Cryptographic Message Syntax Services Reference

Security > Cryptography

2007-10-31

Contents

Cryptographic Message Syntax Services Reference

Framework: Security.framework

Declared in   CMSDecoder.h
                         CMSEncoder.h

Overview

Cryptographic Message Syntax Services is an API that implements Cryptographic Message Syntax (CMS) digital signatures and encryption for S/MIME messages.

A CMS message can be signed, encrypted, or both. A message can be signed by an arbitrary number of signers and can be encrypted for an arbitrary number of recipients. In CMS terminology, this module performs encryption using the enveloped-data content type and signing using the signed-data content type.

If the message is both signed and encrypted, it uses *nested ContentInfo types* in CMS terminology. In this case, the enveloped data content contains the signed data content; that is, the message is first signed and then the signed content is encrypted.

Cryptographic Message Syntax is defined in RFC 3852: *Cryptographic Message Syntax (CMS)* (http://www.ietf.org/rfc/rfc3852.txt).

S/MIME is defined in RFC 3851: *Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification* (http://www.ietf.org/rfc/rfc3851.txt)

Functions by Task

Initializing The Encoder

CMSEncoderCreate (page 26)
     Creates a CMSEncoder reference.

CMSEncoderGetTypeID (page 28)
     Returns the type identifier for the CMSEncoder opaque type.

Specifying Message Characteristics

CMSEncoderAddSigners (page 21)
     Specifies signers of the message.

Encoding The Message

Initializing The Decoder

Decoding The Message

`CMSDecoderUpdateMessage` (page 18)

Feeds raw bytes of the message to be decoded into the decoder.

`CMSDecoderFinalizeMessage` (page 14)

Indicates that there is no more data to decode.

`CMSDecoderSetDetachedContent` (page 16)

Specifies the message's detached content, if any.

`CMSDecoderCopyDetachedContent` (page 9)

Obtains the detached content specified with the `CMSDecoderSetDetachedContent` function.

Verifying The Signature

`CMSDecoderSetSearchKeychain` (page 17)

Specifies the keychains to search for intermediate certificates to be used in verifying a signed message's signer certificates.

`CMSDecoderGetNumSigners` (page 15)

Obtains the number of signers of a message.

`CMSDecoderCopySignerStatus` (page 12)

Obtains the status of a CMS message's signature.

`CMSDecoderCopySignerEmailAddress` (page 11)

Obtains the email address of the specified signer of a CMS message.

`CMSDecoderCopySignerCert` (page 10)

Obtains the certificate of the specified signer of a CMS message.

Obtaining Message Content

`CMSDecoderIsContentEncrypted` (page 16)

Determines whether a CMS message was encrypted.

`CMSDecoderCopyEncapsulatedContentType` (page 10)

Obtains the object identifier for the encapsulated data of a signed message.

`CMSDecoderCopyAllCerts` (page 7)

Obtain an array of all of the certificates in a message.

`CMSDecoderCopyContent` (page 8)

Obtain the message content, if any.

Functions

CMSDecoderCopyAllCerts

Obtain an array of all of the certificates in a message.

```
OSStatus CMSDecoderCopyAllCerts(
    CMSDecoderRef          cmsDecoder,
    CFArrayRef              *certsOut
);
```

Parameters

cmsDecoder

>   The CMSDecoder reference returned by the `CMSDecoderCreate` function.

certsOut

>   On return, points to an array of `SecCertificateRef` objects. Returns `NULL` if the message does not
>   contain any certificates (the message was encrypted but not signed); this is not considered an error.
>   You must use the `CFRelease` function to free this reference when you are finished using it.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

A CMS message can contain arbitrary sets of certificates other than or in addition to those indicating the
identity of signers. You can use this function to retrieve such certificates from a message. If the message was
signed, it contains signer certificates. You can use the `CMSDecoderGetNumSigners` and
`CMSDecoderCopySignerCert` functions to retrieve the certificates for a specific signer.

You cannot call this function until after you have called the `CMSDecoderFinalizeMessage` function.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSDecoderCreate` (page 14)

`CMSDecoderFinalizeMessage` (page 14)

`CMSEncoderAddSupportingCerts` (page 22)

`CMSDecoderGetNumSigners` (page 15)

`CMSDecoderCopySignerCert` (page 10)

Declared In

`CMSDecoder.h`

CMSDecoderCopyContent

Obtain the message content, if any.

```
OSStatus CMSDecoderCopyContent(
    CMSDecoderRef          cmsDecoder,
    CFDataRef              *contentOut
);
```

Parameters

cmsDecoder

>   The CMSDecoder reference returned by the `CMSDecoderCreate` function.

contentOut

>   On return, points to the message's content. Returns `NULL` if the content is detached. You must use
>   the `CFRelease` function to free this reference when you are finished using it.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion
If the message has detached content, you are responsible for retrieving the content. In that case, you use the `CMSDecoderSetDetachedContent` function to tell the decoder the location of the content.

You cannot call this function until after you have called the `CMSDecoderFinalizeMessage` function.

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

See Also
`CMSDecoderCreate` (page 14)
`CMSDecoderSetDetachedContent` (page 16)
`CMSEncoderSetHasDetachedContent` (page 29)
`CMSDecoderFinalizeMessage` (page 14)

Declared In
`CMSDecoder.h`

CMSDecoderCopyDetachedContent

Obtains the detached content specified with the `CMSDecoderSetDetachedContent` function.

```
OSStatus CMSDecoderCopyDetachedContent(
    CMSDecoderRef          cmsDecoder,
    CFDataRef              *detachedContentOut
);
```

Parameters
cmsDecoder
> The CMSDecoder reference returned by the `CMSDecoderCreate` function.

detachedContentOut
> On return, points to the data reference specified by an earlier call to the `CMSDecoderSetDetachedContent` function. Returns a NULL data reference if no detached content has been specified. You must use the `CFRelease` function to free this reference when you are finished using it.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35).

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

See Also
`CMSDecoderCreate` (page 14)
`CMSDecoderSetDetachedContent` (page 16)

Declared In
`CMSDecoder.h`

CMSDecoderCopyEncapsulatedContentType

Obtains the object identifier for the encapsulated data of a signed message.

```
OSStatus CMSDecoderCopyEncapsulatedContentType(
    CMSDecoderRef        cmsDecoder,
    CFDataRef           *eContentTypeOut
);
```

Parameters

cmsDecoder

> The CMSDecoder reference returned by the `CMSDecoderCreate` function.

eContentTypeOut

> On return, the object identifier for the encapsulated data in a signed message. Returns `NULL` if the message was not signed.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

In a signed message, the signed data consists of any type of content (referred to as the *encapsulated content*, because it is encapsulated in the signed data) plus the signature values. The content type of the encapsulated data is indicated by an object identifier. The default value for the OID is `id-data`, which indicates MIME-encoded content.

You cannot call this function until after you have called the `CMSDecoderFinalizeMessage` function.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSDecoderCreate` (page 14)

`CMSDecoderFinalizeMessage` (page 14)

`CMSEncoderSetEncapsulatedContentType` (page 29)

Declared In

CMSDecoder.h

CMSDecoderCopySignerCert

Obtains the certificate of the specified signer of a CMS message.

```
OSStatus CMSDecoderCopySignerCert(
    CMSDecoderRef        cmsDecoder,
    size_t               signerIndex,
    SecCertificateRef   *signerCertOut
);
```

Parameters

cmsDecoder

> The CMSDecoder reference returned by the `CMSDecoderCreate` function.

signerIndex

> A number indicating which signer's email address to return. Signer index numbers start with 0. Use the `CMSDecoderGetNumSigners` function to determine the total number of signers for a message.

signerCertOut
>   On return, points to the certificate of the specified signer.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35). Returns `paramErr` (–50) if the CMS message was not signed or if `signerIndex` is greater than the number of signers of the message minus one (signerIndex > (numSigners – 1)).

Discussion
You cannot call this function until after you have called the `CMSDecoderFinalizeMessage` function.

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

See Also
`CMSDecoderCreate` (page 14)
`CMSDecoderGetNumSigners` (page 15)
`CMSDecoderFinalizeMessage` (page 14)

Declared In
`CMSDecoder.h`

CMSDecoderCopySignerEmailAddress

Obtains the email address of the specified signer of a CMS message.

```
OSStatus CMSDecoderCopySignerEmailAddress(
    CMSDecoderRef        cmsDecoder,
    size_t               signerIndex,
    CFStringRef          *signerEmailAddressOut
);
```

Parameters
cmsDecoder
>   The CMSDecoder reference returned by the `CMSDecoderCreate` function.

signerIndex
>   A number indicating which signer's email address to return. Signer index numbers start with 0. Use the `CMSDecoderGetNumSigners` function to determine the total number of signers for a message.

signerEmailAddressOut
>   On return, points to the email address of the specified signer.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35). Returns `paramErr` (–50) if the CMS message was not signed or if `signerIndex` is greater than the number of signers of the message minus one (`signerIndex > (numSigners - 1)`).

Discussion
You cannot call this function until after you have called the `CMSDecoderFinalizeMessage` function.

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

See Also
`CMSDecoderCreate` (page 14)

CMSDecoderGetNumSigners (page 15)
CMSDecoderFinalizeMessage (page 14)

Declared In
CMSDecoder.h

CMSDecoderCopySignerStatus

Obtains the status of a CMS message's signature.

```
OSStatus CMSDecoderCopySignerStatus(
    CMSDecoderRef        cmsDecoder,
    size_t               signerIndex,
    CFTypeRef            policyOrArray,
    Boolean              evaluateSecTrust,
    CMSSignerStatus      *signerStatusOut,
    SecTrustRef          *secTrustOut,
    OSStatus             *certVerifyResultCodeOut
);
```

Parameters

cmsDecoder

The CMSDecoder reference returned by the CMSDecoderCreate function.

signerIndex

A number indicating which signer to examine. Signer index numbers start with 0. Use the CMSDecoderGetNumSigners function to determine the total number of signers for a message.

policyOrArray

The trust policy or policies to be used to verify the signer's certificate. You can specify either a single SecPolicyRef object or a CFArray of SecPolicyRef objects. Trust policies are discussed in Certificate, Key, and Trust Services Concepts in *Certificate, Key, and Trust Services Programming Guide*.

evaluateSecTrust

Set to TRUE to cause the decoder to call the SecTrustEvaluate function to evaluate the SecTrust object created for the evaluation of the signer certificate. Set to FALSE if you intend to call the SecTrustEvaluate function for the SecTrust object returned by the secTrustOut parameter.

signerStatusOut

If you specify TRUE for the evaluateSecTrust parameter, on return this parameter indicates the status of the signature. See "Signature Status Constants" (page 34) for possible results. Pass in NULL if you don't want a value returned.

secTrustOut

On return this parameter points to a SecTrust object. If you specified TRUE for the evaluateTrust parameter, this is the trust object that was used to verify the signer's certificate. If you specified FALSE for the evaluateTrust parameter, you can call the SecTrustEvaluate function to evaluate the SecTrust object. Pass NULL if you do not want this object returned. You must use the CFRelease function to free this reference when you are finished using it.

certVerifyResultCodeOut

> If you specify `TRUE` for the `evaluateSecTrust` parameter, on return this parameter indicates the result of the certificate verification. Pass in `NULL` if you don't want a value returned.
>
> Some of the most common results returned in this parameter include:

`CSSMERR_TP_INVALID_ANCHOR_CERT`

> The certificate was verified through the certificate chain to a self-signed root certificate that was present in the message, but that root certificate is not a known, trusted root certificate.

`CSSMERR_TP_NOT_TRUSTED`

> The certificate could not be verified back to a root certificate.

`CSSMERR_TP_VERIFICATION_FAILURE`

> The root certificate failed verification.

`CSSMERR_TP_VERIFY_ACTION_FAILED`

> Trust could not be established according to the specified trust policy.

`CSSMERR_TP_INVALID_CERTIFICATE`

> The signer's leaf certificate was not valid.

`CSSMERR_TP_CERT_EXPIRED`

> A certificate in the chain was expired at the time of verification.

`CSSMERR_TP_CERT_NOT_VALID_YET`

> A certificate in the chain was not yet valid at the time of verification.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35). A result of `noErr` indicates only that the function completed successfully; it does not indicate that the signature is verified or the certificates are valid. See the `signerStatusOut` and `certVerifyResultCodeOut` parameters for the verification and certificate validation results.

Discussion

You cannot call this function until after you have called the `CMSDecoderFinalizeMessage` function. Although the message has been fully decoded when the `CMSDecoderFinalizeMessage` function returns with no error, the signature can't be validated or certificates verified until this function is called.

A CMS message can be signed by multiple signers; this function returns the status associated with one signer as specified by the `signerIndex` parameter.

If you both pass in `FALSE` for the `evaluateSecTrust` parameter and `NULL` for the `secTrustOut` parameter, no evaluation of the signer certificate can occur.

Availability

Available in Mac OS X v10.5 and later.

See Also

CMSDecoderCreate (page 14)

CMSDecoderFinalizeMessage (page 14)

`SecTrustEvaluate`

Declared In

`CMSDecoder.h`

CMSDecoderCreate

Creates a CMSDecoder.

```
OSStatus CMSDecoderCreate(
    CMSDecoderRef        *cmsDecoderOut
);
```

Parameters

cmsDecoderOut

> On return, points to a CMSDecoder reference. You must use the `CFRelease` function to free this reference when you are finished using it.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

This is the first function in a sequence of decoder functions that you call to get information from a CMS message. The other functions in the sequence require you to pass in the CMSDecoder reference returned by this function. The next function in the sequence is `CMSDecoderUpdateMessage`.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSDecoderUpdateMessage` (page 18)

Declared In

`CMSDecoder.h`

CMSDecoderFinalizeMessage

Indicates that there is no more data to decode.

```
OSStatus CMSDecoderFinalizeMessage(      CMSDecoderRef         cmsDecoder
);
```

Parameters

cmsDecoder

> The CMSDecoder reference returned by the `CMSDecoderCreate` function.

Return Value

A result code. Returns `errSecUnknownFormat` upon detection of an improperly formatted CMS message. See "Cryptographic Message Services Result Codes" (page 35) for other result codes.

Discussion

When you call this function, the decoder finishes decoding the message. If the message was encrypted and this function returns a result code of `noErr`, the message was successfully decrypted. Call the `CMSDecoderCopyContent` function to retrieve the message content. Call the `CMSDecoderGetNumSigners` function to find out if the message was signed and, if so, how many signers there were.

Availability

Available in Mac OS X v10.5 and later.

See Also
CMSDecoderCreate (page 14)
CMSDecoderCopyContent (page 8)
CMSDecoderGetNumSigners (page 15)

Declared In
CMSDecoder.h

CMSDecoderGetNumSigners

Obtains the number of signers of a message.

```
OSStatus CMSDecoderGetNumSigners(
    CMSDecoderRef          cmsDecoder,
    size_t                 *numSignersOut
);
```

Parameters

cmsDecoder
> The CMSDecoder reference returned by the CMSDecoderCreate function.

numSignersOut
> On return, the number of signers of the message. Zero indicates that the message was not signed.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion
Call the CMSDecoderCopySignerStatus function to determine the status of a signature.

You cannot call this function until after you have called the CMSDecoderFinalizeMessage function.

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

See Also
CMSDecoderCreate (page 14)
CMSDecoderCopySignerStatus (page 12)
CMSDecoderFinalizeMessage (page 14)

Declared In
CMSDecoder.h

CMSDecoderGetTypeID

Returns the type identifier for the CMSDecoder opaque type.

```
CFTypeID CMSDecoderGetTypeID(void);
```

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

Declared In
CMSDecoder.h

CMSDecoderIsContentEncrypted

Determines whether a CMS message was encrypted.

```
OSStatus CMSDecoderIsContentEncrypted(
    CMSDecoderRef          cmsDecoder,
    Boolean                *isEncryptedOut
);
```

Parameters

cmsDecoder

> The CMSDecoder reference returned by the `CMSDecoderCreate` function.

isEncryptedOut

> Returns `TRUE` if the message was encrypted.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

Note that if the message was encrypted and the decoding succeeded (`CMSDecoderFinalizeMessage` returned `noErr`), then the message was successfully decrypted. Call `CMSDecoderCopyContent` to retrieve the decrypted content.

You cannot call this function until after you have called the `CMSDecoderFinalizeMessage` function.

Availability

Available in Mac OS X v10.5 and later.

See Also

CMSDecoderCreate (page 14)

CMSDecoderFinalizeMessage (page 14)

CMSDecoderCopyContent (page 8)

CMSEncoderAddRecipients (page 20)

Declared In

CMSDecoder.h

CMSDecoderSetDetachedContent

Specifies the message's detached content, if any.

```
OSStatus CMSDecoderSetDetachedContent(
    CMSDecoderRef          cmsDecoder,
    CFDataRef              detachedContent
);
```

Parameters

cmsDecoder

> The CMSDecoder reference returned by the `CMSDecoderCreate` function.

detachedContent

> A reference to the message's detached content.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

The data of a signed CMS message can optionally be sent separately from the message. If the message's content is detached from the message, you must call this function to tell the decoder where to find the message content.

Encrypted messages, including those that are also signed, cannot use detached content.

You can call this function either before or after decoding the message (by calling the `CMSDecoderUpdateMessage` and `CMSDecoderFinalizeMessage` functions). If a signed message has detached content, however, you must call this function before you can use the `CMSDecoderCopySignerStatus` function to ascertain the signature status.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSDecoderCreate` (page 14)

`CMSDecoderUpdateMessage` (page 18)

`CMSDecoderFinalizeMessage` (page 14)

`CMSDecoderCopySignerStatus` (page 12)

`CMSDecoderCopyDetachedContent` (page 9)

`CMSEncoderSetHasDetachedContent` (page 29)

Declared In

`CMSDecoder.h`

CMSDecoderSetSearchKeychain

Specifies the keychains to search for intermediate certificates to be used in verifying a signed message's signer certificates.

```
OSStatus CMSDecoderSetSearchKeychain(
    CMSDecoderRef          cmsDecoder,
    CFTypeRef              keychainOrArray
);
```

Parameters

cmsDecoder

   The CMSDecoder reference returned by the `CMSDecoderCreate` function.

keychainOrArray

   Either a single keychain to search, specified as a keychain object (type `SecKeychainRef`), or a set of keychains specified as a `CFArray` of keychain objects. If you specify an empty `CFArrayRef`, no keychains are searched for intermediate certificates.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

If you don't call this function, the decoder uses the default keychain search list to search for intermediate certificates.

If you do call this function, you must call it before you call the `CMSDecoderCopySignerStatus` function.

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

See Also
CMSDecoderCreate (page 14)
CMSDecoderCopySignerStatus (page 12)

Declared In
CMSDecoder.h

CMSDecoderUpdateMessage

Feeds raw bytes of the message to be decoded into the decoder.

```
OSStatus CMSDecoderUpdateMessage(
    CMSDecoderRef          cmsDecoder,
    const void            *msgBytes,
    size_t                 msgBytesLen
);
```

Parameters

cmsDecoder
> The CMSDecoder reference returned by the CMSDecoderCreate function.

msgBytes
> A pointer to the data to be decoded.

msgBytesLen
> The length of the data, in bytes.

Return Value
A result code. Returns errSecUnknownFormat upon detection of an improperly formatted CMS message. See "Cryptographic Message Services Result Codes" (page 35) for other result codes.

Discussion
This function can be called multiple times. Call the CMSDecoderFinalizeMessage function when you have no more data to decode.

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

See Also
CMSDecoderCreate (page 14)
CMSDecoderFinalizeMessage (page 14)

Declared In
CMSDecoder.h

CMSEncode

Encodes a message and obtains the result in one high-level function call.

```
OSStatus CMSEncode(
    CFTypeRef              signers,
    CFTypeRef              recipients,
    const CSSM_OID        *eContentType,
    Boolean                detachedContent,
    CMSSignedAttributes    signedAttributes,
    const void            *content,
    size_t                 contentLen,
    CFDataRef             *encodedContentOut
);
```

Parameters

signers

> The identity object for the identity of one signer, specified as type `SecIdentityRef`, or a `CFArray` of identity objects of type`SecIdentityRef`. Pass `NULL` for this parameter if you do not want to sign the message.

recipients

> A certificate containing a public encryption key for one message recipient, specified as a certificate object (type `SecCertificateRef`), or a `CFArray` of certificate objects. Pass NULL for this parameter if you do not want to encrypt the message.

eContentType

> The object identifier for the encapsulated data in a signed message. This parameter is optional. If you pass `0`, the value `id-data` is used. (This is the normal encapsulated content type for applications such as S/MIME, which uses it to indicate MIME-encoded content.) You can pass any value that is meaningful to your application.

detachedContent

> Specify `TRUE` if the signed data is to be separate from the message; that is, if the message is*not* to include the data to be signed. You cannot specify `TRUE` for this parameter for an encrypted message.

signedAttributes

> Attribute flags as defined in "Attributes For Signed Messages" (page 32). Attributes are optional for signed messages and are not used in other types of CMS messages. The use of attributes is described in section 2.5 of the S/MIME 3.1 specification.

content

> The content that you want to add to the message. The content must conform to the type set in the `CMSEncoderSetEncapsulatedContentType` parameter (or type `id-data` if that function has not been called).

contentLen

> The length of the content being added, in bytes.

encodedContentOut

> On return, points to the encoded message. You must use the `CFRelease` function to free this reference when you are finished using it.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

If you use this function, you must include content and you must provide valid non-`NULL` input for at least one of the `signers` and `recipients` parameters. You can both encrypt and sign a message; however, you cannot use detached content with an encrypted message. If you want to create a message that contains certificates and no other content, you must use the `CMSEncoderAddSupportingCerts` function instead of this one. To gain more control over the process of encoding a message, call the sequence of functions beginning with the `CMSEncoderCreate` function instead of this one.

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

See Also
CMSEncoderAddSupportingCerts (page 22)
CMSEncoderCreate (page 26)

Declared In
CMSEncoder.h

CMSEncoderAddRecipients

Specifies a message is to be encrypted and specifies the recipients of the message.

```
OSStatus CMSEncoderAddRecipients(
    CMSEncoderRef          cmsEncoder,
    CFTypeRef              recipientOrArray
);
```

Parameters

cmsEncoder

    The CMSEncoder reference returned by the CMSEncoderCreate function.

recipientOrArray

    Either a single certificate containing a public encryption key for one message recipient, specified as a certificate object (type SecCertificateRef), or a set of certificates specified as a CFArray of certificate objects.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion
Your keychain must contain a certificate that supports encryption for each recipient. You can call this function more than once for the same message.

You can both sign and encrypt the same message; however, you cannot call both this function and the CMSEncoderSetHasDetachedContent function for the same message.

If you do call this function, you must call it before the first call to the CMSEncoderUpdateContent function.

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

See Also
CMSEncoderCreate (page 26)
CMSEncoderSetHasDetachedContent (page 29)
CMSEncoderUpdateContent (page 30)
CMSEncoderCopyRecipients (page 24)
CMSDecoderIsContentEncrypted (page 16)

Declared In
CMSEncoder.h

CMSEncoderAddSignedAttributes

Specifies attributes for a signed message.

```
OSStatus CMSEncoderAddSignedAttributes(
    CMSEncoderRef          cmsEncoder,
    CMSSignedAttributes    signedAttributes
);
```

Parameters

cmsEncoder

      The CMSEncoder reference returned by the `CMSEncoderCreate` function.

signedAttributes

      Attribute flags as defined in "Attributes For Signed Messages" (page 32).

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

Attributes are optional for signed messages. They are not used for other types of CMS messages. The use of attributes is described in section 2.5 of the S/MIME 3.1 specification.

If you do call this function, you must call it before the first call to the `CMSEncoderUpdateContent` function.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSEncoderCreate` (page 26)

`CMSEncoderUpdateContent` (page 30)

Declared In

`CMSEncoder.h`

CMSEncoderAddSigners

Specifies signers of the message.

```
OSStatus CMSEncoderAddSigners(
    CMSEncoderRef          cmsEncoder,
    CFTypeRef              signerOrArray
);
```

Parameters

cmsEncoder

      The CMSEncoder reference returned by the `CMSEncoderCreate` function.

signerOrArray

      The identity object for the identity of one signer, specified as type `SecIdentityRef`, or a `CFArray` of identity objects of type `SecIdentityRef`.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

Call this function only if the message is to be signed. You can call this function more than once for the same message.

If you do call this function, you must call it before the first call to the `CMSEncoderUpdateContent` function.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSEncoderCreate` (page 26)
`CMSEncoderUpdateContent` (page 30)
`CMSEncoderCopySigners` (page 25)
`CMSDecoderCopySignerStatus` (page 12)

Declared In

`CMSEncoder.h`

CMSEncoderAddSupportingCerts

Adds certificates to a message.

```
OSStatus CMSEncoderAddSupportingCerts(
    CMSEncoderRef          cmsEncoder,
    CFTypeRef              certOrArray
);
```

Parameters

cmsEncoder

The CMSEncoder reference returned by the `CMSEncoderCreate` function.

certOrArray

Either a single certificate, specified as a certificate object (type `SecCertificateRef`), or a set of certificates specified as a `CFArray` of certificate objects.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

A CMS message can contain arbitrary sets of certificates other than or in addition to those indicating the identity of signers. You can use this function to add such certificates to a message. It is not necessary to call this function for a normal signed message. When you create a signed message, Cryptographic Message Services automatically adds the signer certificates and any intermediate certificates needed to verify the signers.

You can use this function even if you don't sign or encrypt the message, in order to transport one or more certificates. To do so, call `CMSEncoderCreate` to obtain a `CMSEncoderRef` reference, call `CMSEncoderAddSupportingCerts` one or more times, and then call `CMSEncoderCopyEncodedContent` to complete the message. No additional content need be specified.

If you do add content to the message in addition to the certificates, you must call this function before the first call to the `CMSEncoderUpdateContent` function.

Availability

Available in Mac OS X v10.5 and later.

See Also
CMSEncoderCreate (page 26)
CMSEncoderCopyEncodedContent (page 23)
CMSEncoderUpdateContent (page 30)
CMSEncoderCopySupportingCerts (page 25)
CMSDecoderCopyAllCerts (page 7)

Declared In
CMSEncoder.h

CMSEncoderCopyEncapsulatedContentType

Obtains the object identifier for the encapsulated data of a signed message.

```
OSStatus CMSEncoderCopyEncapsulatedContentType(
    CMSEncoderRef        cmsEncoder,
    CFDataRef            *eContentTypeOut
);
```

Parameters

cmsEncoder
> The CMSEncoder reference returned by the CMSEncoderCreate function.

eContentTypeOut
> On return, points to the object identifier for the encapsulated data in the signed message.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion
In a signed message, the signed data consists of any type of data (the *encapsulated content*) plus the signature values. This function returns the object identifier (OID) of the encapsulated content as it was specified with the CMSEncoderSetEncapsulatedContentType function.

If the CMSEncoderSetEncapsulatedContentType function has not been called for this message, this function returns a NULL pointer.

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

See Also
CMSEncoderCreate (page 26)
CMSEncoderSetEncapsulatedContentType (page 29)

Declared In
CMSEncoder.h

CMSEncoderCopyEncodedContent

Finishes encoding the message and obtains the encoded result.

```
OSStatus CMSEncoderCopyEncodedContent(
    CMSEncoderRef        cmsEncoder,
    CFDataRef            *encodedContentOut
);
```

Parameters

cmsEncoder

> The CMSEncoder reference returned by the `CMSEncoderCreate` function.

encodedContentOut

> On return, points to the encoded message. You must use the `CFRelease` function to free this reference when you are finished using it.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

This is the last function in the sequence of encoding functions you call when creating a signed or encrypted message. In many cases, you can call the `CMSEncode` function alone instead of using the sequence of encoding functions.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSEncoderCreate` (page 26)

`CMSEncode` (page 18)

Declared In

`CMSEncoder.h`

CMSEncoderCopyRecipients

Obtains the array of recipients specified with the `CMSEncoderAddRecipients` function.

```
OSStatus CMSEncoderCopyRecipients(
    CMSEncoderRef        cmsEncoder,
    CFArrayRef           *recipientsOut
);
```

Parameters

cmsEncoder

> The CMSEncoder reference returned by the `CMSEncoderCreate` function.

recipientsOut

> On return, points to an array of certificate objects of type `SecCertificateRef` of the recipients of the message. If the `CMSEncoderAddRecipients` function has not been called for this message, this function returns a `NULL` array. You must use the `CFRelease` function to free this reference when you are finished using it.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Availability

Available in Mac OS X v10.5 and later.

See Also

CMSEncoderCreate (page 26)

CMSEncoderAddRecipients (page 20)

Declared In

CMSEncoder.h

CMSEncoderCopySigners

Obtains the array of signers specified with the CMSEncoderAddSigners function.

```
OSStatus CMSEncoderCopySigners(
    CMSEncoderRef         cmsEncoder,
    CFArrayRef            *signersOut
);
```

Parameters

cmsEncoder

> The CMSEncoder reference returned by the CMSEncoderCreate function.

signersOut

> On return, points to an array of identity objects of type SecIdentityRef of the signers of the message.
> If the CMSEncoderAddSigners function has not been called for this message, this function returns
> a NULL array. You must use the CFRelease function to free this reference when you are finished
> using it.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Availability

Available in Mac OS X v10.5 and later.

See Also

CMSEncoderCreate (page 26)

CMSEncoderAddSigners (page 21)

Declared In

CMSEncoder.h

CMSEncoderCopySupportingCerts

Obtains the certificates added to a message with CMSEncoderAddSupportingCerts.

```
OSStatus CMSEncoderCopySupportingCerts(
    CMSEncoderRef         cmsEncoder,
    CFArrayRef            *certsOut
);
```

Parameters

cmsEncoder

> The CMSEncoder reference returned by the CMSEncoderCreate function.

certsOut

> On return, points to a CFArray of `SecCertificateRef` objects. You must use the `CFRelease` function to free this reference when you are finished using it.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

A CMS message can contain arbitrary sets of certificates other than or in addition to those indicating the identity of signers. You can use this function to obtain any such certificates added with the `CMSEncoderAddSupportingCerts` function. If `CMSEncoderAddSupportingCerts` has not been called, this function returns a `NULL` value for `certsOut`. Note that this function does not return the signing certificates, if any.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSEncoderCreate` (page 26)

`CMSEncoderAddSupportingCerts` (page 22)

Declared In

`CMSEncoder.h`

CMSEncoderCreate

Creates a CMSEncoder reference.

```
OSStatus CMSEncoderCreate(
    CMSEncoderRef        *cmsEncoderOut
);
```

Parameters

cmsEncoderOut

> On return, points to a CMSEncoder reference. You must use the `CFRelease` function to free this reference when you are finished using it.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

This is the first function in a sequence of encoder functions that you call to sign or encrypt a message. The other functions in the sequence require you to pass in the CMSEncoder reference returned by this function. In many cases, you can call the `CMSEncode` function alone instead of this sequence of encoder functions.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSEncode` (page 18)

Declared In

`CMSEncoder.h`

CMSEncoderGetCertificateChainMode

Obtains a constant that indicates which certificates are to be included in a signed CMS message.

```
OSStatus CMSEncoderGetCertificateChainMode(
    CMSEncoderRef              cmsEncoder,
    CMSCertificateChainMode    *chainModeOut
);
```

Parameters

cmsEncoder

The CMSEncoder reference returned by the `CMSEncoderCreate` function.

chainMode

On return, a constant that indicates which certificate or certificates are to be included in the message. See "Certificate Inclusion Constants" (page 33).

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSEncoderCreate` (page 26)

`CMSEncoderSetCertificateChainMode` (page 28)

Declared In

CMSEncoder.h

CMSEncoderGetHasDetachedContent

Indicates whether the message is to have detached content.

```
OSStatus CMSEncoderGetHasDetachedContent(
    CMSEncoderRef        cmsEncoder,
    Boolean              *detachedContentOut
);
```

Parameters

cmsEncoder

The CMSEncoder reference returned by the `CMSEncoderCreate` function.

detachedContentOut

Returns `TRUE` if the message has detached content.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

This function returns the value specified in `CMSEncoderSetHasDetachedContent` if that function has been called; otherwise it returns `FALSE`.

Availability

Available in Mac OS X v10.5 and later.

See Also
CMSDecoderCreate (page 14)
CMSEncoderSetHasDetachedContent (page 29)

Declared In
CMSEncoder.h

CMSEncoderGetTypeID

Returns the type identifier for the CMSEncoder opaque type.

```
CFTypeID CMSEncoderGetTypeID(void);
```

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

Declared In
CMSEncoder.h

CMSEncoderSetCertificateChainMode

Specifies which certificates to include in a signed CMS message.

```
OSStatus CMSEncoderSetCertificateChainMode(
    CMSEncoderRef              cmsEncoder,
    CMSCertificateChainMode    chainMode
);
```

Parameters
cmsEncoder
> The CMSEncoder reference returned by the CMSEncoderCreate function.

chainMode
> A constant that indicates which certificate or certificates to include in the message. See "Certificate Inclusion Constants" (page 33).

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion
This function is used only for signed messages and is optional. If you don't call this function, the default, kCMSCertificateChain, is used. In this case the message includes the signer certificate plus all certificates needed to verify the signer certificate, up to but not including the root certificate.

If you do call this function, you must call it before the first call to the CMSEncoderUpdateContent function.

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

See Also
CMSEncoderCreate (page 26)
CMSEncoderUpdateContent (page 30)
CMSEncoderGetCertificateChainMode (page 27)

Declared In
CMSEncoder.h

CMSEncoderSetEncapsulatedContentType

Specifies an object identifier for the encapsulated data of a signed message.

```
OSStatus CMSEncoderSetEncapsulatedContentType(
    CMSEncoderRef         cmsEncoder,
    const CSSM_OID       *eContentType
);
```

Parameters

cmsEncoder
        The CMSEncoder reference returned by the CMSEncoderCreate function.

eContentType
        The object identifier for the encapsulated data in a signed message.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion
In a signed message, the signed data consists of any type of content (referred to as the *encapsulated content*, because it is encapsulated in the signed data) plus the signature values. You can indicate the content type of the encapsulated data by specifying an object identifier (OID) in the eContentType parameter of this function. The default value for the OID (used if this function is not called) is id-data. This is the normal encapsulated content type for applications such as S/MIME, which uses it to indicate MIME-encoded content. You can pass any value that is meaningful to your application. Examples of CMS OIDs are listed in http://www.imc.org/ietf-smime/other-smime-oids.asn.

If you do call this function, you must call it before the first call to the CMSEncoderUpdateContent function.

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

See Also
CMSEncoderCreate (page 26)
CMSEncoderUpdateContent (page 30)
CMSEncoderCopyEncapsulatedContentType (page 23)
CMSDecoderCopyEncapsulatedContentType (page 10)

Declared In
CMSEncoder.h

CMSEncoderSetHasDetachedContent

Specifies whether the signed data is to be separate from the message.

```
OSStatus CMSEncoderSetHasDetachedContent(
    CMSEncoderRef        cmsEncoder,
    Boolean              detachedContent
);
```

Parameters

cmsEncoder

>   The CMSEncoder reference returned by the `CMSEncoderCreate` function.

detachedContent

>   Specify `TRUE` if the signed data is to be separate from the message; that is, if the message is *not* to include the data to be signed.

Return Value

A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion

A signed CMS message can optionally be sent separately from the signed data. The default, if this function is not called, is `detachedContent=FALSE`; that is, the message contains the data to be signed.

Encrypted messages, including those that are also signed, cannot use detached content.

If you do call this function, you must call it before the first call to the `CMSEncoderUpdateContent` function.

Availability

Available in Mac OS X v10.5 and later.

See Also

`CMSEncoderCreate` (page 26)

`CMSEncoderUpdateContent` (page 30)

`CMSEncoderGetHasDetachedContent` (page 27)

`CMSDecoderSetDetachedContent` (page 16)

Declared In

`CMSEncoder.h`

CMSEncoderUpdateContent

Feeds content bytes into the encoder.

```
OSStatus CMSEncoderUpdateContent(
    CMSEncoderRef        cmsEncoder,
    const void           *content,
    size_t               contentLen
);
```

Parameters

cmsEncoder

>   The CMSEncoder reference returned by the `CMSEncoderCreate` function.

content

>   The content that you want to add to the message. The content must conform to the type set with the `CMSEncoderSetEncapsulatedContentType` function (or type `id-data` if that function has not been called).

contentLen
>   The length of the content being added, in bytes.

Return Value
A result code. See "Cryptographic Message Services Result Codes" (page 35).

Discussion
You use this function to add the content that is to be signed or encrypted. If the message is only a container for certificates added with the `CMSEncoderAddSupportingCerts` function and has no other content, do not call this function. This function can be called multiple times.

After you are finished adding content, call the `CMSEncoderCopyEncodedContent` function to complete the message creation process.

None of the setter functions (`CMSEncoderSetHasDetachedContent`, `CMSEncoderSetEncapsulatedContentType`, or `CMSEncoderSetCertificateChainMode`) can be called after this function has been called.

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

See Also
`CMSEncoderCreate` (page 26)
`CMSEncoderSetEncapsulatedContentType` (page 29)
`CMSEncoderAddSupportingCerts` (page 22)
`CMSEncoderCopyEncodedContent` (page 23)

Declared In
`CMSEncoder.h`

Data Types

CMSEncoderRef

Opaque reference to a CMS encoder object.

```
typedef struct _CMSEncoder *CMSEncoderRef;
```

Discussion
This is object is compatible with Core Foundation and uses standard Core Foundation semantics. Dispose of it with the `CFRelease` function.

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

Declared In
`CMSEncoder.h`

CMSDecoderRef

Opaque reference to a CMS decoder object.

```
typedef struct _CMSDecoder *CMSDecoderRef;
```

Discussion
This is object is compatible with Core Foundation and uses standard Core Foundation semantics. Dispose of it with the `CFRelease` function.

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

Declared In
`CMSDecoder.h`

Constants

Attributes For Signed Messages

Optional attributes that can be specified with the `CMSEncoderAddSignedAttributes` (page 21) function.

```
enum {
    kCMSAttrNone                        = 0x0000,
    kCMSAttrSmimeCapabilities           = 0x0001,
    kCMSAttrSmimeEncryptionKeyPrefs     = 0x0002,
    kCMSAttrSmimeMSEncryptionKeyPrefs   = 0x0004,
    kCMSAttrSigningTime                 = 0x0008
};
typedef uint32_t CMSSignedAttributes;
```

Constants
`kCMSAttrNone`

No attributes.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

`kCMSAttrSmimeCapabilities`

Adds information to the signed message that identifies signature, encryption, and digest algorithms supported by the encoder. Adding this attribute does not change the way in which the message is encoded. See RFC 2311: *S/MIME Version 2 Message Specification* (http://www.ietf.org/rfc/rfc2311.txt) section 2.5.2 for more information about the capabilities attribute.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

`kCMSAttrSmimeEncryptionKeyPrefs`

Indicates that the signing certificate included with the message is the preferred one for S/MIME encryption.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

`kCMSAttrSmimeMSEncryptionKeyPrefs`

Same as `kCMSSmimeEncryptionKeyPrefs`, using an attribute object identifier (OID) preferred by Microsoft.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

`kCMSAttrSigningTime`

Causes the encoder to include the signing time.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

Discussion

These flags cause the encoder to add attributes to a signed message that can be interpreted by the recipient. These attributes are not used for unsigned messages.

Declared In

`CMSEncoder.h`

Certificate Inclusion Constants

Constants that can be set by the `CMSEncoderSetCertificateChainMode` (page 28) function to specify what certificates to include in a signed message.

```
enum {
    kCMSCertificateNone = 0,
    kCMSCertificateSignerOnly,
    kCMSCertificateChain,
    kCMSCertificateChainWithRoot
};
typedef uint32_t CMSCertificateChainMode;
```

Constants

`kCMSCertificateNone`

Don't include any certificates.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

`kCMSCertificateSignerOnly`

Only include signer certificates.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

`kCMSCertificateChain`

Include the signer certificate chain up to but not including the root certificate.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

`kCMSCertificateChainWithRoot`

Include the entire signer certificate chain, including the root certificate.

Available in Mac OS X v10.5 and later.

Declared in `CMSEncoder.h`.

Declared In
CMSEncoder.h

Signature Status Constants

Constants that indicate the status of the signature and signer information in a signed message, as obtained by the CMSDecoderCopySignerStatus (page 12) function.

```
enum {
    kCMSSignerUnsigned = 0,
    kCMSSignerValid,
    kCMSSignerNeedsDetachedContent,
    kCMSSignerInvalidSignature,
    kCMSSignerInvalidCert,
    kCMSSignerInvalidIndex
};
typedef uint32_t CMSSignerStatus;
```

Constants

kCMSSignerUnsigned

> The message was not signed.

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

> Declared in CMSDecoder.h.

kCMSSignerValid

> The message was signed and both the signature and the signer certificate have been verified.

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

> Declared in CMSDecoder.h.

kCMSSignerNeedsDetachedContent

> The message was signed but has detached content. You must call the CMSDecoderSetDetachedContent (page 16) function before ascertaining the signature status.

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

> Declared in CMSDecoder.h.

kCMSSignerInvalidSignature

> The message was signed but the signature is invalid.

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

> Declared in CMSDecoder.h.

kCMSSignerInvalidCert

> The message was signed but the signer's certificate could not be verified.

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

> Declared in CMSDecoder.h.

kCMSSignerInvalidIndex

> The specified value for the signer index (signerIndex parameter) is greater than the number of signers of the message minus one (signerIndex > (numSigners - 1)).

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

> Declared in CMSDecoder.h.

Declared In
CMSDecoder.h

Result Codes

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

Result Code	Value	Description
errSecNotAvailable	−25291	No trust results are available. Available in Mac OS X v10.2 and later.
errSecReadOnly	−25292	Read only error. Available in Mac OS X v10.2 and later.
errSecAuthFailed	−25293	Authorization/Authentication failed. Available in Mac OS X v10.2 and later.
errSecNoSuchKeychain	−25294	The keychain does not exist. Available in Mac OS X v10.2 and later.
errSecInvalidKeychain	−25295	The keychain is not valid. Available in Mac OS X v10.2 and later.
errSecDuplicateKeychain	−25296	A keychain with the same name already exists. Available in Mac OS X v10.2 and later.
errSecDuplicateCallback	−25297	More than one callback of the same name exists. Available in Mac OS X v10.2 and later.
errSecInvalidCallback	−25298	The callback is not valid. Available in Mac OS X v10.2 and later.
errSecDuplicateItem	−25299	The item already exists. Available in Mac OS X v10.2 and later.
errSecItemNotFound	−25300	The item cannot be found. Available in Mac OS X v10.2 and later.
errSecBufferTooSmall	−25301	The buffer is too small. Available in Mac OS X v10.2 and later.
errSecDataTooLarge	−25302	The data is too large for the particular data type. Available in Mac OS X v10.2 and later.
errSecNoSuchAttr	−25303	The attribute does not exist. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
errSecInvalidItemRef	–25304	The item reference is invalid. Available in Mac OS X v10.2 and later.
errSecInvalidSearchRef	–25305	The search reference is invalid. Available in Mac OS X v10.2 and later.
errSecNoSuchClass	–25306	The keychain item class does not exist. Available in Mac OS X v10.2 and later.
errSecNoDefaultKeychain	–25307	A default keychain does not exist. Available in Mac OS X v10.2 and later.
errSecInteractionNotAllowed	–25308	Interaction with the Security Server is not allowed. Available in Mac OS X v10.2 and later.
errSecReadOnlyAttr	–25309	The attribute is read only. Available in Mac OS X v10.2 and later.
errSecWrongSecVersion	–25310	The version is incorrect. Available in Mac OS X v10.2 and later.
errSecKeySizeNotAllowed	–25311	The key size is not allowed. Available in Mac OS X v10.2 and later.
errSecNoStorageModule	–25312	There is no storage module available. Available in Mac OS X v10.2 and later.
errSecNoCertificateModule	–25313	There is no certificate module available. Available in Mac OS X v10.2 and later.
errSecNoPolicyModule	–25314	There is no policy module available. Available in Mac OS X v10.2 and later.
errSecInteractionRequired	–25315	User interaction is required. Available in Mac OS X v10.2 and later.
errSecDataNotAvailable	–25316	The data is not available. Available in Mac OS X v10.2 and later.
errSecDataNotModifiable	–25317	The data is not modifiable. Available in Mac OS X v10.2 and later.
errSecCreateChainFailed	–25318	The attempt to create a certificate chain failed. Available in Mac OS X v10.2 and later.

Result Code	Value	Description
errSecInvalidPrefsDomain	−25319	The preference domain specified is invalid. This error is available in Mac OS X v10.3 and later. Available in Mac OS X v10.3 and later.
errSecACLNotSimple	−25240	The access control list is not in standard simple form. Available in Mac OS X v10.2 and later.
errSecPolicyNotFound	−25241	The policy specified cannot be found. Available in Mac OS X v10.2 and later.
errSecInvalidTrustSetting	−25242	The trust setting is invalid. Available in Mac OS X v10.2 and later.
errSecNoAccessForItem	−25243	The specified item has no access control. Available in Mac OS X v10.2 and later.
errSecInvalidOwnerEdit	−25244	An invalid attempt to change the owner of an item. Available in Mac OS X v10.2 and later.
errSecTrustNotAvailable	−25245	No trust results are available. Available in Mac OS X v10.3 and later.
errSecUnsupportedFormat	−25256	The specified import or export format is not supported. Available in Mac OS X v10.4 and later.
errSecUnknownFormat	−25257	The item you are trying to import has an unknown format. Available in Mac OS X v10.4 and later.
errSecKeyIsSensitive	−25258	The key must be wrapped to be exported. Available in Mac OS X v10.4 and later.
errSecMultiplePrivKeys	−25259	An attempt was made to import multiple private keys. Available in Mac OS X v10.4 and later.
errSecPassphraseRequired	−25260	A password is required for import or export. Available in Mac OS X v10.4 and later.

Document Revision History

This table describes the changes to *Cryptographic Message Syntax Services Reference*.

Date	Notes
2007-10-31	New document that describes the API for encoding and decoding signed and encrypted messages using Cryptographic Message Syntax (CMS).

Index