NAME

tsget - Time Stamping HTTP/HTTPS client

SYNOPSIS

tsget -h *server_url* [**-e** *extension*] [**-o** *output*] [**-v**] [**-d**] [**-k** *private_key.pem*] [**-p** *key_password*] [**-c** *client_cert.pem*] [**-C** *CA_certs.pem*] [**-P** *CA_path*] [**-r** *files*] [**-g** *EGD_socket*] [*request* ...]

DESCRIPTION

This command can be used for sending a timestamp request, as specified in RFC 3161, to a timestamp server over HTTP or HTTPS and storing the timestamp response in a file. It cannot be used for creating the requests and verifying responses, you have to use **openssl-ts**(1) to do that. This command can send several requests to the server without closing the TCP connection if more than one requests are specified on the command line.

This command sends the following HTTP request for each timestamp request:

```
POST url HTTP/1.1
User-Agent: OpenTSA tsget.pl/<version>
Host: <host>:<port>
Pragma: no-cache
Content-Type: application/timestamp-query
Accept: application/timestamp-reply
Content-Length: length of body

...binary request specified by the user...
```

It expects a response of type application/timestamp-reply, which is written to a file without any interpretation.

OPTIONS

-h *server_url*

The URL of the HTTP/HTTPS server listening for timestamp requests.

-e *extension*

If the **-o** option is not given this argument specifies the extension of the output files. The base name of the output file will be the same as those of the input files. Default extension is *.tsr*. (Optional)

-o *output*

This option can be specified only when just one request is sent to the server. The timestamp response will be written to the given output file. '-' means standard output. In case of multiple timestamp requests or the absence of this argument the names of the output files will be derived from the names of the input files and the default or specified extension argument. (Optional)

-v The name of the currently processed request is printed on standard error. (Optional)

-d Switches on verbose mode for the underlying perl module WWW::Curl::Easy. You can see detailed debug messages for the connection. (Optional)

-k *private_key.pem*
(HTTPS) In case of certificate-based client authentication over HTTPS *private_key.pem* must contain the private key of the user. The private key file can optionally be protected by a passphrase. The **-c** option must also be specified. (Optional)

-p *key_password*
(HTTPS) Specifies the passphrase for the private key specified by the **-k** argument. If this option is omitted and the key is passphrase protected, it will be prompted for. (Optional)

-c *client_cert.pem*
(HTTPS) In case of certificate-based client authentication over HTTPS *client_cert.pem* must contain the X.509 certificate of the user. The **-k** option must also be specified. If this option is not specified no certificate-based client authentication will take place. (Optional)

-C *CA_certs.pem*
(HTTPS) The trusted CA certificate store. The certificate chain of the peer's certificate must include one of the CA certificates specified in this file. Either option **-C** or option **-P** must be given in case of HTTPS. (Optional)

-P *CA_path*
(HTTPS) The path containing the trusted CA certificates to verify the peer's certificate. The directory must be prepared with **openssl-rehash**(1). Either option **-C** or option **-P** must be given in case of HTTPS. (Optional)

-r *files*
See "Random State Options" in **openssl**(1) for more information.

-g *EGD_socket*
The name of an EGD socket to get random data from. (Optional)

request ...

List of files containing RFC 3161 DER-encoded timestamp requests. If no requests are specified only one request will be sent to the server and it will be read from the standard input. (Optional)

ENVIRONMENT VARIABLES

The **TSGET** environment variable can optionally contain default arguments. The content of this variable is added to the list of command line arguments.

EXAMPLES

The examples below presume that *file1.tsq* and *file2.tsq* contain valid timestamp requests, tsa.opentsa.org listens at port 8080 for HTTP requests and at port 8443 for HTTPS requests, the TSA service is available at the /tsa absolute path.

Get a timestamp response for *file1.tsq* over HTTP, output is written to *file1.tsr*:

```
 tsget -h http://tsa.opentsa.org:8080/tsa file1.tsq
```

Get a timestamp response for *file1.tsq* and *file2.tsq* over HTTP showing progress, output is written to *file1.reply* and *file2.reply* respectively:

```
 tsget -h http://tsa.opentsa.org:8080/tsa -v -e .reply \
       file1.tsq file2.tsq
```

Create a timestamp request, write it to *file3.tsq*, send it to the server and write the response to *file3.tsr*:

```
 openssl ts -query -data file3.txt -cert | tee file3.tsq \
       | tsget -h http://tsa.opentsa.org:8080/tsa \
       -o file3.tsr
```

Get a timestamp response for *file1.tsq* over HTTPS without client authentication:

```
 tsget -h https://tsa.opentsa.org:8443/tsa \
       -C cacerts.pem file1.tsq
```

Get a timestamp response for *file1.tsq* over HTTPS with certificate-based client authentication (it will ask for the passphrase if *client_key.pem* is protected):

```
 tsget -h https://tsa.opentsa.org:8443/tsa -C cacerts.pem \
       -k client_key.pem -c client_cert.pem file1.tsq
```

You can shorten the previous command line if you make use of the **TSGET** environment variable. The following commands do the same as the previous example:

```
 TSGET='-h https://tsa.opentsa.org:8443/tsa -C cacerts.pem \
       -k client_key.pem -c client_cert.pem'
 export TSGET
 tsget file1.tsq
```

SEE ALSO

openssl(1), **openssl-ts**(1), WWW::Curl::Easy, <https://www.rfc-editor.org/rfc/rfc3161.html>

COPYRIGHT