PowerShell Get-Help on command 'Protect-CmsMessage'

PS C:\Users\wahid> Get-Help Protect-CmsMessage

NAME

Protect-CmsMessage

SYNOPSIS

Encrypts content by using the Cryptographic Message Syntax format.

SYNTAX

```
Protect-CmsMessage [-To] <System.Management.Automation.CmsMessageRecipient[]>
[-Content] <System.Management.Automation.PSObject> [[-OutFile]
<System.String>] [<CommonParameters>]

Protect-CmsMessage [-To] <System.Management.Automation.CmsMessageRecipient[]>
[-LiteralPath] <System.String> [[-OutFile] <System.String>]
[<CommonParameters>]

Protect-CmsMessage [-To] <System.Management.Automation.CmsMessageRecipient[]>
[-Path] <System.String> [[-OutFile] <System.String>] [<CommonParameters>]
```

DESCRIPTION

The `Protect-CmsMessage` cmdlet encrypts content by using the Cryptographic Message Syntax (CMS) format.

The CMS cmdlets support encryption and decryption of content using the IETF format as documented by RFC5652 (https://tools.ietf.org/html/rfc5652.html).

The CMS encryption standard uses public key cryptography, where the keys used to encrypt content (the public key) and the keys used to decrypt content (the private key) are separate. Your public key can be shared widely, and is not sensitive data. If any content is encrypted with this public key, only your private key can decrypt it. For more information, see Public-key cryptography (https://en.wikipedia.org/wiki/Public-key_cryptography).

Before you can run the `Protect-CmsMessage` cmdlet, you must have an encryption certificate set up. To be recognized in PowerShell, encryption certificates require a unique extended key usage (EKU (/windows/desktop/SecCrypto/eku))ID to identify them as data encryption certificates (such as the IDs for Code Signing and Encrypted Mail). For an example of a certificate that would work for document encryption, see Example 1 in this topic.

PARAMETERS

-Content <System.Management.Automation.PSObject>

Specifies a PSObject that contains content that you want to encrypt. For example, you can encrypt the content of an event message, and then use the variable containing the message (`$Event`, in this example) as the value of the Content parameter: `$event = Get-WinEvent -ProviderName "PowerShell" -MaxEvents 1`. You can also use the `Get-Content` cmdlet to get the contents of a file, such as a Microsoft Word document, and save the content in a variable that you use as the value of the Content parameter.

-LiteralPath <System.String>

Specifies the path to content that you want to encrypt. Unlike Path , the value of LiteralPath is used exactly as it is typed. No characters are interpreted as wildcards. If the path includes escape characters, enclose it in single quotation marks. Single quotation marks tell PowerShell not to interpret any characters as escape sequences.

-OutFile <System.String>

Specifies the path and file name of a file to which you want to send the encrypted content.

-Path <System.String>

Specifies the path to content that you want to encrypt.

-To <System.Management.Automation.CmsMessageRecipient[]>

Specifies one or more CMS message recipients, identified in any of the following formats:

- An actual certificate (as retrieved from the certificate provider).

- Path to the file containing the certificate.

- Path to a directory containing the certificate.

- Thumbprint of the certificate (used to look in the certificate store).

- Subject name of the certificate (used to look in the certificate store).

<CommonParameters>

This cmdlet supports the common parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see about_CommonParameters (https:/go.microsoft.com/fwlink/?LinkID=113216).

---- Example 1: Create a certificate for encrypting content ----

```
# Create .INF file for certreq
{[Version]
Signature = "$Windows NT$"

[Strings]
szOID_ENHANCED_KEY_USAGE = "2.5.29.37"
szOID_DOCUMENT_ENCRYPTION = "1.3.6.1.4.1.311.80.1"

[NewRequest]
Subject = "cn=youralias@emailaddress.com"
MachineKeySet = false
KeyLength = 2048
KeySpec = AT_KEYEXCHANGE
HashAlgorithm = Sha1
Exportable = true
RequestType = Cert
KeyUsage = "CERT_KEY_ENCIPHERMENT_KEY_USAGE | CERT_DATA_ENCIPHERMENT_KEY_USAGE"
ValidityPeriod = "Years"
ValidityPeriodUnits = "1000"

[Extensions]
%szOID_ENHANCED_KEY_USAGE% = "{text}%szOID_DOCUMENT_ENCRYPTION%"
} | Out-File -FilePath DocumentEncryption.inf

# After you have created your certificate file, run the following command to
add
# the certificate file to the certificate store. Now you are ready to encrypt
and
# decrypt content with the next two examples.
certreq.exe -new DocumentEncryption.inf DocumentEncryption.cer
```

---------- Example 2: Encrypt a message sent by email ----------

```
$Protected = "Hello World" | Protect-CmsMessage -To
"*youralias@emailaddress.com*"
```

In the following example, you encrypt a message, "Hello World", by piping it to the `Protect-CmsMessage` cmdlet, and then save the encrypted message in a variable. The To parameter uses the value of the Subject line in the certificate.

------- Example 3: View document encryption certificates -------

```
PS C:\> cd Cert:\CurrentUser\My
PS Cert:\CurrentUser\My> Get-ChildItem -DocumentEncryptionCert
```

To view document encryption certificates in the certificate provider, you can add the DocumentEncryptionCert dynamic parameter of Get-ChildItem (../Microsoft.PowerShell.Management/Get-ChildItem.md), available only when the certificate provider is loaded.

REMARKS

To see the examples, type: "get-help Protect-CmsMessage -examples".

For more information, type: "get-help Protect-CmsMessage -detailed".

For technical information, type: "get-help Protect-CmsMessage -full".

For online help, type: "get-help Protect-CmsMessage -online"