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-ts.1ossl' command

$ man openssl-ts.1ossl

OPENSSL-TS(1ossl) OpenSSL OPENSSL-TS(1ossl)

NAME

openssl-ts - Time Stamping Authority command

SYNOPSIS

openssl ts -help

openssl ts -query [-config configfile] [-data file_to_hash] [-digest

digest_bytes] [-digest] [-tspolicy object_id] [-no_nonce] [-cert] [-in

request.tsq] [-out request.tsq] [-text] [-rand files] [-writerand file]

[-provider name] [-provider-path path] [-propquery propq]

openssl ts -reply [-config configfile] [-section tsa_section]

[-queryfile request.tsq] [-passin password_src] [-signer tsa_cert.pem]

[-inkey filename|uri] [-digest] [-chain certs_file.pem] [-tspolicy

object_id] [-in response.tsr] [-token_in] [-out response.tsr]

[-token_out] [-text] [-engine id] [-provider name] [-provider-path

path] [-propquery propq]

openssl ts -verify [-data file_to_hash] [-digest digest_bytes]

[-queryfile request.tsq] [-in response.tsr] [-token_in] [-untrusted

files|uris] [-CAfile file] [-CApath dir] [-CAstore uri]

[-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]
[-provider name] [-provider-path path] [-propquery propq]

DESCRIPTION

This command is a basic Time Stamping Authority (TSA) client and server application as specified in RFC 3161 (Time-Stamp Protocol, TSP). A TSA can be part of a PKI deployment and its role is to provide long term proof of the existence of a certain datum before a particular time.

Here is a brief description of the protocol:

1. The TSA client computes a one-way hash value for a data file and sends the hash to the TSA.
2. The TSA attaches the current date and time to the received hash value, signs them and sends the timestamp token back to the client. By creating this token the TSA certifies the existence of the original data file at the time of response generation.
3. The TSA client receives the timestamp token and verifies the signature on it. It also checks if the token contains the same hash value that it had sent to the TSA.

There is one DER encoded protocol data unit defined for transporting a timestamp request to the TSA and one for sending the timestamp response back to the client. This command has three main functions: creating a timestamp request based on a data file, creating a timestamp response based on a request, verifying if a response corresponds to a particular request or a data file.

There is no support for sending the requests/responses automatically over HTTP or TCP yet as suggested in RFC 3161. The users must send the requests either by ftp or e-mail.

OPTIONS

-help

Print out a usage message.

-query

Generate a TS query. For details see "Timestamp Request

generation".

-reply

Generate a TS reply. For details see "Timestamp Response generation".

-verify

Verify a TS response. For details see "Timestamp Response verification".

Timestamp Request generation

The -query command can be used for creating and printing a timestamp request with the following options:

-config configfile

The configuration file to use. Optional; for a description of the default value, see "COMMAND SUMMARY" in openssl(1).

-data file_to_hash

The data file for which the timestamp request needs to be created. stdin is the default if neither the -data nor the -digest parameter is specified. (Optional)

-digest digest_bytes

It is possible to specify the message imprint explicitly without the data file. The imprint must be specified in a hexadecimal format, two characters per byte, the bytes optionally separated by colons (e.g. 1A:F6:01:... or 1AF601...). The number of bytes must match the message digest algorithm in use. (Optional)

-digest

The message digest to apply to the data file. Any digest supported by the openssl-dgst(1) command can be used. The default is SHA-256. (Optional)

-tspolicy object_id

The policy that the client expects the TSA to use for creating the timestamp token. Either the dotted OID notation or OID names defined in the config file can be used. If no policy is requested the TSA will use its own default policy. (Optional)

-no_nonce

No nonce is specified in the request if this option is given. Otherwise a 64 bit long pseudo-random none is included in the request. It is recommended to use nonce to protect against replay-attacks. (Optional)

-cert

The TSA is expected to include its signing certificate in the response. (Optional)

-in request.tsq

This option specifies a previously created timestamp request in DER format that will be printed into the output file. Useful when you need to examine the content of a request in human-readable format. (Optional)

-out request.tsq

Name of the output file to which the request will be written. Default is stdout. (Optional)

-text

If this option is specified the output is human-readable text format instead of DER. (Optional)

-rand files, -writerand file

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

Timestamp Response generation

A timestamp response (TimeStampResp) consists of a response status and the timestamp token itself (ContentInfo), if the token generation was successful. The -reply command is for creating a timestamp response or timestamp token based on a request and printing the response/token in human-readable format. If -token_out is not specified the output is always a timestamp response (TimeStampResp), otherwise it is a timestamp token (ContentInfo).

-config configfile

The configuration file to use. Optional; for a description of the default value, see "COMMAND SUMMARY" in openssl(1). See "CONFIGURATION FILE OPTIONS" for configurable variables.

-section tsa_section

The name of the config file section containing the settings for the response generation. If not specified the default TSA section is used, see "CONFIGURATION FILE OPTIONS" for details. (Optional)

-queryfile request.tsq

The name of the file containing a DER encoded timestamp request. (Optional)

-passin password_src

Specifies the password source for the private key of the TSA. See description in openssl(1). (Optional)

-signer tsa_cert.pem

The signer certificate of the TSA in PEM format. The TSA signing certificate must have exactly one extended key usage assigned to it: timeStamping. The extended key usage must also be critical, otherwise the certificate is going to be refused. Overrides the signer_cert variable of the config file. (Optional)

-inkey filename|uri

The signer private key of the TSA in PEM format. Overrides the signer_key config file option. (Optional)

-digest

Signing digest to use. Overrides the signer_digest config file option. (Mandatory unless specified in the config file)

-chain certs_file.pem

The collection of certificates in PEM format that will all be included in the response in addition to the signer certificate if the -cert option was used for the request. This file is supposed to contain the certificate chain for the signer certificate from its issuer upwards. The -reply command does not build a certificate chain automatically. (Optional)

-tspolicy object_id

The default policy to use for the response unless the client explicitly requires a particular TSA policy. The OID can be specified either in dotted notation or with its name. Overrides the default_policy config file option. (Optional)

-in response.tsr

Specifies a previously created timestamp response or timestamp token (if -token_in is also specified) in DER format that will be written to the output file. This option does not require a request, it is useful e.g. when you need to examine the content of a response or token or you want to extract the timestamp token from a response. If the input is a token and the output is a timestamp response a default 'granted' status info is added to the token. (Optional)

-token_in

This flag can be used together with the -in option and indicates that the input is a DER encoded timestamp token (ContentInfo) instead of a timestamp response (TimeStampResp). (Optional)

-out response.tsr

The response is written to this file. The format and content of the file depends on other options (see -text, -token_out). The default is stdout. (Optional)

-token_out

The output is a timestamp token (ContentInfo) instead of timestamp response (TimeStampResp). (Optional)

-text

If this option is specified the output is human-readable text format instead of DER. (Optional)

-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).

Timestamp Response verification

The -verify command is for verifying if a timestamp response or timestamp token is valid and matches a particular timestamp request or data file. The -verify command does not use the configuration file.

-data file_to_hash

The response or token must be verified against file_to_hash. The file is hashed with the message digest algorithm specified in the token. The -digest and -queryfile options must not be specified with this one. (Optional)

-digest digest_bytes

The response or token must be verified against the message digest specified with this option. The number of bytes must match the message digest algorithm specified in the token. The -data and -queryfile options must not be specified with this one. (Optional)

-queryfile request.tsq

The original timestamp request in DER format. The -data and -digest options must not be specified with this one. (Optional)

-in response.tsr

The timestamp response that needs to be verified in DER format. (Mandatory)

-token_in

This flag can be used together with the -in option and indicates that the input is a DER encoded timestamp token (ContentInfo) instead of a timestamp response (TimeStampResp). (Optional)

-untrusted files|uris

A set of additional untrusted certificates which may be needed when building the certificate chain for the TSA's signing certificate. These do not need to contain the TSA signing certificate and intermediate CA certificates as far as the response already includes them. (Optional)

Multiple sources may be given, separated by commas and/or whitespace. Each file may contain multiple certificates.

-CAfile file, -CApath dir, -CAstore uri

See "Trusted Certificate Options" in openssl-verification-options(1) for details. At least one of -CAfile, -CApath or -CAstore must be specified.

-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 verification errors cause the command to exit.

CONFIGURATION FILE OPTIONS

The -query and -reply commands make use of a configuration file. See config(5) for a general description of the syntax of the config file.

The -query command uses only the symbolic OID names section and it can work without it. However, the -reply command needs the config file for its operation.

When there is a command line switch equivalent of a variable the switch always overrides the settings in the config file.

tsa section, default_tsa

This is the main section and it specifies the name of another section that contains all the options for the -reply command. This default section can be overridden with the -section command line switch. (Optional)

oid_file

This specifies a file containing additional OBJECT IDENTIFIERS. Each line of the file should consist of the numerical form of the object identifier followed by whitespace then the short name followed by whitespace and finally the long name. (Optional)

oid_section

This specifies a section in the configuration file containing extra object identifiers. Each line should consist of the short name of the object identifier followed by = and the numerical form. The short and long names are the same when this option is used.

(Optional)

RANDFILE

At startup the specified file is loaded into the random number generator, and at exit 256 bytes will be written to it. (Note: Using a RANDFILE is not necessary anymore, see the "HISTORY" section.

serial

The name of the file containing the hexadecimal serial number of the last timestamp response created. This number is incremented by 1 for each response. If the file does not exist at the time of response generation a new file is created with serial number 1. (Mandatory)

crypto_device

Specifies the OpenSSL engine that will be set as the default for all available algorithms. The default value is built-in, you can specify any other engines supported by OpenSSL (e.g. use chil for the NCipher HSM). (Optional)

signer_cert

TSA signing certificate in PEM format. The same as the -signer command line option. (Optional)

certs

A file containing a set of PEM encoded certificates that need to be included in the response. The same as the -chain command line option. (Optional)

signer_key

The private key of the TSA in PEM format. The same as the -inkey command line option. (Optional)

signer_digest

Signing digest to use. The same as the -digest command line option. (Mandatory unless specified on the command line)

default_policy

The default policy to use when the request does not mandate any policy. The same as the -tspolicy command line option. (Optional)

other_policies

Comma separated list of policies that are also acceptable by the TSA and used only if the request explicitly specifies one of them. (Optional)

digests

The list of message digest algorithms that the TSA accepts. At least one algorithm must be specified. (Mandatory)

accuracy

The accuracy of the time source of the TSA in seconds, milliseconds and microseconds. E.g. secs:1, millisecs:500, microsecs:100. If any of the components is missing zero is assumed for that field. (Optional)

clock_precision_digits

Specifies the maximum number of digits, which represent the fraction of seconds, that need to be included in the time field. The trailing zeros must be removed from the time, so there might actually be fewer digits, or no fraction of seconds at all. Supported only on UNIX platforms. The maximum value is 6, default is 0. (Optional)

ordering

If this option is yes the responses generated by this TSA can always be ordered, even if the time difference between two responses is less than the sum of their accuracies. Default is no. (Optional)

tsa_name

Set this option to yes if the subject name of the TSA must be included in the TSA name field of the response. Default is no. (Optional)

ess_cert_id_chain

The SignedData objects created by the TSA always contain the certificate identifier of the signing certificate in a signed attribute (see RFC 2634, Enhanced Security Services). If this variable is set to no, only this signing certificate identifier is

included in the SigningCertificate signed attribute. If this variable is set to yes and the certs variable or the -chain option is specified then the certificate identifiers of the chain will also be included, where the -chain option overrides the certs variable. Default is no. (Optional)

ess_cert_id_alg

This option specifies the hash function to be used to calculate the TSA's public key certificate identifier. Default is sha256. (Optional)

EXAMPLES

All the examples below presume that OPENSSL_CONF is set to a proper configuration file, e.g. the example configuration file openssl/apps/openssl.cnf will do.

Timestamp Request

To create a timestamp request for design1.txt with SHA-256 digest, without nonce and policy, and without requirement for a certificate in the response:

```
openssl ts -query -data design1.txt -no_nonce \
    -out design1.tsq
```

To create a similar timestamp request with specifying the message imprint explicitly:

```
openssl ts -query -digest b7e5d3f93198b38379852f2c04e78d73abdd0f4b \
    -no_nonce -out design1.tsq
```

To print the content of the previous request in human readable format:

```
openssl ts -query -in design1.tsq -text
```

To create a timestamp request which includes the SHA-512 digest of design2.txt, requests the signer certificate and nonce, and specifies a policy id (assuming the tsa_policy1 name is defined in the OID section of the config file):

```
openssl ts -query -data design2.txt -sha512 \
    -tspolicy tsa_policy1 -cert -out design2.tsq
```

Timestamp Response

Before generating a response a signing certificate must be created for

the TSA that contains the timeStamping critical extended key usage extension without any other key usage extensions. You can add this line to the user certificate section of the config file to generate a proper certificate;

extendedKeyUsage = critical,timeStamping

See openssl-req(1), openssl-ca(1), and openssl-x509(1) for instructions. The examples below assume that cacert.pem contains the certificate of the CA, tsacert.pem is the signing certificate issued by cacert.pem and tsakey.pem is the private key of the TSA.

To create a timestamp response for a request:

```
openssl ts -reply -queryfile design1.tsq -inkey tsakey.pem \
      -signer tsacert.pem -out design1.tsr
```

If you want to use the settings in the config file you could just write:

```
openssl ts -reply -queryfile design1.tsq -out design1.tsr
```

To print a timestamp reply to stdout in human readable format:

```
openssl ts -reply -in design1.tsr -text
```

To create a timestamp token instead of timestamp response:

```
openssl ts -reply -queryfile design1.tsq -out design1_token.der -token_out
```

To print a timestamp token to stdout in human readable format:

```
openssl ts -reply -in design1_token.der -token_in -text -token_out
```

To extract the timestamp token from a response:

```
openssl ts -reply -in design1.tsr -out design1_token.der -token_out
```

To add 'granted' status info to a timestamp token thereby creating a valid response:

```
openssl ts -reply -in design1_token.der -token_in -out design1.tsr
```

Timestamp Verification

To verify a timestamp reply against a request:

```
openssl ts -verify -queryfile design1.tsq -in design1.tsr \
      -CAfile cacert.pem -untrusted tsacert.pem
```

To verify a timestamp reply that includes the certificate chain:

```
openssl ts -verify -queryfile design2.tsq -in design2.tsr \
      -CAfile cacert.pem
```

To verify a timestamp token against the original data file:

```
openssl ts -verify -data design2.txt -in design2.tsr \
    -CAfile cacert.pem
```

To verify a timestamp token against a message imprint:

```
openssl ts -verify -digest b7e5d3f93198b38379852f2c04e78d73abdd0f4b \
     -in design2.tsr -CAfile cacert.pem
```

You could also look at the 'test' directory for more examples.

BUGS

? No support for timestamps over SMTP, though it is quite easy to implement an automatic e-mail based TSA with procmail(1) and perl(1). HTTP server support is provided in the form of a separate apache module. HTTP client support is provided by tsget(1). Pure TCP/IP protocol is not supported.

? The file containing the last serial number of the TSA is not locked when being read or written. This is a problem if more than one instance of openssl(1) is trying to create a timestamp response at the same time. This is not an issue when using the apache server module, it does proper locking.

? Look for the FIXME word in the source files.

? The source code should really be reviewed by somebody else, too.

? More testing is needed, I have done only some basic tests (see test/testtsa).

HISTORY

OpenSSL 1.1.1 introduced a new random generator (CSPRNG) with an improved seeding mechanism. The new seeding mechanism makes it unnecessary to define a RANDFILE for saving and restoring randomness. This option is retained mainly for compatibility reasons.

The -engine option was deprecated in OpenSSL 3.0.

SEE ALSO

openssl(1), tsget(1), openssl-req(1), openssl-x509(1), openssl-ca(1), openssl-genrsa(1), config(5), ossl_store-file(7)

COPYRIGHT

Licensed under the Apache License 2.0 (the "License"). You may not use this file except in compliance with the License. You can obtain a copy in the file LICENSE in the source distribution or at <https://www.openssl.org/source/license.html>.