

[MailKit](#) / [MEMessageSecurityHandler](#)

Protocol

MEMessageSecurityHandler

An object that digitally signs or encrypts messages the user sends and receives.

macOS 12.0+

```
@MainActor  
protocol MEMessageSecurityHandler : MEMessageDecoder, MEMessageEncoder
```

Overview

When users enable an extension that implements a message security handler, Mail passes incoming and outgoing message content to the extension for encryption and digital signing.

To encompass the symmetrical halves for encoding and decoding, MailKit defines two protocols that [MEMessageSecurityHandler](#) conforms to:

[MEMessageEncoder](#)

Methods that encrypt and digitally sign an email message.

[MEMessageDecoder](#)

Methods that decrypt email messages and verify digital signatures.

As the user composes a mail message, MailKit calls [getEncodingStatus\(for:composeContext:completionHandler:\)](#) to determine if the handler can sign or encrypt the message. The handler indicates the capabilities by providing an instance of [MEOutgoingMessageEncodingStatus](#). Mail reflects this status in the compose window by enabling the appropriate buttons to let the user choose how to encode the message. When the user sends the message, MailKit invokes the [encode\(_:composeContext:completionHandler:\)](#) method, and indicates whether the user chose to encrypt or sign the message.

When MailKit needs the original message content, it invokes the handler's [decodedMessage\(forMessageData:\)](#) method. This method creates an instance of [MEDecodedMessage](#).

Message that includes the raw decoded message data and the details of who signed the message in an instance of MEMessageSecurityInformation.

Note

MailKit stores the encrypted and signed message content. Therefore, MailKit may ask a message security handler to decode the same message repeatedly over time when it needs the decoded original message content.

To indicate that your extension contains a message security handler, add `MEMessageSecurityHandler` to the `MEEExtensionCapabilities` array in the extension's `Info.plist` file:

```
<key>NSExtensionAttributes</key>
<dict>
    <key>MEEExtensionCapabilities</key>
    <array>
        <string>MEMessageSecurityHandler</string>
    </array>
</dict>
```

Topics

Encrypting and Signing Messages

`protocol MEMessageEncoder`

An object that encrypts or digitally signs outgoing messages.

`class MEEncodedOutgoingMessage`

An object that contains the signed or encrypted representation of a message's RFC 2822 data.

`class MEOutgoingMessageEncodingStatus`

An object that contains information about security measures the user can apply when composing a message.

`class MEMessageEncodingResult`

An object that contains a signed or encrypted message, or errors that indicate failure to encode the message.

Decrypting Messages and Verifying Signatures

```
protocol MEMessageDecoder
```

An object that decrypts messages and provides details about digital signatures.

```
class MEDecodedMessage
```

An object that contains the RFC 2822 data for a message, without encryption or digital signatures.

```
class MEMessageSigner
```

An object that contains details about the person who signed a message.

```
class MEMessageSecurityInformation
```

An object that contains details about a message's content, such as if it's encrypted and who digitally signed it.

Displaying Signature Details

```
func extensionViewController(signers: [MEMessageSigner]) -> MEEExtensionViewController?
```

Returns a view controller that displays details about a message's digital signature.

Required

Instance Methods

```
func extensionViewController(messageContext: Data) -> MEEExtensionViewController?
```

Required

```
func primaryActionClicked(forMessageContext: Data, completionHandler: (MEEExtensionViewController?) -> Void)
```

Required

Relationships

Inherits From

`MEMessageDecoder`, `MEMessageEncoder`, `NSObjectProtocol`