Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'openssl-cms.1ossl' command

$ man openssl-cms.1ossl

OPENSSL-CMS(1ossl) OpenSSL OPENSSL-CMS(1ossl)

NAME

openssl-cms - CMS command

SYNOPSIS

openssl cms [-help]

General options:

[-in filename] [-out filename] [-config configfile]

Operation options:

[-encrypt] [-decrypt] [-sign] [-verify] [-resign] [-sign_receipt]

[-verify_receipt receipt] [-digest_create] [-digest_verify] [-compress]

[-uncompress] [-EncryptedData_encrypt] [-EncryptedData_decrypt]

[-data_create] [-data_out] [-cmsout]

File format options:

[-inform DER|PEM|SMIME] [-outform DER|PEM|SMIME] [-rctform

DER|PEM|SMIME] [-stream] [-indef] [-noindef] [-binary] [-crlfeol]

[-asciicrlf]

Keys and password options:

[-pwri_password password] [-secretkey key] [-secretkeyid id] [-inkey

filename|uri] [-passin arg] [-keyopt name:parameter] [-keyform

DER|PEM|P12|ENGINE] [-engine id] [-provider name] [-provider-path path]

[-propquery propq] [-rand files] [-writerand file]

Encryption options:

[-originator file] [-recip file] [recipient-cert ...] [-cipher] [-wrap

cipher] [-aes128-wrap] [-aes192-wrap] [-aes256-wrap] [-des3-wrap]

[-debug_decrypt]

Signing options:

[-md digest] [-signer file] [-certfile file] [-cades] [-nodetach]

[-nocerts] [-noattr] [-nosmimecap] [-receipt_request_all]

[-receipt_request_first] [-receipt_request_from emailaddress]

[-receipt_request_to emailaddress]

Verification options:

[-signer file] [-content filename] [-no_content_verify]

[-no_attr_verify] [-nosigs] [-noverify] [-nointern] [-cades]

[-verify_retcode] [-CAfile file] [-no-CAfile] [-CApath dir]

[-no-CApath] [-CAstore uri] [-no-CAstore]

Output options:

[-keyid] [-econtent_type type] [-text] [-certsout file] [-to addr]

[-from addr] [-subject subj]

Printing options:

[-noout] [-print] [-nameopt option] [-receipt_request_print]

Validation options:

[-allow_proxy_certs] [-attime timestamp] [-no_check_time]

[-check_ss_sig] [-crl_check] [-crl_check_all] [-explicit_policy]

[-extended_crl] [-ignore_critical] [-inhibit_any] [-inhibit_map]

[-partial_chain] [-policy arg] [-policy_check] [-policy_print]

[-purpose purpose] [-suiteB_128] [-suiteB_128_only] [-suiteB_192]

[-trusted_first] [-no_alt_chains] [-use_deltas] [-auth_level num]

[-verify_depth num] [-verify_email email] [-verify_hostname hostname]

[-verify_ip ip] [-verify_name name] [-x509_strict] [-issuer_checks]

DESCRIPTION

This command handles data in CMS format such as S/MIME v3.1 email messages. It can encrypt, decrypt, sign, verify, compress, uncompress, and print messages.

OPTIONS

There are a number of operation options that set the type of operation to be performed: encrypt, decrypt, sign, verify, resign, sign_receipt,

verify_receipt, digest_create, digest_verify, compress, uncompress, EncryptedData_encrypt, EncryptedData_decrypt, data_create, data_out, or cmsout. The relevance of the other options depends on the operation type and their meaning may vary according to it.

-help

Print out a usage message.

General options

-in filename

The input message to be encrypted or signed or the message to be decrypted or verified.

-out filename

The message text that has been decrypted or verified or the output MIME format message that has been signed or verified.

-config configfile

See "Configuration Option" in openssl(1).

Operation options

-encrypt

Encrypt data for the given recipient certificates. Input file is the message to be encrypted. The output file is the encrypted data in MIME format. The actual CMS type is EnvelopedData.

Note that no revocation check is done for the recipient cert, so if that key has been compromised, others may be able to decrypt the text.

-decrypt

Decrypt data using the supplied certificate and private key. Expects encrypted datain MIME format for the input file. The decrypted data is written to the output file.

-sign

Sign data using the supplied certificate and private key. Input file is the message to be signed. The signed data in MIME format is written to the output file.

-verify

Verify signed data. Expects a signed data on input and outputs the

signed data. Both clear text and opaque signing is supported.

-resign

Resign a message: take an existing message and one or more new signers.

-sign_receipt

Generate and output a signed receipt for the supplied message. The input message must contain a signed receipt request. Functionality is otherwise similar to the -sign operation.

-verify_receipt receipt

Verify a signed receipt in filename receipt. The input message must contain the original receipt request. Functionality is otherwise similar to the -verify operation.

-digest_create

Create a CMS DigestedData type.

-digest_verify

Verify a CMS DigestedData type and output the content.

-compress

Create a CMS CompressedData type. OpenSSL must be compiled with zlib support for this option to work, otherwise it will output an error.

-uncompress

Uncompress a CMS CompressedData type and output the content. OpenSSL must be compiled with zlib support for this option to work, otherwise it will output an error.

-EncryptedData_encrypt

Encrypt content using supplied symmetric key and algorithm using a CMS EncryptedData type and output the content.

-EncryptedData_decrypt

Decrypt content using supplied symmetric key and algorithm using a CMS EncryptedData type and output the content.

-data_create

Create a CMS Data type.

-data_out

Data type and output the content.

-cmsout

Takes an input message and writes out a PEM encoded CMS structure.

File format options

-inform DER|PEM|SMIME

The input format of the CMS structure (if one is being read); the default is SMIME. See openssl-format-options(1) for details.

-outform DER|PEM|SMIME

The output format of the CMS structure (if one is being written); the default is SMIME. See openssl-format-options(1) for details.

-rctform DER|PEM|SMIME

The signed receipt format for use with the -receipt_verify; the default is SMIME. See openssl-format-options(1) for details.

-stream, -indef

The -stream and -indef options are equivalent and enable streaming I/O for encoding operations. This permits single pass processing of data without the need to hold the entire contents in memory, potentially supporting very large files. Streaming is automatically set for S/MIME signing with detached data if the output format is SMIME it is currently off by default for all other operations.

-noindef

Disable streaming I/O where it would produce and indefinite length constructed encoding. This option currently has no effect. In future streaming will be enabled by default on all relevant operations and this option will disable it.

-binary

Normally the input message is converted to "canonical" format which is effectively using CR and LF as end of line: as required by the S/MIME specification. When this option is present no translation occurs. This is useful when handling binary data which may not be in MIME format.

-crlfeol

Normally the output file uses a single LF as end of line. When this

option is present CRLF is used instead.

-asciicrlf

When signing use ASCII CRLF format canonicalisation. This strips trailing whitespace from all lines, deletes trailing blank lines at EOF and sets the encapsulated content type. This option is normally used with detached content and an output signature format of DER. This option is not normally needed when verifying as it is enabled automatically if the encapsulated content format is detected.

Keys and password options

-pwri_password password

Specify password for recipient.

-secretkey key

Specify symmetric key to use. The key must be supplied in hex format and be consistent with the algorithm used. Supported by the -EncryptedData_encrypt -EncryptedData_decrypt, -encrypt and -decrypt options. When used with -encrypt or -decrypt the supplied key is used to wrap or unwrap the content encryption key using an AES key in the KEKRecipientInfo type.

-secretkeyid id

The key identifier for the supplied symmetric key for KEKRecipientInfo type. This option must be present if the -secretkey option is used with -encrypt. With -decrypt operations the id is used to locate the relevant key if it is not supplied then an attempt is used to decrypt any KEKRecipientInfo structures.

-inkey filename|uri

The private key to use when signing or decrypting. This must match the corresponding certificate. If this option is not specified then the private key must be included in the certificate file specified with the -recip or -signer file. When signing this option can be used multiple times to specify successive keys.

-passin arg

The private key password source. For more information about the format of arg see openssl-passphrase-options(1).

-keyopt name:parameter

For signing and encryption this option can be used multiple times to set customised parameters for the preceding key or certificate. It can currently be used to set RSA-PSS for signing, RSA-OAEP for encryption or to modify default parameters for ECDH.

-keyform DER|PEM|P12|ENGINE

The format of the private key file; unspecified by default. See openssl-format-options(1) for details.

-engine id

See "Engine Options" in openssl(1). This option is deprecated.

-provider name

-provider-path path

-propquery propq

See "Provider Options" in openssl(1), provider(7), and property(7).

-rand files, -writerand file

See "Random State Options" in openssl(1) for details.

Encryption and decryption options

-originator file

A certificate of the originator of the encrypted message. Necessary for decryption when Key Agreement is in use for a shared key.

-recip file

When decrypting a message this specifies the certificate of the recipient. The certificate must match one of the recipients of the message.

When encrypting a message this option may be used multiple times to specify each recipient. This form must be used if customised parameters are required (for example to specify RSA-OAEP). Only certificates carrying RSA, Diffie-Hellman or EC keys are supported by this option.

recipient-cert ...

This is an alternative to using the -recip option when encrypting a message. One or more certificate filennames may be given.

-cipher

The encryption algorithm to use. For example triple DES (168 bits) - -des3 or 256 bit AES - -aes256. Any standard algorithm name (as used by the EVP_get_cipherbyname() function) can also be used preceded by a dash, for example -aes-128-cbc. See openssl-enc(1) for a list of ciphers supported by your version of OpenSSL.

Currently the AES variants with GCM mode are the only supported AEAD algorithms.

If not specified triple DES is used. Only used with -encrypt and -EncryptedData_create commands.

-wrap cipher

Cipher algorithm to use for key wrap when encrypting the message using Key Agreement for key transport. The algorithm specified should be suitable for key wrap.

-aes128-wrap, -aes192-wrap, -aes256-wrap, -des3-wrap

Use AES128, AES192, AES256, or 3DES-EDE, respectively, to wrap key. Depending on the OpenSSL build options used, -des3-wrap may not be supported.

-debug_decrypt

This option sets the CMS_DEBUG_DECRYPT flag. This option should be used with caution: see the notes section below.

Signing options

-md digest

Digest algorithm to use when signing or resigning. If not present then the default digest algorithm for the signing key will be used (usually SHA1).

-signer file

A signing certificate. When signing or resigning a message, this option can be used multiple times if more than one signer is required.

-certfile file

Allows additional certificates to be specified. When signing these will be included with the message. When verifying these will be searched for the signers certificates. The input can be in PEM,

DER, or PKCS#12 format.

-cades

When used with -sign, add an ESS signingCertificate or ESS signingCertificateV2 signed-attribute to the SignerInfo, in order to make the signature comply with the requirements for a CAdES Basic Electronic Signature (CAdES-BES).

-nodetach

When signing a message use opaque signing: this form is more resistant to translation by mail relays but it cannot be read by mail agents that do not support S/MIME. Without this option cleartext signing with the MIME type multipart/signed is used.

-nocerts

When signing a message the signer's certificate is normally included with this option it is excluded. This will reduce the size of the signed message but the verifier must have a copy of the signers certificate available locally (passed using the -certfile option for example).

-noattr

Normally when a message is signed a set of attributes are included which include the signing time and supported symmetric algorithms. With this option they are not included.

-nosmimecap

Exclude the list of supported algorithms from signed attributes, other options such as signing time and content type are still included.

-receipt_request_all, -receipt_request_first

For -sign option include a signed receipt request. Indicate requests should be provided by all recipient or first tier recipients (those mailed directly and not from a mailing list). Ignored it -receipt_request_from is included.

-receipt_request_from emailaddress

For -sign option include a signed receipt request. Add an explicit email address where receipts should be supplied.

-receipt_request_to emailaddress

Add an explicit email address where signed receipts should be sent to. This option must but supplied if a signed receipt is requested.

Verification options

-signer file

If a message has been verified successfully then the signers certificate(s) will be written to this file if the verification was successful.

-content filename

This specifies a file containing the detached content for operations taking S/MIME input, such as the -verify command. This is only usable if the CMS structure is using the detached signature form where the content is not included. This option will override any content if the input format is S/MIME and it uses the multipart/signed MIME content type.

-no_content_verify

Do not verify signed content signatures.

-no_attr_verify

Do not verify signed attribute signatures.

-nosigs

Don't verify message signature.

-noverify

Do not verify the signers certificate of a signed message.

-nointern

When verifying a message normally certificates (if any) included in the message are searched for the signing certificate. With this option only the certificates specified in the -certfile option are used. The supplied certificates can still be used as untrusted CAs however.

-cades

When used with -verify, require and check signer certificate digest. See the NOTES section for more details.

-verify_retcode

Exit nonzero on verification failure.

-CAfile file, -no-CAfile, -CApath dir, -no-CApath, -CAstore uri, -no-CAstore

See "Trusted Certificate Options" in openssl-verification-options(1) for details.

Output options

-keyid

Use subject key identifier to identify certificates instead of issuer name and serial number. The supplied certificate must include a subject key identifier extension. Supported by -sign and -encrypt options.

-econtent_type type

Set the encapsulated content type to type if not supplied the Data type is used. The type argument can be any valid OID name in either text or numerical format.

-text

This option adds plain text (text/plain) MIME headers to the supplied message if encrypting or signing. If decrypting or verifying it strips off text headers: if the decrypted or verified message is not of MIME type text/plain then an error occurs.

-certsout file

Any certificates contained in the input message are written to file.

-to, -from, -subject

The relevant email headers. These are included outside the signed portion of a message so they may be included manually. If signing then many S/MIME mail clients check the signers certificate's email address matches that specified in the From: address.

Printing options

-noout

For the -cmsout operation do not output the parsed CMS structure. This is useful if the syntax of the CMS structure is being checked.

-print

For the -cmsout operation print out all fields of the CMS structure. This implies -noout. This is mainly useful for testing purposes.

-nameopt option

For the -cmsout operation when -print option is in use, specifies printing options for string fields. For most cases utf8 is reasonable value. See openssl-namedisplay-options(1) for details.

-receipt_request_print

For the -verify operation print out the contents of any signed receipt requests.

Validation options

-allow_proxy_certs, -attime, -no_check_time, -check_ss_sig, -crl_check, -crl_check_all, -explicit_policy, -extended_crl, -ignore_critical, -inhibit_any, -inhibit_map, -no_alt_chains, -partial_chain, -policy, -policy_check, -policy_print, -purpose, -suiteB_128, -suiteB_128_only, -suiteB_192, -trusted_first, -use_deltas, -auth_level, -verify_depth, -verify_email, -verify_hostname, -verify_ip, -verify_name, -x509_strict

-issuer_checks

Set various options of certificate chain verification. See "Verification Options" in openssl-verification-options(1) for details.

Any validation errors cause the command to exit.

NOTES

The MIME message must be sent without any blank lines between the headers and the output. Some mail programs will automatically add a blank line. Piping the mail directly to sendmail is one way to achieve the correct format.

The supplied message to be signed or encrypted must include the necessary MIME headers or many S/MIME clients won't display it properly (if at all). You can use the -text option to automatically add plain text headers.

A "signed and encrypted" message is one where a signed message is then encrypted. This can be produced by encrypting an already signed

message: see the examples section.

This version of the program only allows one signer per message but it will verify multiple signers on received messages. Some S/MIME clients choke if a message contains multiple signers. It is possible to sign messages "in parallel" by signing an already signed message.

The options -encrypt and -decrypt reflect common usage in S/MIME clients. Strictly speaking these process CMS enveloped data: CMS encrypted data is used for other purposes.

The -resign option uses an existing message digest when adding a new signer. This means that attributes must be present in at least one existing signer using the same message digest or this operation will fail.

The -stream and -indef options enable streaming I/O support. As a result the encoding is BER using indefinite length constructed encoding and no longer DER. Streaming is supported for the -encrypt operation and the -sign operation if the content is not detached.

Streaming is always used for the -sign operation with detached data but since the content is no longer part of the CMS structure the encoding remains DER.

If the -decrypt option is used without a recipient certificate then an attempt is made to locate the recipient by trying each potential recipient in turn using the supplied private key. To thwart the MMA attack (Bleichenbacher's attack on PKCS #1 v1.5 RSA padding) all recipients are tried whether they succeed or not and if no recipients match the message is "decrypted" using a random key which will typically output garbage. The -debug_decrypt option can be used to disable the MMA attack protection and return an error if no recipient can be found: this option should be used with caution. For a fuller description see CMS_decrypt(3)).

CADES BASIC ELECTRONIC SIGNATURE (CADES-BES)

A CAdES Basic Electronic Signature (CAdES-BES), as defined in the European Standard ETSI EN 319 122-1 V1.1.1, contains:

? The signed user data as defined in CMS (RFC 3852);

? Content-type of the EncapsulatedContentInfo value being signed;

? Message-digest of the eContent OCTET STRING within encapContentInfo being signed;

? An ESS signingCertificate or ESS signingCertificateV2 attribute, as defined in Enhanced Security Services (ESS), RFC 2634 and RFC 5035. An ESS signingCertificate attribute only allows for SHA-1 as digest algorithm. An ESS signingCertificateV2 attribute allows for any digest algorithm.

? The digital signature value computed on the user data and, when present, on the signed attributes.

NOTE that the -cades option applies to the -sign or -verify operations. With this option, the -verify operation also requires that the signingCertificate attribute is present and checks that the given identifiers match the verification trust chain built during the verification process.

EXIT CODES

0 The operation was completely successfully.

1 An error occurred parsing the command options.

2 One of the input files could not be read.

3 An error occurred creating the CMS file or when reading the MIME message.

4 An error occurred decrypting or verifying the message.

5 The message was verified correctly but an error occurred writing out the signers certificates.

COMPATIBILITY WITH PKCS#7 FORMAT

openssl-smime(1) can only process the older PKCS#7 format. openssl cms supports Cryptographic Message Syntax format. Use of some features will result in messages which cannot be processed by applications which only support the older format. These are detailed below.

The use of the -keyid option with -sign or -encrypt.

The -outform PEM option uses different headers.

The -compress option.

The -secretkey option when used with -encrypt.

The use of PSS with -sign.

The use of OAEP or non-RSA keys with -encrypt.

Additionally the -EncryptedData_create and -data_create type cannot be processed by the older openssl-smime(1) command.

EXAMPLES

Create a cleartext signed message:

```
openssl cms -sign -in message.txt -text -out mail.msg \
    -signer mycert.pem
```

Create an opaque signed message

```
openssl cms -sign -in message.txt -text -out mail.msg -nodetach \
    -signer mycert.pem
```

Create a signed message, include some additional certificates and read the private key from another file:

```
openssl cms -sign -in in.txt -text -out mail.msg \
    -signer mycert.pem -inkey mykey.pem -certfile mycerts.pem
```

Create a signed message with two signers, use key identifier:

```
openssl cms -sign -in message.txt -text -out mail.msg \
    -signer mycert.pem -signer othercert.pem -keyid
```

Send a signed message under Unix directly to sendmail, including headers:

```
openssl cms -sign -in in.txt -text -signer mycert.pem \
    -from steve@openssl.org -to someone@somewhere \
    -subject "Signed message" | sendmail someone@somewhere
```

Verify a message and extract the signer's certificate if successful:

```
openssl cms -verify -in mail.msg -signer user.pem -out signedtext.txt
```

Send encrypted mail using triple DES:

```
openssl cms -encrypt -in in.txt -from steve@openssl.org \
    -to someone@somewhere -subject "Encrypted message" \
    -des3 user.pem -out mail.msg
```

Sign and encrypt mail:

```
openssl cms -sign -in ml.txt -signer my.pem -text \
    | openssl cms -encrypt -out mail.msg \
    -from steve@openssl.org -to someone@somewhere \
```

-subject "Signed and Encrypted message" -des3 user.pem

Note: the encryption command does not include the -text option because the message being encrypted already has MIME headers.

Decrypt a message:

openssl cms -decrypt -in mail.msg -recip mycert.pem -inkey key.pem

The output from Netscape form signing is a PKCS#7 structure with the detached signature format. You can use this program to verify the signature by line wrapping the base64 encoded structure and surrounding it with:

-----BEGIN PKCS7-----

-----END PKCS7-----

and using the command,

openssl cms -verify -inform PEM -in signature.pem -content content.txt

alternatively you can base64 decode the signature and use

openssl cms -verify -inform DER -in signature.der -content content.txt

Create an encrypted message using 128 bit Camellia:

openssl cms -encrypt -in plain.txt -camellia128 -out mail.msg cert.pem

Add a signer to an existing message:

openssl cms -resign -in mail.msg -signer newsign.pem -out mail2.msg

Sign a message using RSA-PSS:

openssl cms -sign -in message.txt -text -out mail.msg \

-signer mycert.pem -keyopt rsa_padding_mode:pss

Create an encrypted message using RSA-OAEP:

openssl cms -encrypt -in plain.txt -out mail.msg \

-recip cert.pem -keyopt rsa_padding_mode:oaep

Use SHA256 KDF with an ECDH certificate:

openssl cms -encrypt -in plain.txt -out mail.msg \

-recip ecdhcert.pem -keyopt ecdh_kdf_md:sha256

Print CMS signed binary data in human-readable form:

openssl cms -in signed.cms -binary -inform DER -cmsout -print

BUGS

The MIME parser isn't very clever: it seems to handle most messages that I've thrown at it but it may choke on others.

The code currently will only write out the signer's certificate to a file: if the signer has a separate encryption certificate this must be manually extracted. There should be some heuristic that determines the correct encryption certificate.

Ideally a database should be maintained of a certificates for each email address.

The code doesn't currently take note of the permitted symmetric encryption algorithms as supplied in the SMIMECapabilities signed attribute. this means the user has to manually include the correct encryption algorithm. It should store the list of permitted ciphers in a database and only use those.

No revocation checking is done on the signer's certificate.

SEE ALSO

ossl_store-file(7)

HISTORY

The use of multiple -signer options and the -resign command were first added in OpenSSL 1.0.0.

The -keyopt option was added in OpenSSL 1.0.2.

Support for RSA-OAEP and RSA-PSS was added in OpenSSL 1.0.2.

The use of non-RSA keys with -encrypt and -decrypt was added in OpenSSL 1.0.2.

The -no_alt_chains option was added in OpenSSL 1.0.2b.

The -nameopt option was added in OpenSSL 3.0.0.

The -engine option was deprecated in OpenSSL 3.0.

COPYRIGHT