access or process a request; however, user interaction with the Security Server has been disabled by the program. Available in Mac OS X v10.2 and later.
`errSecReadOnlyAttr`	–25309	The attribute is read-only. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
`errSecWrongSecVersion`	–25310	The version is incorrect. Available in Mac OS X v10.2 and later.
`errSecKeySizeNotAllowed`	–25311	The key size is not allowed. Available in Mac OS X v10.2 and later.
`errSecNoStorageModule`	–25312	No storage module is available. Available in Mac OS X v10.2 and later.
`errSecNoCertificateModule`	–25313	No certificate module is available. Available in Mac OS X v10.2 and later.
`errSecNoPolicyModule`	–25314	No policy module is available. Available in Mac OS X v10.2 and later.
`errSecInteractionRequired`	–25315	Interaction with the user is required in order to grant access or process a request; however, user interaction with the Security Server is impossible because the program is operating in a session incapable of graphics (such as a root session or ssh session). Available in Mac OS X v10.2 and later.
`errSecDataNotAvailable`	–25316	The data is not available. Available in Mac OS X v10.2 and later.
`errSecDataNotModifiable`	–25317	The data is not modifiable. Available in Mac OS X v10.2 and later.
`errSecCreateChainFailed`	–25318	One or more certificates required in order to validate this certificate cannot be found. Available in Mac OS X v10.2 and later.
`errSecInvalidPrefsDomain`	–25319	The preference domain specified is invalid. This error can occur in Mac OS X v10.3 and later. Available in Mac OS X v10.3 and later.
`errSecACLNotSimple`	–25240	The access control list is not in standard simple form. Available in Mac OS X v10.2 and later.
`errSecPolicyNotFound`	–25241	The policy specified cannot be found. Available in Mac OS X v10.2 and later.
`errSecInvalidTrustSetting`	–25242	The trust setting is invalid. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
`errSecNoAccessForItem`	–25243	The specified item has no access control. Available in Mac OS X v10.2 and later.
`errSecInvalidOwnerEdit`	–25244	An invalid attempt has been made to change the owner of an item. Available in Mac OS X v10.2 and later.
`errSecTrustNotAvailable`	–25245	No trust results are available. Available in Mac OS X v10.3 and later.

Deprecated Certificate, Key, and Trust Services Functions

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

Deprecated in Mac OS X v10.5

SecTrustGetCSSMAnchorCertificates

Retrieves the CSSM anchor certificates. (Deprecated in Mac OS X v10.5.)

```
OSStatus SecTrustGetCSSMAnchorCertificates (
   const CSSM_DATA **cssmAnchors,
   uint32 *cssmAnchorCount
);
```

Parameters

cssmAnchors

On return, points to an array of anchor certificates. This array is allocated by the system; you should not deallocate it. This data is not guaranteed to remain valid indefinitely; you should retrieve the data immediately and either pass it to other functions or copy it for future use.

cssmAnchorCount

On return, points to the number of `CSSM_DATA` structures returned in the `cssmAnchors` parameter.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

This function returns the certificates in the system's store of anchor certificates (see `SecTrustSetAnchorCertificates` (page 46). You can use the `CSSM_DATA` structures returned by this function as input to functions in the CSSM API. If you want references to the anchor certificates in a form appropriate for calls to the Certificate, Key, and Trust API, use the `SecTrustCopyAnchorCertificates` (page 38) function instead.

Availability

Available in Mac OS X v10.2 and later.

Deprecated in Mac OS X v10.5.

Declared In

`SecTrust.h`

SecTrustGetUserTrust

Retrieves the user-specified trust setting for a certificate and policy. (Deprecated in Mac OS X v10.5.)

```
OSStatus SecTrustGetUserTrust (
   SecCertificateRef certificate,
   SecPolicyRef policy,
   SecTrustUserSetting *trustSetting
);
```

Parameters

`certificate`
: The certificate object from which to obtain the user-specified trust setting.

`policy`
: The policy object for the policy for which to obtain the user-specified trust setting. Use the `SecPolicySearchCopyNext` (page 36) function to obtain a policy object.

`trustSetting`
: On return, points to the user-specified trust setting for the specified certificate and policy.

Return Value

A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion

Each certificate has one user-specified trust setting per policy. For each policy, the user can specify that the certificate is always to be trusted, is never to be trusted, or can be trusted only after permission is requested from—and granted by—the user. It is also possible for there to be no user-specified trust setting for a policy. See `SecTrustEvaluate` (page 41) for a discussion of the use of user-specified trust settings in a trust evaluation.

The `SecTrustGetUserTrust` function returns the effective user trust setting for the certificate and policy specified. You can obtain a certificate from a keychain and typecast the keychain item object (data type `SecKeychainItemRef`) to a certificate object (`SecCertificateRef`).

See "Trust Result Type Constants" (page 62) for values and descriptions of the user-specified trust settings. The user can set these values in the Keychain Access utility. If you provide your own UI for these settings, you can use the `SecTrustSetUserTrust` (page 74) function to set them.

Availability

Available in Mac OS X v10.2 and later.

Deprecated in Mac OS X v10.5.

Declared In

`SecTrust.h`

SecTrustSetUserTrust

Sets the user-specified trust settings of a certificate and policy. (Deprecated in Mac OS X v10.5.)

```
OSStatus SecTrustSetUserTrust (
   SecCertificateRef certificate,
   SecPolicyRef policy,
   SecTrustUserSetting trustSetting
);
```

Parameters

`certificate`
: The certificate object for which to set the user-specified trust settings. Use the `SecCertificateCreateFromData` (page 16) function to obtain a certificate object.

policy
: The policy object for the policy for which to set the user-specified trust settings. Use the `SecPolicySearchCopyNext` (page 36) function to obtain a policy object.

trustSetting
: The user-specified trust setting to be set. See "Trust Result Type Constants" (page 62) for possible values.

Return Value
A result code. See "Certificate, Key, and Trust Services Result Codes" (page 69).

Discussion
Each certificate has one user-specified trust setting per policy. These trust settings are used by the `SecTrustEvaluate` (page 41) function when evaluating trust. See "Trust Result Type Constants" (page 62) for values and descriptions of the user-specified trust settings. The user can set these values in the Keychain Access utility. Under certain circumstances, it might be appropriate for an administrative application to change a user trust setting. In that case, you can use the `SecTrustSetUserTrust` function to do so. You can obtain a certificate from a keychain and typecast the keychain item object (data type `SecKeychainItemRef`) to a certificate object (`SecCertificateRef`).

When you call the `SecTrustSetUserTrust` function, the user might be prompted to confirm the new setting before it is changed.

You can use the `SecTrustGetUserTrust` (page 73) function to get the current user-specified trust settings for a certificate.

Availability
Available in Mac OS X v10.2 and later.
Deprecated in Mac OS X v10.5.

Declared In
`SecTrust.h`

AppleX509TP Trust Policies

The trust policies implemented by the AppleX509TP CDSA module are shown in Table B-1. A pointer to a policy-specific option structure is placed in `CSSM_FIELD.FieldValue.Data`; this field is one of the fields in `CSSM_TP_CALLERAUTH_CONTEXT.Policy.PolicyIds` array.

Table B-1 AppleX509TP trust policies

Policy	OID	Options	Description
Apple Basic	`CSSMOID_APPLE_-X509_BASIC`	None	Basic X509-style certificate evaluation
SSL	`CSSMOID_APPLE_TP_SSL`	`CSSM_APPLE_TP_-SSL_OPTIONS`	Basic X509 plus host name verification per RFC 2818
SMIME	`CSSMOID_APPLE_-TP_SMIME`	`CSSM_APPLE_TP_-SMIME_OPTIONS`	Basic X509 plus email address verification and KeyUsage enforcement per RFC 2632
Extensible Authentication Protocol (EAP)	`CSSMOID_APPLE_TP_EAP`	`CSSM_APPLE_TP_-SSL_OPTIONS`	Functionally identical to SSL policy. A separate OID is provided to facilitate per-policy, per-certificate trust settings using the SecTrust mechanism
CRL Revocation	`CSSMOID_APPLE_TP_-REVOCATION_CRL`	`CSSM_APPLE_TP_-CRL_OPTIONS` (see option flags below)	Revocation using certificate revocation lists

CRL Policy Options

Certificate revocation list policy options are defined in the following structure:

```
typedef uint32 CSSM_APPLE_TP_CRL_OPT_FLAGS;
enum {
    // require CRL verification for each cert; default is "try"
    CSSM_TP_ACTION_REQUIRE_CRL_PER_CERT = 0x00000001,
    // enable fetch from network
    CSSM_TP_ACTION_FETCH_CRL_FROM_NET  = 0x00000002
};
typedef struct {
    // CSSM_APPLE_TP_CRL_OPTS_VERSION
    uint32                          Version
    CSSM_APPLE_TP_CRL_OPT_FLAGS         CrlFlags;
    /*
    * When non-NULL, store CRLs fetched from net here.
    * This is most likely a pointer to one of the
    * CSSM_TP_CALLERAUTH_CONTEXT.DBList entries but that
    * is not a strict requirement.
    */
    CSSM_DL_DB_HANDLE_PTR              crlStore;
} CSSM_APPLE_TP_CRL_OPTIONS;
```

When the `CSSM_TP_ACTION_REQUIRE_CRL_PER_CERT` flag is set, a certificate is not valid unless every certificate in the certificate chain has been successfully verified using a certificate revocation list. This check is in addition to any other certificate-specific or policy-specific checks required for validation. When this flag is not set, CRLs are evaluated if they are available, but it is not an error if the trust policy module cannot find a CRL. In either case, the certificate is not considered valid if a CRL is found that indicates that any certificate in the certificate chain has been revoked.

Evaluating Multiple Policies

If multiple policies are specified, they are evaluated sequentially. In this case, the `VerificationAbortOn` field of the `CSSM_TP_CALLERAUTH_CONTEXT` structure specifies when to abort the verification process (see the "Trust Policy Services API" chapter in *Common Security: CDSA and CSSM, version 2 (with corrigenda)* from The Open Group (http://www.opengroup.org/security/cdsa.htm)). The AppleX509TP CDSA module supports the following values for the `VerificationAbortOn` field:

- `CSSM_TP_STOP_ON_NONE`; continue to subsequent policies if one policy evaluation fails
- `CSSM_TP_STOP_ON_FIRST_FAIL`; stop immediately if a policy evaluation fails
- `CSSM_TP_STOP_ON_POLICY`; treated as the same as `CSSM_TP_STOP_ON_FIRST_FAIL`

Document Revision History

This table describes the changes to *Certificate, Key, and Trust Services Reference*.

Date	Notes
2008-11-19	Updated to Mac OS X v 10.5.
	Trust settings functions and constants are new to this revision.
2008-06-25	Added constants and functions to sign documents, evaluate signatures, encrypt and decrypt data, return information about certificates, and import identities.
2005-03-03	Corrected the parameter list and description of SecKeyCreatePair.
	Corrected the parameter list and description of SecKeyCreatePair.
2004-05-27	Moved trusted applications functions to *Keychain Services Reference*.
2004-04-29	Corrected and expanded descriptions of functions.
	Added appendix listing AppleX509TP trust policies.
2003-05-15	Preliminary version of this document.

Index