

libpki Reference Manual

0.5.1

Generated by Doxygen 1.5.1

Thu Sep 2 20:22:36 2010

Contents

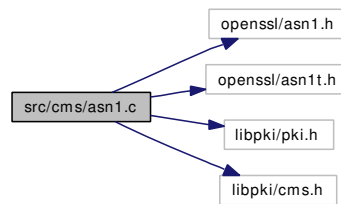
1 libpki File Documentation

1

1 libpki File Documentation

1.1 src/cms/asn1.c File Reference

Include dependency graph for `asn1.c`:



1.2 src/cms/cms_cert_req.c File Reference

Include dependency graph for `cms_cert_req.c`:



Functions

- `CERT_REQ_MSG * CERT_REQ_MSG_get (char *url_s)`
Retrieves a CERT_REQ_MSG request from the resource specified in the provided URI string.
- `CERT_REQ_MSG * CERT_REQ_MSG_get_fd (int fd)`
Retrieves a CERT_REQ_MSG request from the specified file descriptor.
- `CERT_REQ_MSG * CERT_REQ_MSG_get_mem (PKI_MEM *mem)`
Retrieves a CERT_REQ_MSG request from the passed PKI_MEM.
- `CERT_REQ_MSG * CERT_REQ_MSG_get_url (URL *url)`
Retrieves a CERT_REQ_MSG request from the resource specified in the provided URL structure.
- `int CERT_REQ_MSG_put (CERT_REQ_MSG *req, char *url_s, int format, PKI_MEM_STACK **ret_sk)`
Sends/Writes a CERT_REQ_MSG request in the resource specified in the provided URI string.
- `int CERT_REQ_MSG_put_fp (CERT_REQ_MSG *req, FILE *file, int format)`
Writes a CERT_REQ_MSG request in the provided file descriptor.
- `int CERT_REQ_MSG_put_mem (CERT_REQ_MSG *req, PKI_MEM *mem, int format)`
Sends/Store a CERT_REQ_MSG request in a PKI_MEM structure.

- int [CERT_REQ_MSG_put_url](#) (CERT_REQ_MSG *req, URL *url, int format, PKI_MEM_STACK **ret_sk)

Sends/Writes a CERT_REQ_MSG request in the resource specified in the provided URL structure.

- CERT_REQ_MSG * [d2i_CERT_REQ_MSG_bio](#) (BIO *bp, CERT_REQ_MSG *p)
- int [i2d_CERT_REQ_MSG_bio](#) (BIO *bp, CERT_REQ_MSG *o)
- CERT_REQ_MSG * [PEM_read_bio_CERT_REQ_MSG](#) (BIO *bp)
- int [PEM_write_bio_CERT_REQ_MSG](#) (BIO *bp, CERT_REQ_MSG *o)

1.2.1 Function Documentation

1.2.1.1 CERT_REQ_MSG* CERT_REQ_MSG_get (char * url_s)

Retrieves a CERT_REQ_MSG request from the resource specified in the provided URI string.

1.2.1.2 CERT_REQ_MSG* CERT_REQ_MSG_get_fd (int fd)

Retrieves a CERT_REQ_MSG request from the specified file descriptor.

1.2.1.3 CERT_REQ_MSG* CERT_REQ_MSG_get_mem (PKI_MEM * mem)

Retrieves a CERT_REQ_MSG request from the passed PKI_MEM.

1.2.1.4 CERT_REQ_MSG* CERT_REQ_MSG_get_url (URL * url)

Retrieves a CERT_REQ_MSG request from the resource specified in the provided URL structure.

1.2.1.5 int CERT_REQ_MSG_put (CERT_REQ_MSG * req, char * url_s, int format, PKI_MEM_STACK ** ret_sk)

Sends/Writes a CERT_REQ_MSG request in the resource specified in the provided URI string.

1.2.1.6 int CERT_REQ_MSG_put_fp (CERT_REQ_MSG * req, FILE * file, int format)

Writes a CERT_REQ_MSG request in the provided file descriptor.

1.2.1.7 int CERT_REQ_MSG_put_mem (CERT_REQ_MSG * req, PKI_MEM * mem, int format)

Sends/Store a CERT_REQ_MSG request in a PKI_MEM structure.

1.2.1.8 int CERT_REQ_MSG_put_url (CERT_REQ_MSG * req, URL * url, int format, PKI_MEM_STACK ** ret_sk)

Sends/Writes a CERT_REQ_MSG request in the resource specified in the provided URL structure.

1.2.1.9 CERT_REQ_MSG* d2i_CERT_REQ_MSG_bio (BIO * bp, CERT_REQ_MSG * p)

1.2.1.10 int i2d_CERT_REQ_MSG_bio (BIO * bp, CERT_REQ_MSG * o)

1.2.1.11 CERT_REQ_MSG* PEM_read_bio_CERT_REQ_MSG (BIO * *bp*)

1.2.1.12 int PEM_write_bio_CERT_REQ_MSG (BIO * *bp*, CERT_REQ_MSG * *o*)

1.3 src/cms/cms_simple.c File Reference

Include dependency graph for cms_simple.c:



Typedefs

- typedef void [PKI_CMS_REQ](#)
- typedef void [PKI_CMS_RESP](#)

Functions

- [PKI_CMS_RESP](#) * [PKI_MSG_CMS_write](#) ([PKI_CMS_REQ](#) *req, URL *url)

1.3.1 Typedef Documentation

1.3.1.1 typedef void [PKI_CMS_REQ](#)

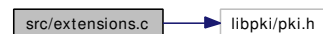
1.3.1.2 typedef void [PKI_CMS_RESP](#)

1.3.2 Function Documentation

1.3.2.1 [PKI_CMS_RESP](#)* [PKI_MSG_CMS_write](#) ([PKI_CMS_REQ](#) * *req*, URL * *url*)

1.4 src/extensions.c File Reference

Include dependency graph for extensions.c:



Functions

- int [PKI_X509_EXTENSIONS_cert_add_profile](#) (PKI_X509_PROFILE *conf, PKI_CONFIG *oids, PKI_X509_CERT *x)
- int [PKI_X509_EXTENSIONS_crl_add_profile](#) (PKI_X509_PROFILE *conf, PKI_CONFIG *oids, PKI_X509_CRL *crl)
- int [PKI_X509_EXTENSIONS_req_add_profile](#) (PKI_X509_PROFILE *conf, PKI_CONFIG *oids, PKI_X509_REQ *req)

1.4.1 Function Documentation

1.4.1.1 `int PKI_X509_EXTENSIONS_cert_add_profile (PKI_X509_PROFILE * conf, PKI_CONFIG * oids, PKI_X509_CERT * x)`

1.4.1.2 `int PKI_X509_EXTENSIONS_crl_add_profile (PKI_X509_PROFILE * conf, PKI_CONFIG * oids, PKI_X509_CRL * crl)`

1.4.1.3 `int PKI_X509_EXTENSIONS_req_add_profile (PKI_X509_PROFILE * conf, PKI_CONFIG * oids, PKI_X509_REQ * req)`

1.5 src/io/pki_keypair_io.c File Reference

Include dependency graph for pki_keypair_io.c:



Functions

- `PKI_X509_KEYPAIR * PKI_X509_KEYPAIR_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
Returns a `PKI_X509_KEYPAIR` object from a url address (string).
- `PKI_X509_KEYPAIR * PKI_X509_KEYPAIR_get_mem (PKI_MEM *mem, PKI_CRED *cred)`
Reads a `PKI_X509_KEYPAIR` object from a `PKI_MEM`.
- `PKI_X509_KEYPAIR * PKI_X509_KEYPAIR_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
Returns a `PKI_X509_KEYPAIR` object from a URL.
- `int PKI_X509_KEYPAIR_put (PKI_X509_KEYPAIR *x, PKI_DATA_FORMAT format, char *url_string, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_KEYPAIR_put_mem (PKI_X509_KEYPAIR *key, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
Puts a `X509_KEYPAIR` to a `PKI_MEM`.
- `int PKI_X509_KEYPAIR_put_url (PKI_X509_KEYPAIR *x, PKI_DATA_FORMAT format, URL *url, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_KEYPAIR_STACK * PKI_X509_KEYPAIR_STACK_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
Returns a `STACK` of `PKI_X509_KEYPAIR` objects from a url address.
- `PKI_X509_KEYPAIR_STACK * PKI_X509_KEYPAIR_STACK_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
Returns a `STACK` of `PKI_X509_KEYPAIR` objects from the passed URL.

1.5.1 Function Documentation

1.5.1.1 PKI_X509_KEYPAIR* PKI_X509_KEYPAIR_get (char * url_s, PKI_CRED * cred, HSM * hsm)

Returns a PKI_X509_KEYPAIR object from a url address (string).

1.5.1.2 PKI_X509_KEYPAIR* PKI_X509_KEYPAIR_get_mem (PKI_MEM * mem, PKI_CRED * cred)

Reads a PKI_X509_KEYPAIR object from a PKI_MEM.

1.5.1.3 PKI_X509_KEYPAIR* PKI_X509_KEYPAIR_get_url (URL * url, PKI_CRED * cred, HSM * hsm)

Returns a PKI_X509_KEYPAIR object from a URL.

1.5.1.4 int PKI_X509_KEYPAIR_put (PKI_X509_KEYPAIR * x, PKI_DATA_FORMAT format, char * url_string, PKI_CRED * cred, HSM * hsm)

1.5.1.5 PKI_MEM* PKI_X509_KEYPAIR_put_mem (PKI_X509_KEYPAIR * key, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)

Puts a X509_KEYPAIR to a PKI_MEM.

1.5.1.6 int PKI_X509_KEYPAIR_put_url (PKI_X509_KEYPAIR * x, PKI_DATA_FORMAT format, URL * url, PKI_CRED * cred, HSM * hsm)

1.5.1.7 PKI_X509_KEYPAIR_STACK* PKI_X509_KEYPAIR_STACK_get (char * url_s, PKI_CRED * cred, HSM * hsm)

Returns a STACK of PKI_X509_KEYPAIR objects from a url address.

1.5.1.8 PKI_X509_KEYPAIR_STACK* PKI_X509_KEYPAIR_STACK_get_url (URL * url, PKI_CRED * cred, HSM * hsm)

Returns a STACK of PKI_X509_KEYPAIR objects from the passed URL.

1.6 src/io/pki_msg_req_io.c File Reference

Include dependency graph for pki_msg_req_io.c:



Functions

- int [PKI_MSG_REQ_put](#) (PKI_MSG_REQ *msg, PKI_DATA_FORMAT format, char *url, char *mime, PKI_CRED *cred, HSM *hsm, PKI_MEM_STACK **ret_sk)

Writes the message to a URL.

- PKI_MEM * [PKI_MSG_REQ_put_mem](#) (PKI_MSG_REQ *msg, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)

Writes the message in a memory buffer.

1.6.1 Function Documentation

1.6.1.1 int [PKI_MSG_REQ_put](#) (PKI_MSG_REQ *msg, PKI_DATA_FORMAT format, char *url, char *mime, PKI_CRED *cred, HSM *hsm, PKI_MEM_STACK **ret_sk)

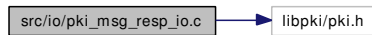
Writes the message to a URL.

1.6.1.2 PKI_MEM* [PKI_MSG_REQ_put_mem](#) (PKI_MSG_REQ *msg, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)

Writes the message in a memory buffer.

1.7 src/io/pki_msg_resp_io.c File Reference

Include dependency graph for pki_msg_resp_io.c:



Functions

- int [PKI_MSG_RESP_put](#) (PKI_MSG_RESP *msg, PKI_DATA_FORMAT format, char *url, char *mime, PKI_CRED *cred, HSM *hsm)

Writes the message to a URL.

- PKI_MEM * [PKI_MSG_RESP_put_mem](#) (PKI_MSG_RESP *msg, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)

Writes the message in a memory buffer.

1.7.1 Function Documentation

1.7.1.1 int [PKI_MSG_RESP_put](#) (PKI_MSG_RESP *msg, PKI_DATA_FORMAT format, char *url, char *mime, PKI_CRED *cred, HSM *hsm)

Writes the message to a URL.

1.7.1.2 PKI_MEM* [PKI_MSG_RESP_put_mem](#) (PKI_MSG_RESP *msg, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)

Writes the message in a memory buffer.

1.8 src/io/pki_ocsp_req_io.c File Reference

Include dependency graph for pki_ocsp_req_io.c:



Functions

- PKI_X509_OCSP_REQ * [PKI_X509_OCSP_REQ_get](#) (char *url_s, PKI_CRED *cred, HSM *hsm)
- PKI_X509_OCSP_REQ * [PKI_X509_OCSP_REQ_get_mem](#) (PKI_MEM *mem, PKI_CRED *cred)
- PKI_X509_OCSP_REQ * [PKI_X509_OCSP_REQ_get_url](#) (URL *url, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_OCSP_REQ_put](#) (PKI_X509_OCSP_REQ *req, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_OCSP_REQ_put_mem](#) (PKI_X509_OCSP_REQ *req, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_OCSP_REQ_put_url](#) (PKI_X509_OCSP_REQ *req, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_X509_OCSP_REQ_STACK * [PKI_X509_OCSP_REQ_STACK_get](#) (char *url_s, PKI_CRED *cred, HSM *hsm)
- PKI_X509_OCSP_REQ_STACK * [PKI_X509_OCSP_REQ_STACK_get_mem](#) (PKI_MEM *mem, PKI_CRED *cred)
- PKI_X509_OCSP_REQ_STACK * [PKI_X509_OCSP_REQ_STACK_get_url](#) (URL *url, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_OCSP_REQ_STACK_put](#) (PKI_X509_OCSP_REQ_STACK *sk, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_OCSP_REQ_STACK_put_mem](#) (PKI_X509_OCSP_REQ_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_OCSP_REQ_STACK_put_url](#) (PKI_X509_OCSP_REQ_STACK *sk, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)

1.8.1 Function Documentation

1.8.1.1 PKI_X509_OCSP_REQ* [PKI_X509_OCSP_REQ_get](#) (char * *url_s*, PKI_CRED * *cred*, HSM * *hsm*)

1.8.1.2 PKI_X509_OCSP_REQ* [PKI_X509_OCSP_REQ_get_mem](#) (PKI_MEM * *mem*, PKI_CRED * *cred*)

1.8.1.3 PKI_X509_OCSP_REQ* [PKI_X509_OCSP_REQ_get_url](#) (URL * *url*, PKI_CRED * *cred*, HSM * *hsm*)

1.8.1.4 int [PKI_X509_OCSP_REQ_put](#) (PKI_X509_OCSP_REQ * *req*, PKI_DATA_FORMAT *format*, char * *url_s*, char * *mime*, PKI_CRED * *cred*, HSM * *hsm*)

1.8.1.5 `PKI_MEM* PKI_X509_OCSP_REQ_put_mem (PKI_X509_OCSP_REQ * req, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.8.1.6 `int PKI_X509_OCSP_REQ_put_url (PKI_X509_OCSP_REQ * req, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.8.1.7 `PKI_X509_OCSP_REQ_STACK* PKI_X509_OCSP_REQ_STACK_get (char * url_s, PKI_CRED * cred, HSM * hsm)`

1.8.1.8 `PKI_X509_OCSP_REQ_STACK* PKI_X509_OCSP_REQ_STACK_get_mem (PKI_MEM * mem, PKI_CRED * cred)`

1.8.1.9 `PKI_X509_OCSP_REQ_STACK* PKI_X509_OCSP_REQ_STACK_get_url (URL * url, PKI_CRED * cred, HSM * hsm)`

1.8.1.10 `int PKI_X509_OCSP_REQ_STACK_put (PKI_X509_OCSP_REQ_STACK * sk, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.8.1.11 `PKI_MEM* PKI_X509_OCSP_REQ_STACK_put_mem (PKI_X509_OCSP_REQ_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.8.1.12 `int PKI_X509_OCSP_REQ_STACK_put_url (PKI_X509_OCSP_REQ_STACK * sk, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.9 src/io/pki_ocsp_resp_io.c File Reference

Include dependency graph for pki_ocsp_resp_io.c:



Functions

- `PKI_X509_OCSP_RESP * PKI_X509_OCSP_RESP_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_OCSP_RESP * PKI_X509_OCSP_RESP_get_mem (PKI_MEM *mem, PKI_CRED *cred)`
- `PKI_X509_OCSP_RESP * PKI_X509_OCSP_RESP_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_OCSP_RESP_put (PKI_X509_OCSP_RESP *r, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_OCSP_RESP_put_mem (PKI_X509_OCSP_RESP *r, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_OCSP_RESP_put_url (PKI_X509_OCSP_RESP *r, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)`

- `PKI_X509_OCSP_RESP_STACK * PKI_X509_OCSP_RESP_STACK_get` (`char *url_s`, `PKI_CRED *cred`, `HSM *hsm`)
- `PKI_X509_OCSP_RESP_STACK * PKI_X509_OCSP_RESP_STACK_get_mem` (`PKI_MEM *mem`, `PKI_CRED *cred`)
- `PKI_X509_OCSP_RESP_STACK * PKI_X509_OCSP_RESP_STACK_get_url` (`URL *url`, `PKI_CRED *cred`, `HSM *hsm`)
- `int PKI_X509_OCSP_RESP_STACK_put` (`PKI_X509_OCSP_RESP_STACK *sk`, `PKI_DATA_FORMAT format`, `char *url_s`, `char *mime`, `PKI_CRED *cred`, `HSM *hsm`)
- `PKI_MEM * PKI_X509_OCSP_RESP_STACK_put_mem` (`PKI_X509_OCSP_RESP_STACK *sk`, `PKI_DATA_FORMAT format`, `PKI_MEM **pki_mem`, `PKI_CRED *cred`, `HSM *hsm`)
- `int PKI_X509_OCSP_RESP_STACK_put_url` (`PKI_X509_OCSP_RESP_STACK *sk`, `PKI_DATA_FORMAT format`, `URL *url`, `char *mime`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1 Function Documentation

1.9.1.1 `PKI_X509_OCSP_RESP* PKI_X509_OCSP_RESP_get` (`char *url_s`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1.2 `PKI_X509_OCSP_RESP* PKI_X509_OCSP_RESP_get_mem` (`PKI_MEM *mem`, `PKI_CRED *cred`)

1.9.1.3 `PKI_X509_OCSP_RESP* PKI_X509_OCSP_RESP_get_url` (`URL *url`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1.4 `int PKI_X509_OCSP_RESP_put` (`PKI_X509_OCSP_RESP *r`, `PKI_DATA_FORMAT format`, `char *url_s`, `char *mime`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1.5 `PKI_MEM* PKI_X509_OCSP_RESP_put_mem` (`PKI_X509_OCSP_RESP *r`, `PKI_DATA_FORMAT format`, `PKI_MEM **pki_mem`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1.6 `int PKI_X509_OCSP_RESP_put_url` (`PKI_X509_OCSP_RESP *r`, `PKI_DATA_FORMAT format`, `URL *url`, `char *mime`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1.7 `PKI_X509_OCSP_RESP_STACK* PKI_X509_OCSP_RESP_STACK_get` (`char *url_s`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1.8 `PKI_X509_OCSP_RESP_STACK* PKI_X509_OCSP_RESP_STACK_get_mem` (`PKI_MEM *mem`, `PKI_CRED *cred`)

1.9.1.9 `PKI_X509_OCSP_RESP_STACK* PKI_X509_OCSP_RESP_STACK_get_url` (`URL *url`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1.10 `int PKI_X509_OCSP_RESP_STACK_put` (`PKI_X509_OCSP_RESP_STACK *sk`, `PKI_DATA_FORMAT format`, `char *url_s`, `char *mime`, `PKI_CRED *cred`, `HSM *hsm`)

1.9.1.11 `PKI_MEM* PKI_X509_OCSP_RESP_STACK_put_mem (PKI_X509_OCSP_RESP_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.9.1.12 `int PKI_X509_OCSP_RESP_STACK_put_url (PKI_X509_OCSP_RESP_STACK * sk, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.10 src/io/pki_x509_cert_io.c File Reference

Include dependency graph for pki_x509_cert_io.c:



Functions

- `PKI_X509_CERT * PKI_X509_CERT_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
Retrieve a certificate from a URL.
- `PKI_X509_CERT * PKI_X509_CERT_get_mem (PKI_MEM *mem, PKI_CRED *cred)`
- `PKI_X509_CERT * PKI_X509_CERT_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
Retrieve a certificate from a URL pointer.
- `int PKI_X509_CERT_put (PKI_X509_CERT *x, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_CERT_put_mem (PKI_X509_CERT *x, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_CERT_put_url (PKI_X509_CERT *x, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_CERT_STACK * PKI_X509_CERT_STACK_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
*Retrieve a stack of certificates from a URL (char *).*
- `PKI_X509_CERT_STACK * PKI_X509_CERT_STACK_get_mem (PKI_MEM *mem, PKI_CRED *cred)`
- `PKI_X509_CERT_STACK * PKI_X509_CERT_STACK_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
*Retrieve a stack of certificates from a URL (URL *) pointer.*
- `int PKI_X509_CERT_STACK_put (PKI_X509_CERT_STACK *sk, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_CERT_STACK_put_mem (PKI_X509_CERT_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_CERT_STACK_put_url (PKI_X509_CERT_STACK *sk, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)`

1.10.1 Function Documentation

1.10.1.1 PKI_X509_CERT* PKI_X509_CERT_get (char * *url_s*, PKI_CRED * *cred*, HSM * *hsm*)

Retrieve a certificate from a URL.

Downloads a certificate from a given URL ([file://](#), [http://](#), [ldap://...](#)) in (char *) format. The returned data is of type PKI_X509_CERT in case of success or NULL if any error occurred. If multiple objects are returned from the URL, only the first one is returned. Use [PKI_X509_CERT_STACK_get\(\)](#) function to retrieve a PKI_X509_CERT_STACK * object.

1.10.1.2 PKI_X509_CERT* PKI_X509_CERT_get_mem (PKI_MEM * *mem*, PKI_CRED * *cred*)

1.10.1.3 PKI_X509_CERT* PKI_X509_CERT_get_url (URL * *url*, PKI_CRED * *cred*, HSM * *hsm*)

Retrieve a certificate from a URL pointer.

Downloads a certificate from a given URL ([file://](#), [http://](#), [ldap://...](#)) in (URL *) format. To generate a URL * from a char * use [URL_new\(\)](#). The returned data is of type PKI_X509_CERT in case of success or NULL if any error occurred. If multiple objects are returned from the URL, only the first one is returned. Use [PKI_X509_CERT_STACK_get_url\(\)](#) function to retrieve a PKI_X509_CERT_STACK * object.

1.10.1.4 int PKI_X509_CERT_put (PKI_X509_CERT * *x*, PKI_DATA_FORMAT *format*, char * *url_s*, char * *mime*, PKI_CRED * *cred*, HSM * *hsm*)

1.10.1.5 PKI_MEM* PKI_X509_CERT_put_mem (PKI_X509_CERT * *x*, PKI_DATA_FORMAT *format*, PKI_MEM ** *pki_mem*, PKI_CRED * *cred*, HSM * *hsm*)

1.10.1.6 int PKI_X509_CERT_put_url (PKI_X509_CERT * *x*, PKI_DATA_FORMAT *format*, URL * *url*, char * *mime*, PKI_CRED * *cred*, HSM * *hsm*)

1.10.1.7 PKI_X509_CERT_STACK* PKI_X509_CERT_STACK_get (char * *url_s*, PKI_CRED * *cred*, HSM * *hsm*)

Retrieve a stack of certificates from a URL (char *).

Downloads a stack of certificates from a given URL ([file://](#), [http://](#), [ldap://...](#)) passed as a (char *).

The returned data is a pointer to a PKI_X509_CERT_STACK data structure in case of success or NULL if any error occurred. If only the first object is required from the URL, use the [PKI_X509_CERT_get_url\(\)](#) function instead.

1.10.1.8 PKI_X509_CERT_STACK* PKI_X509_CERT_STACK_get_mem (PKI_MEM * *mem*, PKI_CRED * *cred*)

1.10.1.9 `PKI_X509_CERT_STACK*` `PKI_X509_CERT_STACK_get_url` (`URL *` *url*, `PKI_CRED *` *cred*, `HSM *` *hsm*)

Retrieve a stack of certificates from a URL (`URL *`) pointer.

Downloads a stack of certificates from a given URL (`file://`, `http://`, `ldap://...`) passed as a (`URL *`). To generate a (`URL *`) from a (`char *`) use `URL_new()`.

The returned data is a pointer to a `PKI_X509_CERT_STACK` data structure in case of success or `NULL` if any error occurred. If only the first object is required from the URL, use the `PKI_X509_CERT_get_url()` function instead.

1.10.1.10 `int` `PKI_X509_CERT_STACK_put` (`PKI_X509_CERT_STACK *` *sk*, `PKI_DATA_FORMAT` *format*, `char *` *url_s*, `char *` *mime*, `PKI_CRED *` *cred*, `HSM *` *hsm*)

1.10.1.11 `PKI_MEM*` `PKI_X509_CERT_STACK_put_mem` (`PKI_X509_CERT_STACK *` *sk*, `PKI_DATA_FORMAT` *format*, `PKI_MEM **` *pki_mem*, `PKI_CRED *` *cred*, `HSM *` *hsm*)

1.10.1.12 `int` `PKI_X509_CERT_STACK_put_url` (`PKI_X509_CERT_STACK *` *sk*, `PKI_DATA_FORMAT` *format*, `URL *` *url*, `char *` *mime*, `PKI_CRED *` *cred*, `HSM *` *hsm*)

1.11 src/io/pki_x509_crl_io.c File Reference

Include dependency graph for `pki_x509_crl_io.c`:



Functions

- `PKI_X509_CRL *` `PKI_X509_CRL_get` (`char *` *url_s*, `PKI_CRED *` *cred*, `HSM *` *hsm*)
Retrieve a CRL from a URL.
- `PKI_X509_CRL *` `PKI_X509_CRL_get_mem` (`PKI_MEM *` *mem*, `PKI_CRED *` *cred*, `HSM *` *hsm*)
- `PKI_X509_CRL *` `PKI_X509_CRL_get_url` (`URL *` *url*, `PKI_CRED *` *cred*, `HSM *` *hsm*)
Retrieve a CRL from a URL pointer.
- `int` `PKI_X509_CRL_put` (`PKI_X509_CRL *` *crl*, `PKI_DATA_FORMAT` *format*, `char *` *url_s*, `PKI_CRED *` *cred*, `HSM *` *hsm*)
- `PKI_MEM *` `PKI_X509_CRL_put_mem` (`PKI_X509_CRL *` *crl*, `PKI_DATA_FORMAT` *format*, `PKI_MEM **` *mem*, `PKI_CRED *` *cred*, `HSM *` *hsm*)
- `int` `PKI_X509_CRL_put_url` (`PKI_X509_CRL *` *crl*, `PKI_DATA_FORMAT` *format*, `URL *` *url*, `PKI_CRED *` *cred*, `HSM *` *hsm*)
- `PKI_X509_CRL_STACK *` `PKI_X509_CRL_STACK_get` (`char *` *url_s*, `PKI_CRED *` *cred*, `HSM *` *hsm*)
*Retrieve a stack of CRLs from a URL (char *).*
- `PKI_X509_CRL_STACK *` `PKI_X509_CRL_STACK_get_mem` (`PKI_MEM *` *mem*, `PKI_CRED *` *cred*)
- `PKI_X509_CRL_STACK *` `PKI_X509_CRL_STACK_get_url` (`URL *` *url*, `PKI_CRED *` *cred*, `HSM *` *hsm*)

*Retrieve a stack of CRLs from a URL (URL *) pointer.*

- int [PKI_X509_CRL_STACK_put](#) (PKI_X509_CRL_STACK *sk, PKI_DATA_FORMAT format, char *url_s, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_CRL_STACK_put_mem](#) (PKI_X509_CRL_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_CRL_STACK_put_url](#) (PKI_X509_CRL_STACK *sk, PKI_DATA_FORMAT format, URL *url, PKI_CRED *cred, HSM *hsm)

1.11.1 Function Documentation

1.11.1.1 PKI_X509_CRL* PKI_X509_CRL_get (char * url_s, PKI_CRED * cred, HSM * hsm)

Retrieve a CRL from a URL.

Downloads a CRL from a given URL ([file:///](#), [http://](#), [ldap://...](#)) in (char *) format. The returned data is of type PKI_X509_CRL in case of success or NULL if any error occurred. If multiple objects are returned from the URL, only the first one is returned. Use [PKI_X509_CRL_STACK_get\(\)](#) function to retrieve a PKI_X509_CERT_STACK * object.

1.11.1.2 PKI_X509_CRL* PKI_X509_CRL_get_mem (PKI_MEM * mem, PKI_CRED * cred, HSM * hsm)

1.11.1.3 PKI_X509_CRL* PKI_X509_CRL_get_url (URL * url, PKI_CRED * cred, HSM * hsm)

Retrieve a CRL from a URL pointer.

Downloads a CRL from a given URL ([file:///](#), [http://](#), [ldap://...](#)) in (URL *) format. To generate a URL * from a char * use [URL_new\(\)](#). The returned data is of type PKI_X509_CRL * in case of success or NULL if any error occurred. If multiple objects are returned from the URL, only the first one is returned. Use [PKI_X509_CRL_get_url\(\)](#) function to retrieve a PKI_X509_CRL_STACK * object.

1.11.1.4 int PKI_X509_CRL_put (PKI_X509_CRL * crl, PKI_DATA_FORMAT format, char * url_s, PKI_CRED * cred, HSM * hsm)

1.11.1.5 PKI_MEM* PKI_X509_CRL_put_mem (PKI_X509_CRL * crl, PKI_DATA_FORMAT format, PKI_MEM ** mem, PKI_CRED * cred, HSM * hsm)

1.11.1.6 int PKI_X509_CRL_put_url (PKI_X509_CRL * crl, PKI_DATA_FORMAT format, URL * url, PKI_CRED * cred, HSM * hsm)

1.11.1.7 PKI_X509_CRL_STACK* PKI_X509_CRL_STACK_get (char * url_s, PKI_CRED * cred, HSM * hsm)

Retrieve a stack of CRLs from a URL (char *).

Downloads a stack of CRLs from a given URL ([file:///](#), [http://](#), [ldap://...](#)) passed as a (char *).

The returned data is a pointer to a PKI_X509_CRL_STACK data structure in case of success or NULL if any error occurred. If only the first object is required from the URL, use the [PKI_X509_CRL_get_url\(\)](#) function instead.

1.11.1.8 `PKI_X509_CRL_STACK* PKI_X509_CRL_STACK_get_mem (PKI_MEM *mem, PKI_CRED *cred)`

1.11.1.9 `PKI_X509_CRL_STACK* PKI_X509_CRL_STACK_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`

Retrieve a stack of CRLs from a URL (URL *) pointer.

Downloads a stack of CRLs from a given URL (`file://`, `http://`, `ldap://...`) passed as a (URL *). To generate a (URL *) from a (char *) use `URL_new()`.

The returned data is a pointer to a `PKI_X509_CRL_STACK` data structure in case of success or NULL if any error occurred. If only the first object is required from the URL, use the `PKI_X509_CRL_get_url()` function instead.

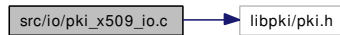
1.11.1.10 `int PKI_X509_CRL_STACK_put (PKI_X509_CRL_STACK *sk, PKI_DATA_FORMAT format, char *url_s, PKI_CRED *cred, HSM *hsm)`

1.11.1.11 `PKI_MEM* PKI_X509_CRL_STACK_put_mem (PKI_X509_CRL_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`

1.11.1.12 `int PKI_X509_CRL_STACK_put_url (PKI_X509_CRL_STACK *sk, PKI_DATA_FORMAT format, URL *url, PKI_CRED *cred, HSM *hsm)`

1.12 src/io/pki_x509_io.c File Reference

Include dependency graph for `pki_x509_io.c`:



Functions

- `void * PKI_get_value (char *url_s, PKI_DATATYPE type, PKI_CRED *cred, HSM *hsm)`
*Returns the PKI_X509_XXX_VALUE * from the passed URL.*
- `PKI_X509 * PKI_X509_get (char *url_s, PKI_DATATYPE type, PKI_CRED *cred, HSM *hsm)`
Retrieve an X509 object from a URL.
- `PKI_X509 * PKI_X509_get_url (URL *url, PKI_DATATYPE type, PKI_CRED *cred, HSM *hsm)`
Retrieve an X509 object from a URL pointer.
- `int PKI_X509_put (PKI_X509 *x, PKI_DATA_FORMAT format, char *url_string, const char *mime, PKI_CRED *cred, HSM *hsm)`
Puts a PKI_X509 object into the passed url (string).
- `int PKI_X509_put_url (PKI_X509 *x, PKI_DATA_FORMAT format, URL *url, const char *mime, PKI_CRED *cred, HSM *hsm)`
Put a PKI_X509 object to the specified URL.

- int [PKI_X509_put_value](#) (void *x, PKI_DATATYPE type, PKI_DATA_FORMAT format, char *url_string, const char *mime, PKI_CRED *cred, HSM *hsm)
Writes the PKI_X509_XXX_VALUE to the passed url.
- PKI_X509_STACK * [PKI_X509_STACK_get](#) (char *url_s, PKI_DATATYPE type, PKI_CRED *cred, HSM *hsm)
*Retrieve a stack of X509 objects from a URL (char *).*
- PKI_X509_CERT_STACK * [PKI_X509_STACK_get_url](#) (URL *url, PKI_DATATYPE type, PKI_CRED *cred, HSM *hsm)
*Retrieve a stack of X509 objects from a URL (URL *) pointer.*
- int [PKI_X509_STACK_put](#) (PKI_X509_STACK *sk, PKI_DATA_FORMAT format, char *url_string, const char *mime, PKI_CRED *cred, HSM *hsm)
Puts a stack of PKI_X509 objects to the url passed as a string.
- int [PKI_X509_STACK_put_url](#) (PKI_X509_STACK *sk, PKI_DATA_FORMAT format, URL *url, const char *mime, PKI_CRED *cred, HSM *hsm)
Puts a stack of PKI_X509 objects to a specified URL.

1.12.1 Function Documentation

1.12.1.1 void* [PKI_get_value](#) (char * url_s, PKI_DATATYPE type, PKI_CRED * cred, HSM * hsm)

Returns the PKI_X509_XXX_VALUE * from the passed URL.

1.12.1.2 PKI_X509* [PKI_X509_get](#) (char * url_s, PKI_DATATYPE type, PKI_CRED * cred, HSM * hsm)

Retrieve an X509 object from a URL.

Downloads an X509 object from a given URL ([file:///](#), [http://](#), [ldap://...](#)) in (char *) format. The returned data is of type PKI_X509 in case of success or NULL if any error occurred. If multiple objects are returned from the URL, only the first one is returned. Use [PKI_X509_STACK_get\(\)](#) function to retrieve a PKI_X509_STACK * object.

1.12.1.3 PKI_X509* [PKI_X509_get_url](#) (URL * url, PKI_DATATYPE type, PKI_CRED * cred, HSM * hsm)

Retrieve an X509 object from a URL pointer.

Downloads a certificate from a given URL ([file:///](#), [http://](#), [ldap://...](#)) in (URL *) format. To generate a URL * from a char * use [URL_new\(\)](#). The returned data is of type PKI_X509 in case of success or NULL if any error occurred. If multiple objects are returned from the URL, only the first one is returned. Use [PKI_X509_STACK_get_url\(\)](#) function to retrieve a PKI_X509_STACK * object.

1.12.1.4 int [PKI_X509_put](#) (PKI_X509 * x, PKI_DATA_FORMAT format, char * url_string, const char * mime, PKI_CRED * cred, HSM * hsm)

Puts a PKI_X509 object into the passed url (string).

1.12.1.5 `int PKI_X509_put_url (PKI_X509 * x, PKI_DATA_FORMAT format, URL * url, const char * mime, PKI_CRED * cred, HSM * hsm)`

Put a PKI_X509 object to the specified URL.

1.12.1.6 `int PKI_X509_put_value (void * x, PKI_DATATYPE type, PKI_DATA_FORMAT format, char * url_string, const char * mime, PKI_CRED * cred, HSM * hsm)`

Writes the PKI_X509_XXX_VALUE to the passed url.

1.12.1.7 `PKI_X509_STACK* PKI_X509_STACK_get (char * url_s, PKI_DATATYPE type, PKI_CRED * cred, HSM * hsm)`

Retrieve a stack of X509 objects from a URL (char *).

Downloads a stack of X509 objects from a given URL ([file:///](#), [http:///](#), [ldap:///](#)...) passed as a (char *).

The returned data is a pointer to a PKI_X509_STACK data structure in case of success or NULL if any error occurred. If only the first object is required from the URL, use the [PKI_X509_get_url\(\)](#) function instead.

1.12.1.8 `PKI_X509_CERT_STACK* PKI_X509_STACK_get_url (URL * url, PKI_DATATYPE type, PKI_CRED * cred, HSM * hsm)`

Retrieve a stack of X509 objects from a URL (URL *) pointer.

Downloads a stack of certificates from a given URL ([file:///](#), [http:///](#), [ldap:///](#)...) passed as a (URL *). To generate a (URL *) from a (char *) use [URL_new\(\)](#).

The returned data is a pointer to a PKI_X509_STACK data structure in case of success or NULL if any error occurred. If only the first object is required from the URL, use the [PKI_X509_get_url\(\)](#) function instead.

1.12.1.9 `int PKI_X509_STACK_put (PKI_X509_STACK * sk, PKI_DATA_FORMAT format, char * url_string, const char * mime, PKI_CRED * cred, HSM * hsm)`

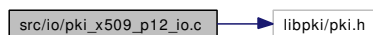
Puts a stack of PKI_X509 objects to the url passed as a string.

1.12.1.10 `int PKI_X509_STACK_put_url (PKI_X509_STACK * sk, PKI_DATA_FORMAT format, URL * url, const char * mime, PKI_CRED * cred, HSM * hsm)`

Puts a stack of PKI_X509 objects to a specified URL.

1.13 src/io/pki_x509_p12_io.c File Reference

Include dependency graph for pki_x509_p12_io.c:



Functions

- `PKI_X509_PKCS12 * PKI_X509_PKCS12_get (char *url_s, PKI_CRED *cred, HSM *hsm)`

- PKI_X509_PKCS12 * [PKI_X509_PKCS12_get_mem](#) (PKI_MEM *mem, PKI_CRED *cred)
- PKI_X509_PKCS12 * [PKI_X509_PKCS12_get_url](#) (URL *url, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_PKCS12_put](#) (PKI_X509_PKCS12 *p12, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_PKCS12_put_mem](#) (PKI_X509_PKCS12 *p12, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)

Puts a PKI_X509_PKCS12 in a PKI_MEM structure.

- int [PKI_X509_PKCS12_put_url](#) (PKI_X509_PKCS12 *p12, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_X509_PKCS12_STACK * [PKI_X509_PKCS12_STACK_get](#) (char *url_s, PKI_CRED *cred, HSM *hsm)
- PKI_X509_PKCS12_STACK * [PKI_X509_PKCS12_STACK_get_mem](#) (PKI_MEM *mem, PKI_CRED *cred)
- PKI_X509_PKCS12_STACK * [PKI_X509_PKCS12_STACK_get_url](#) (URL *url, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_PKCS12_STACK_put](#) (PKI_X509_PKCS12_STACK *sk, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_PKCS12_STACK_put_mem](#) (PKI_X509_PKCS12_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_PKCS12_STACK_put_url](#) (PKI_X509_PKCS12_STACK *sk, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)

1.13.1 Function Documentation

1.13.1.1 PKI_X509_PKCS12* [PKI_X509_PKCS12_get](#) (char * *url_s*, PKI_CRED * *cred*, HSM * *hsm*)

1.13.1.2 PKI_X509_PKCS12* [PKI_X509_PKCS12_get_mem](#) (PKI_MEM * *mem*, PKI_CRED * *cred*)

1.13.1.3 PKI_X509_PKCS12* [PKI_X509_PKCS12_get_url](#) (URL * *url*, PKI_CRED * *cred*, HSM * *hsm*)

1.13.1.4 int [PKI_X509_PKCS12_put](#) (PKI_X509_PKCS12 * *p12*, PKI_DATA_FORMAT *format*, char * *url_s*, char * *mime*, PKI_CRED * *cred*, HSM * *hsm*)

1.13.1.5 PKI_MEM* [PKI_X509_PKCS12_put_mem](#) (PKI_X509_PKCS12 * *p12*, PKI_DATA_FORMAT *format*, PKI_MEM ** *pki_mem*, PKI_CRED * *cred*, HSM * *hsm*)

Puts a PKI_X509_PKCS12 in a PKI_MEM structure.

1.13.1.6 int [PKI_X509_PKCS12_put_url](#) (PKI_X509_PKCS12 * *p12*, PKI_DATA_FORMAT *format*, URL * *url*, char * *mime*, PKI_CRED * *cred*, HSM * *hsm*)

1.13.1.7 PKI_X509_PKCS12_STACK* [PKI_X509_PKCS12_STACK_get](#) (char * *url_s*, PKI_CRED * *cred*, HSM * *hsm*)

1.13.1.8 `PKI_X509_PKCS12_STACK* PKI_X509_PKCS12_STACK_get_mem (PKI_MEM * mem, PKI_CRED * cred)`

1.13.1.9 `PKI_X509_PKCS12_STACK* PKI_X509_PKCS12_STACK_get_url (URL * url, PKI_CRED * cred, HSM * hsm)`

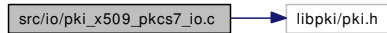
1.13.1.10 `int PKI_X509_PKCS12_STACK_put (PKI_X509_PKCS12_STACK * sk, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.13.1.11 `PKI_MEM* PKI_X509_PKCS12_STACK_put_mem (PKI_X509_PKCS12_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.13.1.12 `int PKI_X509_PKCS12_STACK_put_url (PKI_X509_PKCS12_STACK * sk, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.14 src/io/pki_x509_pkcs7_io.c File Reference

Include dependency graph for pki_x509_pkcs7_io.c:



Functions

- `PKI_X509_PKCS7 * PKI_X509_PKCS7_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_PKCS7 * PKI_X509_PKCS7_get_mem (PKI_MEM *mem, PKI_CRED *cred)`
- `PKI_X509_PKCS7 * PKI_X509_PKCS7_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_PKCS7_put (PKI_X509_PKCS7 *p7, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_PKCS7_put_mem (PKI_X509_PKCS7 *p7, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_PKCS7_put_url (PKI_X509_PKCS7 *p7, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_PKCS7_STACK * PKI_X509_PKCS7_STACK_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_PKCS7_STACK * PKI_X509_PKCS7_STACK_get_mem (PKI_MEM *mem, PKI_CRED *cred)`
- `PKI_X509_PKCS7_STACK * PKI_X509_PKCS7_STACK_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_PKCS7_STACK_put (PKI_X509_PKCS7_STACK *sk, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_PKCS7_STACK_put_mem (PKI_X509_PKCS7_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_PKCS7_STACK_put_url (PKI_X509_PKCS7_STACK *sk, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)`

1.14.1 Function Documentation

1.14.1.1 `PKI_X509_PKCS7* PKI_X509_PKCS7_get (char * url_s, PKI_CRED * cred, HSM * hsm)`

1.14.1.2 `PKI_X509_PKCS7* PKI_X509_PKCS7_get_mem (PKI_MEM * mem, PKI_CRED * cred)`

1.14.1.3 `PKI_X509_PKCS7* PKI_X509_PKCS7_get_url (URL * url, PKI_CRED * cred, HSM * hsm)`

1.14.1.4 `int PKI_X509_PKCS7_put (PKI_X509_PKCS7 * p7, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.14.1.5 `PKI_MEM* PKI_X509_PKCS7_put_mem (PKI_X509_PKCS7 * p7, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.14.1.6 `int PKI_X509_PKCS7_put_url (PKI_X509_PKCS7 * p7, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.14.1.7 `PKI_X509_PKCS7_STACK* PKI_X509_PKCS7_STACK_get (char * url_s, PKI_CRED * cred, HSM * hsm)`

1.14.1.8 `PKI_X509_PKCS7_STACK* PKI_X509_PKCS7_STACK_get_mem (PKI_MEM * mem, PKI_CRED * cred)`

1.14.1.9 `PKI_X509_PKCS7_STACK* PKI_X509_PKCS7_STACK_get_url (URL * url, PKI_CRED * cred, HSM * hsm)`

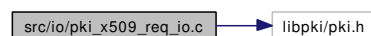
1.14.1.10 `int PKI_X509_PKCS7_STACK_put (PKI_X509_PKCS7_STACK * sk, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.14.1.11 `PKI_MEM* PKI_X509_PKCS7_STACK_put_mem (PKI_X509_PKCS7_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.14.1.12 `int PKI_X509_PKCS7_STACK_put_url (PKI_X509_PKCS7_STACK * sk, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.15 src/io/pki_x509_req_io.c File Reference

Include dependency graph for pki_x509_req_io.c:



Functions

- PKI_X509_REQ * [PKI_X509_REQ_get](#) (char *url_s, PKI_CRED *cred, HSM *hsm)
- PKI_X509_REQ * [PKI_X509_REQ_get_mem](#) (PKI_MEM *mem, PKI_CRED *cred)
- PKI_X509_REQ * [PKI_X509_REQ_get_url](#) (URL *url, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_REQ_put](#) (PKI_X509_REQ *req, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_REQ_put_mem](#) (PKI_X509_REQ *r, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_REQ_put_url](#) (PKI_X509_REQ *req, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_X509_REQ_STACK * [PKI_X509_REQ_STACK_get](#) (char *url_s, PKI_CRED *cred, HSM *hsm)
- PKI_X509_REQ_STACK * [PKI_X509_REQ_STACK_get_mem](#) (PKI_MEM *mem, PKI_CRED *cred)
- PKI_X509_REQ_STACK * [PKI_X509_REQ_STACK_get_url](#) (URL *url, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_REQ_STACK_put](#) (PKI_X509_REQ_STACK *sk, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_REQ_STACK_put_mem](#) (PKI_X509_REQ_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_REQ_STACK_put_url](#) (PKI_X509_REQ_STACK *sk, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)

1.15.1 Function Documentation

1.15.1.1 PKI_X509_REQ* [PKI_X509_REQ_get](#) (char * url_s, PKI_CRED * cred, HSM * hsm)

1.15.1.2 PKI_X509_REQ* [PKI_X509_REQ_get_mem](#) (PKI_MEM * mem, PKI_CRED * cred)

1.15.1.3 PKI_X509_REQ* [PKI_X509_REQ_get_url](#) (URL * url, PKI_CRED * cred, HSM * hsm)

1.15.1.4 int [PKI_X509_REQ_put](#) (PKI_X509_REQ * req, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)

1.15.1.5 PKI_MEM* [PKI_X509_REQ_put_mem](#) (PKI_X509_REQ * r, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)

1.15.1.6 int [PKI_X509_REQ_put_url](#) (PKI_X509_REQ * req, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)

1.15.1.7 PKI_X509_REQ_STACK* [PKI_X509_REQ_STACK_get](#) (char * url_s, PKI_CRED * cred, HSM * hsm)

1.15.1.8 PKI_X509_REQ_STACK* [PKI_X509_REQ_STACK_get_mem](#) (PKI_MEM * mem, PKI_CRED * cred)

1.15.1.9 `PKI_X509_REQ_STACK * PKI_X509_REQ_STACK_get_url (URL * url, PKI_CRED * cred, HSM * hsm)`

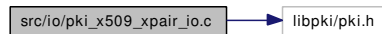
1.15.1.10 `int PKI_X509_REQ_STACK_put (PKI_X509_REQ_STACK * sk, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.15.1.11 `PKI_MEM * PKI_X509_REQ_STACK_put_mem (PKI_X509_REQ_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.15.1.12 `int PKI_X509_REQ_STACK_put_url (PKI_X509_REQ_STACK * sk, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.16 src/io/pki_x509_xpair_io.c File Reference

Include dependency graph for pki_x509_xpair_io.c:



Functions

- `PKI_X509_XPAIR * PKI_X509_XPAIR_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
Retrieve a Cross Cert Pair from a URL.
- `PKI_X509_XPAIR * PKI_X509_XPAIR_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
Retrieve a cross certificate pair from a URL pointer.
- `int PKI_X509_XPAIR_put (PKI_X509_XPAIR *x, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_XPAIR_put_mem (PKI_X509_XPAIR *x, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_XPAIR_STACK * PKI_X509_XPAIR_STACK_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
*Retrieve a stack of cross cert pair from a URL (char *).*
- `PKI_X509_XPAIR_STACK * PKI_X509_XPAIR_STACK_get_mem (PKI_MEM *mem, PKI_CRED *cred)`
- `PKI_X509_XPAIR_STACK * PKI_X509_XPAIR_STACK_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
*Retrieve a stack of cross cert pair from a URL (URL *) pointer.*
- `int PKI_X509_XPAIR_STACK_put (PKI_X509_XPAIR_STACK *sk, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_XPAIR_STACK_put_mem (PKI_X509_XPAIR_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_XPAIR_STACK_put_url (PKI_X509_XPAIR_STACK *sk, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)`
- `int PKI_XPAIR_print (BIO *bio, PKI_XPAIR *xp_val)`

1.16.1 Function Documentation

1.16.1.1 **PKI_X509_XPAIR*** **PKI_X509_XPAIR_get** (*char * url_s*, *PKI_CRED * cred*, *HSM * hsm*)

Retrieve a Cross Cert Pair from a URL.

Downloads a XPAIR from a given URL ([file://](#), [http://](#), [ldap://...](#)) in (*char **) format. The returned data is of type *PKI_X509_XPAIR* in case of success or NULL if any error occurred. If multiple objects are returned from the URL, only the first one is returned. Use [PKI_X509_XPAIR_STACK_get\(\)](#) function to retrieve a *PKI_X509_XPAIR_STACK ** object.

1.16.1.2 **PKI_X509_XPAIR*** **PKI_X509_XPAIR_get_url** (*URL * url*, *PKI_CRED * cred*, *HSM * hsm*)

Retrieve a cross certificate pair from a URL pointer.

Downloads a XPAIR from a given URL ([file://](#), [http://](#), [ldap://...](#)) in (*URL **) format. To generate a *URL ** from a *char ** use [URL_new\(\)](#). The returned data is of type *PKI_X509_XPAIR* in case of success or NULL if any error occurred. If multiple objects are returned from the URL, only the first one is returned. Use [PKI_X509_XPAIR_STACK_get_url\(\)](#) function to retrieve a *PKI_X509_XPAIR_STACK ** object.

1.16.1.3 **int** **PKI_X509_XPAIR_put** (*PKI_X509_XPAIR * x*, *PKI_DATA_FORMAT format*, *char * url_s*, *char * mime*, *PKI_CRED * cred*, *HSM * hsm*)

1.16.1.4 **PKI_MEM*** **PKI_X509_XPAIR_put_mem** (*PKI_X509_XPAIR * x*, *PKI_DATA_FORMAT format*, *PKI_MEM ** pki_mem*, *PKI_CRED * cred*, *HSM * hsm*)

1.16.1.5 **PKI_X509_XPAIR_STACK*** **PKI_X509_XPAIR_STACK_get** (*char * url_s*, *PKI_CRED * cred*, *HSM * hsm*)

Retrieve a stack of cross cert pair from a URL (*char **).

Downloads a stack of certificates from a given URL ([file://](#), [http://](#), [ldap://...](#)) passed as a (*char **).

The returned data is a pointer to a *PKI_X509_XPAIR_STACK* data structure in case of success or NULL if any error occurred. If only the first object is required from the URL, use the [PKI_X509_XPAIR_get_url\(\)](#) function instead.

1.16.1.6 **PKI_X509_XPAIR_STACK*** **PKI_X509_XPAIR_STACK_get_mem** (*PKI_MEM * mem*, *PKI_CRED * cred*)

1.16.1.7 **PKI_X509_XPAIR_STACK*** **PKI_X509_XPAIR_STACK_get_url** (*URL * url*, *PKI_CRED * cred*, *HSM * hsm*)

Retrieve a stack of cross cert pair from a URL (*URL **) pointer.

Downloads a stack of XPAIR from a given URL ([file://](#), [http://](#), [ldap://...](#)) passed as a (*URL **). To generate a (*URL **) from a (*char **) use [URL_new\(\)](#).

The returned data is a pointer to a *PKI_X509_XPAIR_STACK* data structure in case of success or NULL if any error occurred. If only the first object is required from the URL, use the [PKI_X509_XPAIR_get_url\(\)](#) function instead.

1.16.1.8 `int PKI_X509_XPAIR_STACK_put (PKI_X509_XPAIR_STACK * sk, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.16.1.9 `PKI_MEM* PKI_X509_XPAIR_STACK_put_mem (PKI_X509_XPAIR_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.16.1.10 `int PKI_X509_XPAIR_STACK_put_url (PKI_X509_XPAIR_STACK * sk, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.16.1.11 `int PKI_XPAIR_print (BIO * bio, PKI_XPAIR * xp_val)`

1.17 src/net/http_s.c File Reference

Include dependency graph for http_s.c:



Defines

- `#define` [BUFF_MAX_SIZE](#) 2048

Functions

- `PKI_HTTP *` [__pki_http_get_sock_s](#) (int *fd*, PKI_SSL **ssl*, int *timeout*, size_t *max_size*)
- `int` [__url_get_data_http_s](#) (char **url_s*, char **data*, size_t *data_size*, char **cont_type*, int *method*, int *timeout*, size_t *max_size*, PKI_MEM_STACK ***sk*, PKI_SSL **ssl*)
- `int` [__url_get_data_http_s_url](#) (URL **url*, char **data*, size_t *data_size*, char **content_type*, int *method*, int *timeout*, size_t *max_size*, PKI_MEM_STACK ***sk*, PKI_SSL **ssl*)
- `void` [PKI_HTTP_free](#) (PKI_HTTP **rv*)

Frees the memory associated with a PKI_HTTP data structure.

- `char *` [PKI_HTTP_get_header](#) (PKI_HTTP **http*, char **header*)

Returns a PKI_HTTP from the content of a PKI_MEM.

- `char *` [PKI_HTTP_get_header_txt](#) (char **orig_data*, char **header*)

*Returns a PKI_HTTP from the content of a char *.*

- `PKI_HTTP *` [PKI_HTTP_get_sock](#) (int *fd*, int *timeout*, size_t *max_size*)

Returns a parsed PKI_HTTP from a socket (read).

- `PKI_HTTP *` [PKI_HTTP_new](#) (void)

Allocates the memory for a new PKI_HTTP data structure.

- `PKI_HTTP *` [PKI_HTTPS_get_sock](#) (PKI_SSL **ssl*, int *timeout*, size_t *max_size*)

Internal function to get data from a PKI_SSL Returns a parsed PKI_HTTP from a PKI_SSL socket (read).

- int [URL_get_data_http_s](#) (char *url_s, int timeout, size_t max_size, PKI_MEM_STACK **ret, PKI_SSL *ssl)

Returns the data from an HTTP source by using the GET command.

- int [URL_get_data_http_s_url](#) (URL *url, int timeout, size_t max_size, PKI_MEM_STACK **sk, PKI_SSL *ssl)

Returns data from a socket (see [PKI_NET_get_data\(\)](#) for details) Sends a HTTP message to a URL and retrieve the response.

- int [URL_post_data_http_s](#) (char *url_s, char *data, size_t size, char *content_type, int timeout, size_t max_size, PKI_MEM_STACK **ret_sk, PKI_SSL *ssl)

Use POST method to transfer data via HTTP. If a pointer to a PKI_MEM_STACK (eg., &stack) is provided, the returned data is added to it.

- int [URL_post_data_http_s_url](#) (URL *url, char *data, size_t size, char *content_type, int timeout, size_t max_size, PKI_MEM_STACK **ret_sk, PKI_SSL *ssl)

Posts data to an HTTP URL and returns the response data.

- int [URL_put_data_http_s](#) (char *url_s, char *data, size_t size, char *content_type, int timeout, size_t max_size, PKI_MEM_STACK **ret_sk, PKI_SSL *ssl)

Sends a HTTP message to a URL and retrieve the response over PKI_SSL Returns the data from an HTTPS source by using the GET command Posts data to an HTTP/HTTPS url.

1.17.1 Define Documentation

1.17.1.1 #define BUFF_MAX_SIZE 2048

1.17.2 Function Documentation

1.17.2.1 PKI_HTTP* __pki_http_get_sock_s (int fd, PKI_SSL *ssl, int timeout, size_t max_size)

1.17.2.2 int __url_get_data_http_s (char *url_s, char *data, size_t data_size, char *cont_type, int method, int timeout, size_t max_size, PKI_MEM_STACK **sk, PKI_SSL *ssl)

1.17.2.3 int __url_get_data_http_s_url (URL *url, char *data, size_t data_size, char *content_type, int method, int timeout, size_t max_size, PKI_MEM_STACK **sk, PKI_SSL *ssl)

1.17.2.4 void PKI_HTTP_free (PKI_HTTP *rv)

Frees the memory associated with a PKI_HTTP data structure.

1.17.2.5 char* PKI_HTTP_get_header (PKI_HTTP *http, char *header)

Returns a PKI_HTTP from the content of a PKI_MEM.

1.17.2.6 char* PKI_HTTP_get_header_txt (char *orig_data, char *header)

Returns a PKI_HTTP from the content of a char *.

1.17.2.7 PKI_HTTP* PKI_HTTP_get_sock (int *fd*, int *timeout*, size_t *max_size*)

Returns a parsed PKI_HTTP from a socket (read).

1.17.2.8 PKI_HTTP* PKI_HTTP_new (void)

Allocates the memory for a new PKI_HTTP data structure.

1.17.2.9 PKI_HTTP* PKI_HTTPS_get_sock (PKI_SSL * *ssl*, int *timeout*, size_t *max_size*)

Internal function to get data from a PKI_SSL Returns a parsed PKI_HTTP from a PKI_SSL socket (read).

1.17.2.10 int URL_get_data_http_s (char * *url_s*, int *timeout*, size_t *max_size*, PKI_MEM_STACK ** *ret*, PKI_SSL * *ssl*)

Returns the data from an HTTP source by using the GET command.

1.17.2.11 int URL_get_data_http_s_url (URL * *url*, int *timeout*, size_t *max_size*, PKI_MEM_STACK ** *sk*, PKI_SSL * *ssl*)

Returns data from a socket (see [PKI_NET_get_data\(\)](#) for details) Sends a HTTP message to a URL and retrieve the response.

Sends (POST/GET) data to a url and (if a pointer to a mem stack is provided) returns the received response. PKI_ERR is returned in case of error, otherwise PKI_OK is returned.

1.17.2.12 int URL_post_data_http_s (char * *url_s*, char * *data*, size_t *size*, char * *content_type*, int *timeout*, size_t *max_size*, PKI_MEM_STACK ** *ret_sk*, PKI_SSL * *ssl*)

Use POST method to transfer data via HTTP. If a pointer to a PKI_MEM_STACK (eg., &stack) is provided, the returned data is added to it.

1.17.2.13 int URL_post_data_http_s_url (URL * *url*, char * *data*, size_t *size*, char * *content_type*, int *timeout*, size_t *max_size*, PKI_MEM_STACK ** *ret_sk*, PKI_SSL * *ssl*)

Posts data to an HTTP URL and returns the response data.

1.17.2.14 int URL_put_data_http_s (char * *url_s*, char * *data*, size_t *size*, char * *content_type*, int *timeout*, size_t *max_size*, PKI_MEM_STACK ** *ret_sk*, PKI_SSL * *ssl*)

Sends a HTTP message to a URL and retrieve the response over PKI_SSL Returns the data from an HTTPS source by using the GET command Posts data to an HTTP/HTTPS url.

1.18 src/net/ldap.c File Reference

Include dependency graph for ldap.c:



1.19 src/net/mysql.c File Reference

Include dependency graph for mysql.c:



Functions

- char * [parse_url_dbname](#) (URL *url)
- char * [parse_url_put_query](#) (URL *url, PKI_MEM *data)
- char * [parse_url_query](#) (URL *url)
- char * [parse_url_table](#) (URL *url)
- PKI_MEM_STACK * [URL_get_data_mysql](#) (char *url_s, ssize_t size)
- PKI_MEM_STACK * [URL_get_data_mysql_url](#) (URL *url, ssize_t size)
- int [URL_put_data_mysql](#) (char *url_s, PKI_MEM *data)
- int [URL_put_data_mysql_url](#) (URL *url, PKI_MEM *data)

1.19.1 Function Documentation

1.19.1.1 char* [parse_url_dbname](#) (URL * *url*)

1.19.1.2 char* [parse_url_put_query](#) (URL * *url*, PKI_MEM * *data*)

1.19.1.3 char* [parse_url_query](#) (URL * *url*)

1.19.1.4 char* [parse_url_table](#) (URL * *url*)

1.19.1.5 PKI_MEM_STACK* [URL_get_data_mysql](#) (char * *url_s*, ssize_t *size*)

1.19.1.6 PKI_MEM_STACK* [URL_get_data_mysql_url](#) (URL * *url*, ssize_t *size*)

1.19.1.7 int [URL_put_data_mysql](#) (char * *url_s*, PKI_MEM * *data*)

1.19.1.8 int [URL_put_data_mysql_url](#) (URL * *url*, PKI_MEM * *data*)

1.20 src/net/pg.c File Reference

Include dependency graph for pg.c:



Functions

- char * [pg_parse_url_dbname](#) (URL *url)
- char * [pg_parse_url_put_query](#) (URL *url, PKI_MEM *data)
- char * [pg_parse_url_query](#) (URL *url)
- char * [pg_parse_url_table](#) (URL *url)
- PKI_MEM_STACK * [URL_get_data_pg](#) (char *url_s, ssize_t size)
- PKI_MEM_STACK * [URL_get_data_pg_url](#) (URL *url, ssize_t size)
- int [URL_put_data_pg](#) (char *url_s, PKI_MEM *data)
- int [URL_put_data_pg_url](#) (URL *url, PKI_MEM *data)

1.20.1 Function Documentation

1.20.1.1 char* [pg_parse_url_dbname](#) (URL * *url*)

1.20.1.2 char* [pg_parse_url_put_query](#) (URL * *url*, PKI_MEM * *data*)

1.20.1.3 char* [pg_parse_url_query](#) (URL * *url*)

1.20.1.4 char* [pg_parse_url_table](#) (URL * *url*)

1.20.1.5 PKI_MEM_STACK* [URL_get_data_pg](#) (char * *url_s*, ssize_t *size*)

1.20.1.6 PKI_MEM_STACK* [URL_get_data_pg_url](#) (URL * *url*, ssize_t *size*)

1.20.1.7 int [URL_put_data_pg](#) (char * *url_s*, PKI_MEM * *data*)

1.20.1.8 int [URL_put_data_pg_url](#) (URL * *url*, PKI_MEM * *data*)

1.21 src/net/pkcs11.c File Reference

Include dependency graph for pkcs11.c:



Functions

- char * [pkcs11_parse_url_getval](#) (URL *url, char *keyword)
- PKI_MEM_STACK * [URL_get_data_pkcs11](#) (char *url_s, ssize_t size)
- PKI_MEM_STACK * [URL_get_data_pkcs11_url](#) (URL *url, ssize_t size)

1.21.1 Function Documentation

1.21.1.1 char* [pkcs11_parse_url_getval](#) (URL * *url*, char * *keyword*)

1.21.1.2 PKI_MEM_STACK* URL_get_data_pkcs11 (char * *url_s*, ssize_t *size*)

1.21.1.3 PKI_MEM_STACK* URL_get_data_pkcs11_url (URL * *url*, ssize_t *size*)

1.22 src/net/sock.c File Reference

Include dependency graph for sock.c:



Defines

- #define [LISTENQ](#) 30

Functions

- int [_Accept](#) (int listen_sockfd, struct sockaddr *cliaddr, socklen_t *addrlenp)
- int [_Close](#) (int fd)
- int [_Connect](#) (int sockfd, const SA *srvaddr, socklen_t addrlen)
- int [_Listen](#) (char *hostname, int port)
- ssize_t [_Read](#) (int fd, void *bufptr, size_t nbytes)
- int [_Select](#) (int maxfdp1, fd_set *readset, fd_set *writeset, fd_set *exceptset, struct timeval *timeout)
- void [_Shutdown](#) (int fd, int howto)
- int [_Socket](#) (int family, int type, int protocol)
- ssize_t [_Write](#) (int fd, void *bufptr, size_t nbytes)
- hostent * [Gethostbyname](#) (const char *hostname)
- int [inet_close](#) (int fd)
- int [inet_connect](#) (URL *url)
- int [PKI_NET_accept](#) (int sock, int timeout)

Returns the connected socket as a result of an Accept.

- int [PKI_NET_close](#) (int sock)

Closes the connection to an open connection to an host.

- PKI_MEM * [PKI_NET_get_data](#) (int fd, int timeout, size_t max_size)

Returns data read from a socket.

- int [PKI_NET_listen](#) (char *host, int port)

Returns a reference to a listen socket.

- int [PKI_NET_open](#) (URL *url, int timeout)

Connects to an host and returns the connected socket.

- ssize_t [PKI_NET_read](#) (int fd, void *bufptr, size_t nbytes, int timeout)

Reads n-bytes of data from a socket.

- int [PKI_NET_socket](#) (int family, int type, int protocol)

Returns a reference to a new socket (int).

- ssize_t [PKI_NET_write](#) (int fd, void *bufptr, size_t nbytes)

Writes n bytes of data to a socket.

Variables

- int [h_erno](#)

1.22.1 Define Documentation

1.22.1.1 #define LISTENQ 30

1.22.2 Function Documentation

1.22.2.1 int _Accept (int *listen_sockfd*, struct sockaddr * *cliaddr*, socklen_t * *addrlenp*)

1.22.2.2 int _Close (int *fd*)

1.22.2.3 int _Connect (int *sockfd*, const SA * *srvaddr*, socklen_t *addrlen*)

1.22.2.4 int _Listen (char * *hostname*, int *port*)

1.22.2.5 ssize_t _Read (int *fd*, void * *bufptr*, size_t *nbytes*)

1.22.2.6 int _Select (int *maxfdp1*, fd_set * *readset*, fd_set * *writeset*, fd_set * *exceptset*, struct timeval * *timeout*)

1.22.2.7 void _Shutdown (int *fd*, int *howto*)

1.22.2.8 int _Socket (int *family*, int *type*, int *protocol*)

1.22.2.9 ssize_t _Write (int *fd*, void * *bufptr*, size_t *nbytes*)

1.22.2.10 struct hostent* Gethostbyname (const char * *hostname*)

1.22.2.11 int inet_close (int *fd*)

1.22.2.12 int inet_connect (URL * *url*)

1.22.2.13 int PKI_NET_accept (int *sock*, int *timeout*)

Returns the connected socket as a result of an Accept.

1.22.2.14 int PKI_NET_close (int sock)

Closes the connection to an open connection to an host.

1.22.2.15 PKI_MEM* PKI_NET_get_data (int fd, int timeout, size_t max_size)

Returns data read from a socket.

1.22.2.16 int PKI_NET_listen (char * host, int port)

Returns a reference to a listen socket.

1.22.2.17 int PKI_NET_open (URL * url, int timeout)

Connects to an host and returns the connected socket.

1.22.2.18 ssize_t PKI_NET_read (int fd, void * bufptr, size_t nbytes, int timeout)

Reads n-bytes of data from a socket.

1.22.2.19 int PKI_NET_socket (int family, int type, int protocol)

Returns a reference to a new socket (int).

1.22.2.20 ssize_t PKI_NET_write (int fd, void * bufptr, size_t nbytes)

Writes n bytes of data to a socket.

1.22.3 Variable Documentation**1.22.3.1 int [h_errno](#)****1.23 src/net/ssl.c File Reference**

Include dependency graph for ssl.c:

**Defines**

- #define [BUFF_MAX_SIZE](#) 2048

Functions

- int [PKI_SSL_close](#) (PKI_SSL *ssl)
Closes an SSL connection.
- int [PKI_SSL_connect](#) (PKI_SSL *ssl, char *url_s, int timeout)
Initiates an SSL connection to a URL passed as a string.

- int [PKI_SSL_connect_url](#) (PKI_SSL *ssl, URL *url, int timeout)
Initiates an SSL connection to a URL passed as a URL object.
- void [PKI_SSL_free](#) (PKI_SSL *ssl)
Frees memory associated with a PKI_SSL.
- pki_x509_st * [PKI_SSL_get_peer_cert](#) (PKI_SSL *ssl)
Returns the Peer certificate used in a connected PKI_SSL.
- PKI_X509_CERT_STACK * [PKI_SSL_get_peer_chain](#) (PKI_SSL *ssl)
Returns the peer certificate chain as a new PKI_X509_CERT_STACK.
- const char * [PKI_SSL_get_servername](#) (PKI_SSL *ssl)
Returns the extension value provided in Client Hello or NULL.
- PKI_SSL * [PKI_SSL_new](#) (PKI_SSL_ALGOR *algor)
Sets the options for a new PKI_SSL object.
- ssize_t [PKI_SSL_read](#) (PKI_SSL *ssl, char *buf, ssize_t size)
Reads data from a connected PKI_SSL.
- int [PKI_SSL_set_algor](#) (PKI_SSL *ssl, PKI_SSL_ALGOR *algor)
Sets the protocol for a new PKI_SSL object.
- int [PKI_SSL_set_cipher](#) (PKI_SSL *ssl, char *cipher)
Sets the Chiphers to be used.
- int [PKI_SSL_set_flags](#) (PKI_SSL *ssl, int flags)
Sets the SSL connection flags.
- int [PKI_SSL_set_token](#) (PKI_SSL *ssl, struct pki_token_st *tk)
Sets the PKI_TOKEN to be used for the SSL connection.
- int [PKI_SSL_set_trusted](#) (PKI_SSL *ssl, PKI_X509_CERT_STACK *sk)
Sets the list of trusted certificates for SSL connections.
- ssize_t [PKI_SSL_write](#) (PKI_SSL *ssl, char *buf, ssize_t size)
Writes data to a connected PKI_SSL.

1.23.1 Define Documentation

1.23.1.1 #define BUFF_MAX_SIZE 2048

1.23.2 Function Documentation

1.23.2.1 int PKI_SSL_close (PKI_SSL *ssl)

Closes an SSL connection.

1.23.2.2 int PKI_SSL_connect (PKI_SSL * *ssl*, char * *url_s*, int *timeout*)

Initiates an SSL connection to a URL passed as a string.

1.23.2.3 int PKI_SSL_connect_url (PKI_SSL * *ssl*, URL * *url*, int *timeout*)

Initiates an SSL connection to a URL passed as a URL object.

1.23.2.4 void PKI_SSL_free (PKI_SSL * *ssl*)

Frees memory associated with a PKI_SSL.

1.23.2.5 struct pki_x509_st* PKI_SSL_get_peer_cert (PKI_SSL * *ssl*)

Returns the Peer certificate used in a connected PKI_SSL.

1.23.2.6 PKI_X509_CERT_STACK* PKI_SSL_get_peer_chain (PKI_SSL * *ssl*)

Returns the peer certificate chain as a new PKI_X509_CERT_STACK.

1.23.2.7 const char* PKI_SSL_get_servername (PKI_SSL * *ssl*)

Returns the extension value provided in Client Hello or NULL.

1.23.2.8 PKI_SSL* PKI_SSL_new (PKI_SSL_ALGOR * *algor*)

Sets the options for a new PKI_SSL object.

1.23.2.9 ssize_t PKI_SSL_read (PKI_SSL * *ssl*, char * *buf*, ssize_t *size*)

Reads data from a connected PKI_SSL.

1.23.2.10 int PKI_SSL_set_algor (PKI_SSL * *ssl*, PKI_SSL_ALGOR * *algor*)

Sets the protocol for a new PKI_SSL object.

1.23.2.11 int PKI_SSL_set_cipher (PKI_SSL * *ssl*, char * *cipher*)

Sets the Chiphers to be used.

1.23.2.12 int PKI_SSL_set_flags (PKI_SSL * *ssl*, int *flags*)

Sets the SSL connection flags.

1.23.2.13 int PKI_SSL_set_token (PKI_SSL * *ssl*, struct pki_token_st * *tk*)

Sets the PKI_TOKEN to be used for the SSL connection.

1.23.2.14 int PKI_SSL_set_trusted (PKI_SSL * *ssl*, PKI_X509_CERT_STACK * *sk*)

Sets the list of trusted certificates for SSL connections.

1.23.2.15 ssize_t PKI_SSL_write (PKI_SSL *ssl, char *buf, ssize_t size)

Writes data to a connected PKI_SSL.

1.24 src/net/url.c File Reference

Include dependency graph for url.c:



Defines

- #define [BUFF_MAX_SIZE](#) 2048

Typedefs

- typedef uri_protocol [URI_PROTOCOL](#)
Returns a PKI_MEM_STACK object filled from a file descriptor.

Functions

- void [URL_free](#) (URL *url)
Releases the Memory associated to a URL data structure.
- PKI_MEM_STACK * [URL_get_data](#) (char *url_s, int timeout, ssize_t size, PKI_SSL *ssl)
Returns a PKI_MEM_STACK filled with the data from the URL string.
- PKI_MEM_STACK * [URL_get_data_fd](#) (URL *url, ssize_t size)
- PKI_MEM_STACK * [URL_get_data_file](#) (URL *url, ssize_t size)
Returns a PKI_MEM_STACK object filled from a file.
- PKI_MEM_STACK * [URL_get_data_url](#) (URL *url, int timeout, ssize_t size, PKI_SSL *ssl)
Returns a PKI_MEM_STACK filled with data from the passed URL object.
- char * [URL_get_local_addr](#) (void)
- URL * [URL_new](#) (char *url_s)
- const char * [URL_proto_to_string](#) (URI_PROTO proto)
- int [URL_put_data](#) (char *url_s, PKI_MEM *data, char *contType, PKI_MEM_STACK **ret_sk, int timeout, ssize_t max_size, PKI_SSL *ssl)
Sends/Writes a PKI_MEM object into a URL passed as a string.
- int [URL_put_data_fd](#) (URL *url, PKI_MEM *data)
- int [URL_put_data_file](#) (URL *url, PKI_MEM *data)
- int [URL_put_data_url](#) (URL *url, PKI_MEM *data, char *contType, PKI_MEM_STACK **ret_sk, int timeout, ssize_t max_size, PKI_SSL *ssl)

Variables

- [URI_PROTOCOL](#) `proto_list` []

1.24.1 Define Documentation

1.24.1.1 `#define BUFF_MAX_SIZE 2048`

1.24.2 Typedef Documentation

1.24.2.1 `typedef struct uri_protocol` [URI_PROTOCOL](#)

Returns a `PKI_MEM_STACK` object filled from a file descriptor.

This function returns a `PKI_MEM_STACK` object (actually filled with only one object in the stack), with the data retrieved from the URL specified as input. This function will accept only URL with `URI_PROTOCOL_FD` as its protocol.

1.24.3 Function Documentation

1.24.3.1 `void URL_free (URL * url)`

Releases the Memory associated to a URL data structure.

1.24.3.2 `PKI_MEM_STACK* URL_get_data (char * url_s, int timeout, ssize_t size, PKI_SSL * ssl)`

Returns a `PKI_MEM_STACK` filled with the data from the URL string.

Returns a `PKI_MEM_STACK` object filled with the data retrieved from the URL that is passed as input. This is the most general function provided by LibPKI that allows to retrieve an object from many different sources.

In case of failure NULL is returned.

1.24.3.3 `PKI_MEM_STACK* URL_get_data_fd (URL * url, ssize_t size)`

1.24.3.4 `PKI_MEM_STACK* URL_get_data_file (URL * url, ssize_t size)`

Returns a `PKI_MEM_STACK` object filled from a file.

This function returns a `PKI_MEM_STACK` object (actually filled with only one object in the stack), with the data retrieved from the URL specified as input. This function will accept only URL with `URI_PROTOCOL_FILE` as its protocol.

1.24.3.5 `PKI_MEM_STACK* URL_get_data_url (URL * url, int timeout, ssize_t size, PKI_SSL * ssl)`

Returns a `PKI_MEM_STACK` filled with data from the passed URL object.

Returns a `PKI_MEM_STACK` object filled with the data retrieved from the URL that is passed as input. This function is very similar to the `URL_get_data ()`. Use this function when you already have the URL object of your target data.

In case of failure NULL is returned.

Currently supported protocols are: URI_PROTO_FD, URI_PROTO_FILE, URI_PROTO_HTTP, URI_PROTO_LDAP, URI_PROTO_MYSQL, URI_PROTO_PG, URI_PROTO_PKCS11, URI_PROTO_ID.

1.24.3.6 `char* URL_get_local_addr (void)`

1.24.3.7 `URL* URL_new (char * url_s)`

1.24.3.8 `const char* URL_proto_to_string (URI_PROTO proto)`

1.24.3.9 `int URL_put_data (char * url_s, PKI_MEM * data, char * contType, PKI_MEM_STACK ** ret_sk, int timeout, ssize_t max_size, PKI_SSL * ssl)`

Sends/Writes a PKI_MEM object into a URL passed as a string.

This function sends (or writes) the content of a PKI_MEM object into a specific URL that is passed as a string. For a list of valid URL protocols, please refer to the [URL_new\(\)](#) function.

In case of failure PKI_ERR is returned, otherwise PKI_OK is.

Currently supported protocols are: URI_PROTO_FD, URI_PROTO_FILE, URI_PROTO_MYSQL, URI_PROTO_PG.

1.24.3.10 `int URL_put_data_fd (URL * url, PKI_MEM * data)`

1.24.3.11 `int URL_put_data_file (URL * url, PKI_MEM * data)`

1.24.3.12 `int URL_put_data_url (URL * url, PKI_MEM * data, char * contType, PKI_MEM_STACK ** ret_sk, int timeout, ssize_t max_size, PKI_SSL * ssl)`

1.24.4 Variable Documentation

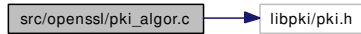
1.24.4.1 [URI_PROTOCOL proto_list\[\]](#)

Initial value:

```
{
    { URI_PROTO_FILE, "file" },
    { URI_PROTO_LDAP, "ldap" },
    { URI_PROTO_HTTP, "http" },
    { URI_PROTO_HTTPS, "https" },
    { URI_PROTO_FTP, "ftp" },
    { URI_PROTO_ID, "id" },
    { URI_PROTO_FD, "fd" },
    { URI_PROTO_MYSQL, "mysql" },
    { URI_PROTO_PG, "pg" },
    { URI_PROTO_PKCS11, "pkcs11" },
    { URI_PROTO SOCK, "sock" },
    { 0, NULL }
}
```

1.25 src/openssl/pki_algor.c File Reference

Include dependency graph for pki_algor.c:



Functions

- `PKI_ALGOR *` [`PKI_ALGOR_get`](#) (`PKI_ALGOR_ID` algor)
Build a `PKI_ALGOR` structure from its ID.
- `PKI_ALGOR *` [`PKI_ALGOR_get_by_name`](#) (`char *`alg_s)
*Build a `PKI_ALGOR` structure from its name (`char *`).*
- `PKI_DIGEST_ALG *` [`PKI_ALGOR_get_digest`](#) (`PKI_ALGOR *`algor)
Get the Digest Algorithm from the passed `PKI_ALGOR`.
- `PKI_ALGOR_ID` [`PKI_ALGOR_get_digest_id`](#) (`PKI_ALGOR *`algor)
Returns the `PKI_ALGOR_ID` of the digest used in the `PKI_ALGOR`.
- `PKI_ALGOR_ID` [`PKI_ALGOR_get_id`](#) (`PKI_ALGOR *`algor)
- `const char *` [`PKI_ALGOR_get_parsed`](#) (`PKI_ALGOR *`algor)
Returns a text representation of the algorithm identifier.
- `PKI_SCHEME_ID` [`PKI_ALGOR_get_scheme`](#) (`PKI_ALGOR *`algor)
Returns the `PKI_SCHEME_ID` (signature scheme ID) of the algorithm.
- `char *` [`PKI_ALGOR_ID_txt`](#) (`PKI_ALGOR_ID` algor)
Returns a text string with the algorithm identifier.
- `PKI_ALGOR_ID *` [`PKI_ALGOR_list`](#) (`PKI_SCHEME_ID` scheme)
- `size_t` [`PKI_ALGOR_list_size`](#) (`PKI_ALGOR_ID *`list)
- `PKI_DIGEST_ALG *` [`PKI_DIGEST_ALG_get`](#) (`PKI_ALGOR_ID` id)
*Returns the `PKI_DIGEST_ALG *` associated with the alg id.*
- `PKI_DIGEST_ALG *` [`PKI_DIGEST_ALG_get_by_key`](#) (`PKI_X509_KEYPAIR *`pkey)
Returns the digest algorithm based on the key.
- `PKI_DIGEST_ALG *` [`PKI_DIGEST_ALG_get_by_name`](#) (`char *`name)
*Returns the `PKI_DIGEST_ALG *` from its name.*
- `const char *` [`PKI_DIGEST_ALG_get_parsed`](#) (`PKI_DIGEST_ALG *`alg)
Returns the string representation of a digest algorithm.
- `PKI_ALGOR_ID *` [`PKI_DIGEST_ALG_list`](#) (`void`)

Variables

- PKI_ALGOR_ID [PKI_ALGOR_LIST_DSA](#) []
- PKI_ALGOR_ID [PKI_ALGOR_LIST_ECDSA](#) []
- PKI_ALGOR_ID [PKI_ALGOR_LIST_RSA](#) []
- PKI_ALGOR_ID [PKI_DIGEST_ALG_LIST](#) []

1.25.1 Function Documentation

1.25.1.1 PKI_ALGOR* PKI_ALGOR_get (PKI_ALGOR_ID *algor*)

Build a PKI_ALGOR structure from its ID.

1.25.1.2 PKI_ALGOR* PKI_ALGOR_get_by_name (char * *alg_s*)

Build a PKI_ALGOR structure from its name (char *).

The function returns the pointer to a PKI_ALGOR structure based on the name. Names are in the form of "RSA-SHA1", "RSA-SHA512", or "DSA-SHA1".

1.25.1.3 PKI_DIGEST_ALG* PKI_ALGOR_get_digest (PKI_ALGOR * *algor*)

Get the Digest Algorithm from the passed PKI_ALGOR.

1.25.1.4 PKI_ALGOR_ID PKI_ALGOR_get_digest_id (PKI_ALGOR * *algor*)

Returns the PKI_ALGOR_ID of the digest used in the PKI_ALGOR.

1.25.1.5 PKI_ALGOR_ID PKI_ALGOR_get_id (PKI_ALGOR * *algor*)

1.25.1.6 const char* PKI_ALGOR_get_parsed (PKI_ALGOR * *algor*)

Returns a text representation of the algorithm identifier.

1.25.1.7 PKI_SCHEME_ID PKI_ALGOR_get_scheme (PKI_ALGOR * *algor*)

Returns the PKI_SCHEME_ID (signature scheme ID) of the algorithm.

1.25.1.8 char* PKI_ALGOR_ID_txt (PKI_ALGOR_ID *algor*)

Returns a text string with the algorithm identifier.

1.25.1.9 PKI_ALGOR_ID* PKI_ALGOR_list (PKI_SCHEME_ID *scheme*)

1.25.1.10 size_t PKI_ALGOR_list_size (PKI_ALGOR_ID * *list*)

1.25.1.11 PKI_DIGEST_ALG* PKI_DIGEST_ALG_get (PKI_ALGOR_ID *id*)

Returns the PKI_DIGEST_ALG * associated with the alg id.

Returns the PKI_DIGEST_ALG * associated with the alg id. If the passed id is equal to 0, the default PKI_DIGEST_ALG is returned.

1.25.1.12 `PKI_DIGEST_ALG*` `PKI_DIGEST_ALG_get_by_key (PKI_X509_KEYPAIR * pkey)`

Returns the digest algorithm based on the key.

1.25.1.13 `PKI_DIGEST_ALG*` `PKI_DIGEST_ALG_get_by_name (char * name)`

Returns the `PKI_DIGEST_ALG *` from its name.

Returns the `PKI_DIGEST_ALG *` from its name (char *). An example of algorithm identifiers are "sha1", "sha224", "ripemd160". If the passed id is equal to 0, the default `PKI_DIGEST_ALG` is returned.

1.25.1.14 `const char*` `PKI_DIGEST_ALG_get_parsed (PKI_DIGEST_ALG * alg)`

Returns the string representation of a digest algorithm.

1.25.1.15 `PKI_ALGOR_ID*` `PKI_DIGEST_ALG_list (void)`**1.25.2** Variable Documentation**1.25.2.1** `PKI_ALGOR_ID` `PKI_ALGOR_LIST_DSA[]`

Initial value:

```
{
    PKI_ALGOR_DSA_SHA1,
    PKI_ALGOR_DSA_SHA224,
    PKI_ALGOR_DSA_SHA256,
    PKI_ALGOR_UNKNOWN
}
```

1.25.2.2 `PKI_ALGOR_ID` `PKI_ALGOR_LIST_ECDSA[]`

Initial value:

```
{
    PKI_ALGOR_ECDSA_SHA1,
    PKI_ALGOR_ECDSA_SHA224,
    PKI_ALGOR_ECDSA_SHA256,
    PKI_ALGOR_ECDSA_SHA384,
    PKI_ALGOR_ECDSA_SHA512,
    PKI_ALGOR_UNKNOWN
}
```

1.25.2.3 `PKI_ALGOR_ID` `PKI_ALGOR_LIST_RSA[]`

Initial value:

```
{
    PKI_ALGOR_RSA_MD4,
    PKI_ALGOR_RSA_MD5,
    PKI_ALGOR_RSA_SHA1,
    PKI_ALGOR_RSA_SHA224,
    PKI_ALGOR_RSA_SHA256,
}
```

```

    PKI_ALGOR_RSA_SHA384,
    PKI_ALGOR_RSA_SHA512,
    PKI_ALGOR_UNKNOWN
}

```

1.25.2.4 PKI_ALGOR_ID PKI_DIGEST_ALG_LIST[]

Initial value:

```

{
    PKI_ALGOR_MD4,
    PKI_ALGOR_MD5,
    PKI_ALGOR_SHA1,
    PKI_ALGOR_SHA224,
    PKI_ALGOR_SHA256,
    PKI_ALGOR_SHA384,
    PKI_ALGOR_SHA512,
    PKI_ALGOR_RIPEMD160,
    PKI_ALGOR_DSS1,
    PKI_ALGOR_UNKNOWN
}

```

1.26 src/openssl/pki_digest.c File Reference

Include dependency graph for pki_digest.c:



Functions

- void [PKI_DIGEST_free](#) (PKI_DIGEST *data)
Free the memory associated with a PKI_DIGEST data structure.
- char * [PKI_DIGEST_get_parsed](#) (PKI_DIGEST *digest)
Returns the parsed (string) version of the digest content.
- size_t [PKI_DIGEST_get_size](#) (PKI_DIGEST_ALG *alg)
Returns the size of the output of the selected digest algorithm.
- PKI_DIGEST * [PKI_DIGEST_MEM_new](#) (PKI_DIGEST_ALG *alg, PKI_MEM *data)
Calculates a digest over data contained in a PKI_MEM.
- PKI_DIGEST * [PKI_DIGEST_MEM_new_by_name](#) (char *alg_name, PKI_MEM *data)
- PKI_DIGEST * [PKI_DIGEST_new](#) (PKI_DIGEST_ALG *alg, unsigned char *data, size_t size)
Calculate digest over data provided in a buffer.
- PKI_DIGEST * [PKI_DIGEST_new_by_name](#) (char *alg_name, unsigned char *data, size_t size)
Calculates a digest over data buffer.
- PKI_DIGEST * [PKI_DIGEST_URL_new](#) (PKI_DIGEST_ALG *alg, URL *url)

Calculate the digest of data retrieved via a URL.

- `PKI_DIGEST * PKI_DIGEST_URL_new_by_name (char *alg_name, URL *url)`

1.26.1 Function Documentation

1.26.1.1 `void PKI_DIGEST_free (PKI_DIGEST * data)`

Free the memory associated with a `PKI_DIGEST` data structure.

1.26.1.2 `char* PKI_DIGEST_get_parsed (PKI_DIGEST * digest)`

Returns the parsed (string) version of the digest content.

1.26.1.3 `size_t PKI_DIGEST_get_size (PKI_DIGEST_ALG * alg)`

Returns the size of the output of the selected digest algorithm.

1.26.1.4 `PKI_DIGEST* PKI_DIGEST_MEM_new (PKI_DIGEST_ALG * alg, PKI_MEM * data)`

Calculates a digest over data contained in a `PKI_MEM`.

1.26.1.5 `PKI_DIGEST* PKI_DIGEST_MEM_new_by_name (char *alg_name, PKI_MEM * data)`

1.26.1.6 `PKI_DIGEST* PKI_DIGEST_new (PKI_DIGEST_ALG * alg, unsigned char * data, size_t size)`

Calculate digest over data provided in a buffer.

1.26.1.7 `PKI_DIGEST* PKI_DIGEST_new_by_name (char *alg_name, unsigned char * data, size_t size)`

Calculates a digest over data buffer.

1.26.1.8 `PKI_DIGEST* PKI_DIGEST_URL_new (PKI_DIGEST_ALG * alg, URL * url)`

Calculate the digest of data retrieved via a URL.

1.26.1.9 `PKI_DIGEST* PKI_DIGEST_URL_new_by_name (char *alg_name, URL * url)`

1.27 src/openssl/pki_id.c File Reference

Include dependency graph for `pki_id.c`:



Functions

- PKI_ID [PKI_ID_get](#) (PKI_ID id)
Checks if a PKI Identifier exists.
- PKI_ID [PKI_ID_get_by_name](#) (char *name)
Create a new ID object.
- const char * [PKI_ID_get_txt](#) (PKI_ID id)

1.27.1 Function Documentation**1.27.1.1 PKI_ID PKI_ID_get (PKI_ID id)**

Checks if a PKI Identifier exists.

This function retrieves an ID generated from the passed ID, if the ID does not exist in the library database, it returns PKI_ID_UNKNOWN.

Basically it checks if it exists or not.

1.27.1.2 PKI_ID PKI_ID_get_by_name (char * name)

Create a new ID object.

Create a new ID by using its name. It returns an int if successful, otherwise it returns NULL

1.27.1.3 const char* PKI_ID_get_txt (PKI_ID id)**1.28 src/openssl/pki_integer.c File Reference**

Include dependency graph for pki_integer.c:

**Functions**

- int [PKI_INTEGER_cmp](#) (PKI_INTEGER *a, PKI_INTEGER *b)
Compare two PKI_INTEGERs.
- PKI_INTEGER * [PKI_INTEGER_dup](#) (PKI_INTEGER *a)
Duplicates a PKI_INTEGER data structure.
- int [PKI_INTEGER_free](#) (PKI_INTEGER *i)
Frees memory associated with a PKI_INTEGER.
- void [PKI_INTEGER_free_void](#) (void *i)
- char * [PKI_INTEGER_get_parsed](#) (PKI_INTEGER *i)
Returns a string representation of the PKI_INTEGER.

- `PKI_INTEGER * PKI_INTEGER_new` (long long val)
Returns a PKI_INTEGER object starting from a long long value.
- `PKI_INTEGER * PKI_INTEGER_new_bin` (unsigned char *data, size_t size)
Generate a new PKI_INTEGER from raw bit data.
- `PKI_INTEGER * PKI_INTEGER_new_char` (char *val)
Returns a PKI_INTEGER object from a string.
- `int PKI_INTEGER_print` (PKI_INTEGER *s)
Prints the contents of a PKI_INTEGER to Standard Output.
- `int PKI_INTEGER_print_fp` (FILE *fp, PKI_INTEGER *s)
Prints the contents of a PKI_INTEGER to a FILE pointer (eg., stdout).

1.28.1 Function Documentation

1.28.1.1 `int PKI_INTEGER_cmp (PKI_INTEGER * a, PKI_INTEGER * b)`

Compare two PKI_INTEGERs.

1.28.1.2 `PKI_INTEGER* PKI_INTEGER_dup (PKI_INTEGER * a)`

Duplicates a PKI_INTEGER data structure.

1.28.1.3 `int PKI_INTEGER_free (PKI_INTEGER * i)`

Frees memory associated with a PKI_INTEGER.

1.28.1.4 `void PKI_INTEGER_free_void (void * i)`

1.28.1.5 `char* PKI_INTEGER_get_parsed (PKI_INTEGER * i)`

Returns a string representation of the PKI_INTEGER.

1.28.1.6 `PKI_INTEGER* PKI_INTEGER_new (long long val)`

Returns a PKI_INTEGER object starting from a long long value.

1.28.1.7 `PKI_INTEGER* PKI_INTEGER_new_bin (unsigned char * data, size_t size)`

Generate a new PKI_INTEGER from raw bit data.

1.28.1.8 `PKI_INTEGER* PKI_INTEGER_new_char (char * val)`

Returns a PKI_INTEGER object from a string.

1.28.1.9 `int PKI_INTEGER_print (PKI_INTEGER * s)`

Prints the contents of a PKI_INTEGER to Standard Output.

1.28.1.10 int PKI_INTEGER_print_fp (FILE *fp, PKI_INTEGER *s)

Prints the contents of a PKI_INTEGER to a FILE pointer (eg., stdout).

1.29 src/openssl/pki_keypair.c File Reference

Include dependency graph for pki_keypair.c:

**Functions**

- void [PKI_X509_KEYPAIR_free](#) (PKI_X509_KEYPAIR *key)
- void [PKI_X509_KEYPAIR_free_void](#) (void *key)
- PKI_MEM * [PKI_X509_KEYPAIR_get_p8](#) (PKI_X509_KEYPAIR *k)
Returns the passed PKI_X509_KEYPAIR in PKCS#8 format.
- char * [PKI_X509_KEYPAIR_get_parsed](#) (PKI_X509_KEYPAIR *pkey)
*Returns a char * with a string representation of the Keypair.*
- PKI_X509_KEYPAIR * [PKI_X509_KEYPAIR_new](#) (int type, int bits, char *label, PKI_CRED *cred, HSM *hsm)
Generate a new Keypair with the passed label (required for PKCS#11 HSMs) as target.
- PKI_X509_KEYPAIR * [PKI_X509_KEYPAIR_new_null](#) ()
- PKI_X509_KEYPAIR * [PKI_X509_KEYPAIR_new_p8](#) (PKI_MEM *buf)
Reads a PKI_X509_KEYPAIR from a PKCS#8 format.
- PKI_X509_KEYPAIR * [PKI_X509_KEYPAIR_new_url](#) (int type, int bits, URL *url, PKI_CRED *cred, HSM *hsm)
Generate a new Keypair with the passed URL (required for PKCS#11 HSMs) as target.
- PKI_DIGEST * [PKI_X509_KEYPAIR_pub_digest](#) (PKI_X509_KEYPAIR *k, PKI_DIGEST_ALG *md)
*Returns the (unsigned char *) digest of the pubkey.*
- PKI_DIGEST * [PKI_X509_KEYPAIR_VALUE_pub_digest](#) (PKI_X509_KEYPAIR_VALUE *pkey, PKI_DIGEST_ALG *md)
*Returns the (unsigned char *) digest of a pubkey value.*

1.29.1 Function Documentation**1.29.1.1 void PKI_X509_KEYPAIR_free (PKI_X509_KEYPAIR *key)****1.29.1.2 void PKI_X509_KEYPAIR_free_void (void *key)**

1.29.1.3 PKI_MEM* PKI_X509_KEYPAIR_get_p8 (PKI_X509_KEYPAIR * k)

Returns the passed PKI_X509_KEYPAIR in PKCS#8 format.

1.29.1.4 char* PKI_X509_KEYPAIR_get_parsed (PKI_X509_KEYPAIR * pkey)

Returns a char * with a string representation of the Keypair.

1.29.1.5 PKI_X509_KEYPAIR* PKI_X509_KEYPAIR_new (int type, int bits, char * label, PKI_CRED * cred, HSM * hsm)

Generate a new Keypair with the passed label (required for PKCS#11 HSMs) as target.

1.29.1.6 PKI_X509_KEYPAIR* PKI_X509_KEYPAIR_new_null ()**1.29.1.7 PKI_X509_KEYPAIR* PKI_X509_KEYPAIR_new_p8 (PKI_MEM * buf)**

Reads a PKI_X509_KEYPAIR from a PKCS#8 format.

1.29.1.8 PKI_X509_KEYPAIR* PKI_X509_KEYPAIR_new_url (int type, int bits, URL * url, PKI_CRED * cred, HSM * hsm)

Generate a new Keypair with the passed URL (required for PKCS#11 HSMs) as target.

1.29.1.9 PKI_DIGEST* PKI_X509_KEYPAIR_pub_digest (PKI_X509_KEYPAIR * k, PKI_DIGEST_ALG * md)

Returns the (unsigned char *) digest of the pubkey.

1.29.1.10 PKI_DIGEST* PKI_X509_KEYPAIR_VALUE_pub_digest (PKI_X509_KEYPAIR_VALUE * pkey, PKI_DIGEST_ALG * md)

Returns the (unsigned char *) digest of a pubkey value.

1.30 src/openssl/pki_ocsp_req.c File Reference

Include dependency graph for pki_ocsp_req.c:

**Functions**

- PKI_X509_OCSP_REQ_VALUE * [d2i_OCSP_REQ_bio](#) (PKI_IO *bp, PKI_X509_OCSP_REQ_VALUE *p)
- int [i2d_OCSP_REQ_bio](#) (PKI_IO *bp, PKI_X509_OCSP_REQ_VALUE *o)
- PKI_X509_OCSP_REQ_VALUE * [PEM_read_bio_OCSP_REQ](#) (PKI_IO *bp, void *a, void *b, void *c)
- int [PEM_write_bio_OCSP_REQ](#) (PKI_IO *bp, PKI_X509_OCSP_REQ_VALUE *o)
- int [PKI_OCSP_nonce_check](#) (PKI_X509_OCSP_REQ *req, PKI_X509_OCSP_RESP *resp)

Checks the NONCE between a Request and a Response.

- int [PKI_X509_OCSP_REQ_add_cert](#) (PKI_X509_OCSP_REQ *req, PKI_X509_CERT *cert, PKI_X509_CERT *issuer, PKI_DIGEST_ALG *digest)

Adds one basic request (one certificate) to the request by using the passed PKI_INTEGER as the serial number of the certificate.

- int [PKI_X509_OCSP_REQ_add_longlong](#) (PKI_X509_OCSP_REQ *req, long long serial, PKI_X509_CERT *issuer, PKI_DIGEST_ALG *digest)

Adds one basic request (one certificate) to the request by using the passed num (long long) as the serial number of the certificate.

- int [PKI_X509_OCSP_REQ_add_nonce](#) (PKI_X509_OCSP_REQ *req, size_t size)

Adds a random nonce to a request. If size = 0, the default size is used instead.

- int [PKI_X509_OCSP_REQ_add_serial](#) (PKI_X509_OCSP_REQ *req, PKI_INTEGER *serial, PKI_X509_CERT *issuer, PKI_DIGEST_ALG *digest)

Adds one basic request (one certificate) to the request by using the passed PKI_INTEGER as the serial number of the certificate.

- int [PKI_X509_OCSP_REQ_add_txt](#) (PKI_X509_OCSP_REQ *req, char *serial, PKI_X509_CERT *issuer, PKI_DIGEST_ALG *digest)

*Adds one basic request (one certificate) to the request by using the passed string (char *) as the serial number of the certificate.*

- int [PKI_X509_OCSP_REQ_DATA_sign](#) (PKI_X509_OCSP_REQ *req, PKI_X509_KEYPAIR *k, PKI_DIGEST_ALG *md)

- int [PKI_X509_OCSP_REQ_elements](#) (PKI_X509_OCSP_REQ *req)

Returns the number of single requests present in the OCSP REQ.

- void [PKI_X509_OCSP_REQ_free](#) (PKI_X509_OCSP_REQ *x)

Frees the memory associated with a PKI_X509_OCSP_REQ object.

- void [PKI_X509_OCSP_REQ_free_void](#) (void *x)

- PKI_OCSP_CERTID * [PKI_X509_OCSP_REQ_get_cid](#) (PKI_X509_OCSP_REQ *req, int num)

Returns the n-th PKI_OCSP_CERTID from an OCSP_REQ.

- void * [PKI_X509_OCSP_REQ_get_data](#) (PKI_X509_OCSP_REQ *req, PKI_X509_DATA type)

Returns a pointer to the data present in the OCSP request.

- char * [PKI_X509_OCSP_REQ_get_parsed](#) (PKI_X509_OCSP_REQ *req, PKI_X509_DATA type)

*Returns a char * representation of the data present in the OCSP request.*

- PKI_INTEGER * [PKI_X509_OCSP_REQ_get_serial](#) (PKI_X509_OCSP_REQ *req, int num)

Returns the serial of the requested certificate from the n-th single request.

- PKI_X509_OCSP_REQ * [PKI_X509_OCSP_REQ_new](#) (void)

Generates an empty OCSP request.

- PKI_X509_OCSP_REQ * [PKI_X509_OCSP_REQ_new_null](#) (void)

- int [PKI_X509_OCSP_REQ_print_parsed](#) (PKI_X509_OCSP_REQ *req, PKI_X509_DATA type, int fd)

Prints the requested data from the OCSP request to the file descriptor passed as an argument.

- int [PKI_X509_OCSP_REQ_sign](#) (PKI_X509_OCSP_REQ *req, PKI_X509_KEYPAIR *keypair, PKI_X509_CERT *cert, PKI_X509_CERT *issuer, PKI_X509_CERT_STACK *otherCerts, PKI_DIGEST_ALG *digest)

Signs a PKI_X509_OCSP_REQ, for a simpler API use PKI_X509_OCSP_REQ_sign_tk.

- int [PKI_X509_OCSP_REQ_sign_tk](#) (PKI_X509_OCSP_REQ *req, PKI_TOKEN *tk)

Signs a PKI_X509_OCSP_REQ object by using a token.

1.30.1 Function Documentation

1.30.1.1 [PKI_X509_OCSP_REQ_VALUE* d2i_OCSP_REQ_bio](#) (PKI_IO * bp, PKI_X509_OCSP_REQ_VALUE * p)

1.30.1.2 [int i2d_OCSP_REQ_bio](#) (PKI_IO * bp, PKI_X509_OCSP_REQ_VALUE * o)

1.30.1.3 [PKI_X509_OCSP_REQ_VALUE* PEM_read_bio_OCSP_REQ](#) (PKI_IO * bp, void * a, void * b, void * c)

1.30.1.4 [int PEM_write_bio_OCSP_REQ](#) (PKI_IO * bp, PKI_X509_OCSP_REQ_VALUE * o)

1.30.1.5 [int PKI_OCSP_nonce_check](#) (PKI_X509_OCSP_REQ * req, PKI_X509_OCSP_RESP * resp)

Checks the NONCE between a Request and a Response.

1.30.1.6 [int PKI_X509_OCSP_REQ_add_cert](#) (PKI_X509_OCSP_REQ * req, PKI_X509_CERT * cert, PKI_X509_CERT * issuer, PKI_DIGEST_ALG * digest)

Adds one basic request (one certificate) to the request by using the passed PKI_INTEGER as the serial number of the certificate.

1.30.1.7 [int PKI_X509_OCSP_REQ_add_longlong](#) (PKI_X509_OCSP_REQ * req, long long serial, PKI_X509_CERT * issuer, PKI_DIGEST_ALG * digest)

Adds one basic request (one certificate) to the request by using the passed num (long long) as the serial number of the certificate.

1.30.1.8 [int PKI_X509_OCSP_REQ_add_nonce](#) (PKI_X509_OCSP_REQ * req, size_t size)

Adds a random nonce to a request. If size = 0, the default size is used instead.

1.30.1.9 `int PKI_X509_OCSP_REQ_add_serial (PKI_X509_OCSP_REQ * req, PKI_INTEGER * serial, PKI_X509_CERT * issuer, PKI_DIGEST_ALG * digest)`

Adds one basic request (one certificate) to the request by using the passed PKI_INTEGER as the serial number of the certificate.

1.30.1.10 `int PKI_X509_OCSP_REQ_add_txt (PKI_X509_OCSP_REQ * req, char * serial, PKI_X509_CERT * issuer, PKI_DIGEST_ALG * digest)`

Adds one basic request (one certificate) to the request by using the passed string (char *) as the serial number of the certificate.

1.30.1.11 `int PKI_X509_OCSP_REQ_DATA_sign (PKI_X509_OCSP_REQ * req, PKI_X509_KEYPAIR * k, PKI_DIGEST_ALG * md)`

1.30.1.12 `int PKI_X509_OCSP_REQ_elements (PKI_X509_OCSP_REQ * req)`

Returns the number of single requests present in the OCSP REQ.

1.30.1.13 `void PKI_X509_OCSP_REQ_free (PKI_X509_OCSP_REQ * x)`

Frees the memory associated with a PKI_X509_OCSP_REQ object.

1.30.1.14 `void PKI_X509_OCSP_REQ_free_void (void * x)`

1.30.1.15 `PKI_OCSP_CERTID* PKI_X509_OCSP_REQ_get_cid (PKI_X509_OCSP_REQ * req, int num)`

Returns the n-th PKI_OCSP_CERTID from an OCSP_REQ.

1.30.1.16 `void* PKI_X509_OCSP_REQ_get_data (PKI_X509_OCSP_REQ * req, PKI_X509_DATA type)`

Returns a pointer to the data present in the OCSP request.

1.30.1.17 `char* PKI_X509_OCSP_REQ_get_parsed (PKI_X509_OCSP_REQ * req, PKI_X509_DATA type)`

Returns a char * representation of the data present in the OCSP request.

1.30.1.18 `PKI_INTEGER* PKI_X509_OCSP_REQ_get_serial (PKI_X509_OCSP_REQ * req, int num)`

Returns the serial of the requested certificate from the n-th single request.

1.30.1.19 `PKI_X509_OCSP_REQ* PKI_X509_OCSP_REQ_new (void)`

Generates an empty OCSP request.

1.30.1.20 `PKI_X509_OCSP_REQ* PKI_X509_OCSP_REQ_new_null (void)`

1.30.1.21 `int PKI_X509_OCSP_REQ_print_parsed (PKI_X509_OCSP_REQ * req, PKI_X509_DATA type, int fd)`

Prints the requested data from the OCSP request to the file descriptor passed as an argument.

1.30.1.22 `int PKI_X509_OCSP_REQ_sign (PKI_X509_OCSP_REQ * req, PKI_X509_KEYPAIR * keypair, PKI_X509_CERT * cert, PKI_X509_CERT * issuer, PKI_X509_CERT_STACK * other-Certs, PKI_DIGEST_ALG * digest)`

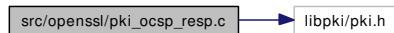
Signs a PKI_X509_OCSP_REQ, for a simpler API use PKI_X509_OCSP_REQ_sign_tk.

1.30.1.23 `int PKI_X509_OCSP_REQ_sign_tk (PKI_X509_OCSP_REQ * req, PKI_TOKEN * tk)`

Signs a PKI_X509_OCSP_REQ object by using a token.

1.31 src/openssl/pki_ocsp_resp.c File Reference

Include dependency graph for pki_ocsp_resp.c:



Functions

- `PKI_OCSP_RESP * d2i_PKI_OCSP_RESP_bio (PKI_IO *bp, PKI_OCSP_RESP **p)`
- `int i2d_PKI_OCSP_RESP_bio (PKI_IO *bp, PKI_OCSP_RESP *o)`
- `PKI_OCSP_RESP * PEM_read_bio_PKI_OCSP_RESP (PKI_IO *bp, void *a, void *b, void *c)`
- `int PEM_write_bio_PKI_OCSP_RESP (PKI_IO *bp, PKI_OCSP_RESP *o)`
- `void PKI_OCSP_RESP_free (PKI_OCSP_RESP *x)`
- `PKI_OCSP_RESP * PKI_OCSP_RESP_new (void)`
Generates an empty OCSP request.
- `int PKI_X509_OCSP_RESP_add (PKI_X509_OCSP_RESP *resp, OCSP_CERTID *cid, PKI_OCSP_CERTSTATUS status, PKI_TIME *revokeTime, PKI_TIME *thisUpdate, PKI_TIME *nextUpdate, PKI_X509_CRL_REASON reason, PKI_X509_EXTENSION *invalidityDate)`
Adds one basic request (one certificate) to the request by using the passed PKI_INTEGER as the serial number of the certificate.
- `int PKI_X509_OCSP_RESP_copy_nonce (PKI_X509_OCSP_RESP *resp, PKI_X509_OCSP_REQ *req)`
Copies the NONCE from a PKI_OCSP_RESP into the response.
- `int PKI_X509_OCSP_RESP_DATA_sign (PKI_X509_OCSP_RESP *resp, PKI_X509_KEYPAIR *k, PKI_DIGEST_ALG *md)`
- `void PKI_X509_OCSP_RESP_free (PKI_X509_OCSP_RESP *x)`
Frees the memory associated with a PKI_X509_OCSP_RESP object.
- `void PKI_X509_OCSP_RESP_free_void (void *x)`
- `void * PKI_X509_OCSP_RESP_get_data (PKI_X509_OCSP_RESP *r, PKI_X509_DATA type)`
Returns a pointer to the data present in the OCSP request.

- char * [PKI_X509_OCSP_RESP_get_parsed](#) (PKI_X509_OCSP_RESP *r, PKI_X509_DATA type)

*Returns a char * representation of the data present in the OCSP request.*

- PKI_X509_OCSP_RESP * [PKI_X509_OCSP_RESP_new](#) (void)
- PKI_X509_OCSP_RESP * [PKI_X509_OCSP_RESP_new_null](#) (void)
- int [PKI_X509_OCSP_RESP_print_parsed](#) (PKI_X509_OCSP_RESP *r, PKI_X509_DATA type, int fd)

Prints the requested data from the OCSP request to the file descriptor passed as an argument.

- int [PKI_X509_OCSP_RESP_set_status](#) (PKI_X509_OCSP_RESP *x, PKI_X509_OCSP_RESP_STATUS status)

Sets the status of the request.

- int [PKI_X509_OCSP_RESP_sign](#) (PKI_X509_OCSP_RESP *resp, PKI_X509_KEYPAIR *keypair, PKI_X509_CERT *cert, PKI_X509_CERT *issuer, PKI_X509_CERT_STACK *other-Certs, PKI_DIGEST_ALG *digest)

Signs a PKI_X509_OCSP_RESP, for a simpler API use PKI_X509_OCSP_RESP_sign_tk.

- int [PKI_X509_OCSP_RESP_sign_tk](#) (PKI_X509_OCSP_RESP *r, PKI_TOKEN *tk)

Signs a PKI_X509_OCSP_RESP object by using a token.

1.31.1 Function Documentation

1.31.1.1 PKI_OCSP_RESP* d2i_PKI_OCSP_RESP_bio (PKI_IO *bp, PKI_OCSP_RESP **p)

1.31.1.2 int i2d_PKI_OCSP_RESP_bio (PKI_IO *bp, PKI_OCSP_RESP *o)

1.31.1.3 PKI_OCSP_RESP* PEM_read_bio_PKI_OCSP_RESP (PKI_IO *bp, void *a, void *b, void *c)

1.31.1.4 int PEM_write_bio_PKI_OCSP_RESP (PKI_IO *bp, PKI_OCSP_RESP *o)

1.31.1.5 void PKI_OCSP_RESP_free (PKI_OCSP_RESP *x)

1.31.1.6 PKI_OCSP_RESP* PKI_OCSP_RESP_new (void)

Generates an empty OCSP request.

1.31.1.7 int PKI_X509_OCSP_RESP_add (PKI_X509_OCSP_RESP *resp, OCSP_CERTID *cid, PKI_OCSP_CERTSTATUS status, PKI_TIME *revokeTime, PKI_TIME *thisUpdate, PKI_TIME *nextUpdate, PKI_X509_CRL_REASON reason, PKI_X509_EXTENSION *invalidityDate)

Adds one basic request (one certificate) to the request by using the passed PKI_INTEGER as the serial number of the certificate.

1.31.1.8 `int PKI_X509_OCSP_RESP_copy_nonce (PKI_X509_OCSP_RESP * resp, PKI_X509_OCSP_REQ * req)`

Copies the NONCE from a PKI_OCSP_RESP into the response.

1.31.1.9 `int PKI_X509_OCSP_RESP_DATA_sign (PKI_X509_OCSP_RESP * resp, PKI_X509_KEYPAIR * k, PKI_DIGEST_ALG * md)`

1.31.1.10 `void PKI_X509_OCSP_RESP_free (PKI_X509_OCSP_RESP * x)`

Frees the memory associated with a PKI_X509_OCSP_RESP object.

1.31.1.11 `void PKI_X509_OCSP_RESP_free_void (void * x)`

1.31.1.12 `void* PKI_X509_OCSP_RESP_get_data (PKI_X509_OCSP_RESP * r, PKI_X509_DATA type)`

Returns a pointer to the data present in the OCSP request.

1.31.1.13 `char* PKI_X509_OCSP_RESP_get_parsed (PKI_X509_OCSP_RESP * r, PKI_X509_DATA type)`

Returns a char * representation of the data present in the OCSP request.

1.31.1.14 `PKI_X509_OCSP_RESP* PKI_X509_OCSP_RESP_new (void)`

1.31.1.15 `PKI_X509_OCSP_RESP* PKI_X509_OCSP_RESP_new_null (void)`

1.31.1.16 `int PKI_X509_OCSP_RESP_print_parsed (PKI_X509_OCSP_RESP * r, PKI_X509_DATA type, int fd)`

Prints the requested data from the OCSP request to the file descriptor passed as an argument.

1.31.1.17 `int PKI_X509_OCSP_RESP_set_status (PKI_X509_OCSP_RESP * x, PKI_X509_OCSP_RESP_STATUS status)`

Sets the status of the request.

1.31.1.18 `int PKI_X509_OCSP_RESP_sign (PKI_X509_OCSP_RESP * resp, PKI_X509_KEYPAIR * keypair, PKI_X509_CERT * cert, PKI_X509_CERT * issuer, PKI_X509_CERT_STACK * otherCerts, PKI_DIGEST_ALG * digest)`

Signs a PKI_X509_OCSP_RESP, for a simpler API use PKI_X509_OCSP_RESP_sign_tk.

1.31.1.19 `int PKI_X509_OCSP_RESP_sign_tk (PKI_X509_OCSP_RESP * r, PKI_TOKEN * tk)`

Signs a PKI_X509_OCSP_RESP object by using a token.

1.32 src/openssl/pki_oid.c File Reference

Include dependency graph for pki_oid.c:



Functions

- int [PKI_OID_cmp](#) (PKI_OID *a, PKI_OID *b)
Compares two PKI_OID and returns 0 if they match.
- PKI_OID * [PKI_OID_dup](#) (PKI_OID *a)
Returns a duplicate of the passed PKI_OID structure.
- void [PKI_OID_free](#) (PKI_OID *oid)
Free memory associated with a PKI_OID structure.
- void [PKI_OID_free_void](#) (void *buf)
- PKI_OID * [PKI_OID_get](#) (char *name)
See PKI_OID_new_text.
- const char * [PKI_OID_get_descr](#) (PKI_OID *a)
Return the description associated with a PKI_OID object.
- PKI_ID [PKI_OID_get_id](#) (PKI_OID *a)
Returns the PKI_ID of the object if recognized.
- char * [PKI_OID_get_str](#) (PKI_OID *a)
Returns a new allocated string representation of an OID.
- PKI_OID * [PKI_OID_new](#) (char *oid, char *name, char *descr)
Create a new OID object.
- PKI_OID * [PKI_OID_new_id](#) (PKI_ID id)
Returns the OID associated with a PKI_ID.
- PKI_OID * [PKI_OID_new_text](#) (char *name)
Retrieve a pointer to an OID.

1.32.1 Function Documentation

1.32.1.1 int [PKI_OID_cmp](#) (PKI_OID * a, PKI_OID * b)

Compares two PKI_OID and returns 0 if they match.

1.32.1.2 PKI_OID* [PKI_OID_dup](#) (PKI_OID * a)

Returns a duplicate of the passed PKI_OID structure.

1.32.1.3 void PKI_OID_free (PKI_OID * oid)

Free memory associated with a PKI_OID structure.

This function frees the memory associated with the provided pointer to a PKI_OID structure.

1.32.1.4 void PKI_OID_free_void (void * buf)**1.32.1.5 PKI_OID* PKI_OID_get (char * name)**

See PKI_OID_new_text.

1.32.1.6 const char* PKI_OID_get_descr (PKI_OID * a)

Return the description associated with a PKI_OID object.

1.32.1.7 PKI_ID PKI_OID_get_id (PKI_OID * a)

Returns the PKI_ID of the object if recognized.

1.32.1.8 char* PKI_OID_get_str (PKI_OID * a)

Returns a new allocated string representation of an OID.

1.32.1.9 PKI_OID* PKI_OID_new (char * oid, char * name, char * descr)

Create a new OID object.

Create a new OID by using its name. It returns a PKI_OID pointer if successful, otherwise it returns NULL.

1.32.1.10 PKI_OID* PKI_OID_new_id (PKI_ID id)

Returns the OID associated with a PKI_ID.

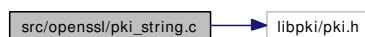
1.32.1.11 PKI_OID* PKI_OID_new_text (char * name)

Retrieve a pointer to an OID.

This function retrieves an OID pointer from the passed name. Check also the configuration options.

1.33 src/openssl/pki_string.c File Reference

Include dependency graph for pki_string.c:

**Functions**

- PKI_STRING * [PKI_STRING_dup](#) (PKI_STRING *a)

Duplicates a PKI_STRING data structure.

- void **PKI_STRING_free** (PKI_STRING *s)
Releases the memory associated with a PKI_STRING.
- char * **PKI_STRING_get_parsed** (PKI_STRING *s)
*Returns the parsed value (char *) of the PKI_STRING (in UTF-8).*
- int **PKI_STRING_get_type** (PKI_STRING *s)
Returns the type of the PKI_STRING (PKI_STRING_IA5, etc.).
- char * **PKI_STRING_get_utf8** (PKI_STRING *s)
*Returns the parsed value (char *) of the PKI_STRING (in UTF-8).*
- PKI_STRING * **PKI_STRING_new** (int type, char *val, ssize_t size)
Returns a new PKI_STRING of type and contents set from the passed parameters.
- PKI_STRING * **PKI_STRING_new_null** (int type)
Create an empty PKI_STRING of type passed as the only argument.
- int **PKI_STRING_print** (PKI_STRING *s)
Prints the contents of a PKI_STRING to Standard Output.
- int **PKI_STRING_print_fp** (FILE *fp, PKI_STRING *s)
Prints the contents of a PKI_STRING to a FILE pointer (eg., stdout).
- int **PKI_STRING_set** (PKI_STRING *s, char *content, ssize_t size)
Sets the content of a PKI_STRING.

1.33.1 Function Documentation

1.33.1.1 PKI_STRING* PKI_STRING_dup (PKI_STRING * a)

Duplicates a PKI_STRING data structure.

1.33.1.2 void PKI_STRING_free (PKI_STRING * s)

Releases the memory associated with a PKI_STRING.

1.33.1.3 char* PKI_STRING_get_parsed (PKI_STRING * s)

Returns the parsed value (char *) of the PKI_STRING (in UTF-8).

1.33.1.4 int PKI_STRING_get_type (PKI_STRING * s)

Returns the type of the PKI_STRING (PKI_STRING_IA5, etc.).

1.33.1.5 char* PKI_STRING_get_utf8 (PKI_STRING * s)

Returns the parsed value (char *) of the PKI_STRING (in UTF-8).

1.33.1.6 `PKI_STRING* PKI_STRING_new (int type, char * val, ssize_t size)`

Returns a new `PKI_STRING` of type and contents set from the passed parameters.

1.33.1.7 `PKI_STRING* PKI_STRING_new_null (int type)`

Create an empty `PKI_STRING` of type passed as the only argument.

1.33.1.8 `int PKI_STRING_print (PKI_STRING * s)`

Prints the contents of a `PKI_STRING` to Standard Output.

1.33.1.9 `int PKI_STRING_print_fp (FILE * fp, PKI_STRING * s)`

Prints the contents of a `PKI_STRING` to a `FILE` pointer (eg., stdout).

1.33.1.10 `int PKI_STRING_set (PKI_STRING * s, char * content, ssize_t size)`

Sets the content of a `PKI_STRING`.

1.34 `src/openssl/pki_time.c` File Reference

Include dependency graph for `pki_time.c`:

**Functions**

- `PKI_TIME * PKI_TIME_dup (PKI_TIME *time)`
Returns a duplicate of the `PKI_TIME` object.
- `int PKI_TIME_free (PKI_TIME *time)`
- `void PKI_TIME_free_void (void *time)`
- `char * PKI_TIME_get_parsed (PKI_TIME *t)`
- `PKI_TIME * PKI_TIME_new (long offset)`
- `int PKI_TIME_print (PKI_TIME *time)`
- `int PKI_TIME_print_fp (FILE *fp, PKI_TIME *time)`

1.34.1 **Function Documentation****1.34.1.1** `PKI_TIME* PKI_TIME_dup (PKI_TIME * time)`

Returns a duplicate of the `PKI_TIME` object.

1.34.1.2 `int PKI_TIME_free (PKI_TIME * time)`**1.34.1.3** `void PKI_TIME_free_void (void * time)`

1.34.1.4 char* **PKI_TIME_get_parsed** (PKI_TIME * *t*)

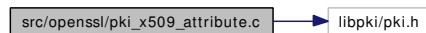
1.34.1.5 PKI_TIME* **PKI_TIME_new** (long *offset*)

1.34.1.6 int **PKI_TIME_print** (PKI_TIME * *time*)

1.34.1.7 int **PKI_TIME_print_fp** (FILE * *fp*, PKI_TIME * *time*)

1.35 src/openssl/pki_x509_attribute.c File Reference

Include dependency graph for pki_x509_attribute.c:



Functions

- int **PKI_STACK_X509_ATTRIBUTE_add** (PKI_X509_ATTRIBUTE_STACK *a_sk, PKI_X509_ATTRIBUTE *a)
- int **PKI_STACK_X509_ATTRIBUTE_delete** (PKI_X509_ATTRIBUTE_STACK *a_sk, PKI_ID attr)
- int **PKI_STACK_X509_ATTRIBUTE_delete_by_name** (PKI_X509_ATTRIBUTE_STACK *a_sk, char *name)
- int **PKI_STACK_X509_ATTRIBUTE_delete_by_num** (PKI_X509_ATTRIBUTE_STACK *a_sk, int num)
- void **PKI_STACK_X509_ATTRIBUTE_free** (PKI_X509_ATTRIBUTE_STACK *sk)

Frees the memory associated with a stack of PKI_X509_ATTRIBUTE.

- void **PKI_STACK_X509_ATTRIBUTE_free_all** (PKI_X509_ATTRIBUTE_STACK *sk)
- PKI_X509_ATTRIBUTE * **PKI_STACK_X509_ATTRIBUTE_get** (PKI_X509_ATTRIBUTE_STACK *a_sk, PKI_ID attribute_id)
- PKI_X509_ATTRIBUTE * **PKI_STACK_X509_ATTRIBUTE_get_by_name** (PKI_X509_ATTRIBUTE_STACK *a_sk, char *name)
- PKI_X509_ATTRIBUTE * **PKI_STACK_X509_ATTRIBUTE_get_by_num** (PKI_X509_ATTRIBUTE_STACK *a_sk, int num)
- int **PKI_STACK_X509_ATTRIBUTE_num** (PKI_X509_ATTRIBUTE_STACK *a_sk)
- int **PKI_STACK_X509_ATTRIBUTE_replace** (PKI_X509_ATTRIBUTE_STACK *a_sk, PKI_X509_ATTRIBUTE *a)
- void **PKI_X509_ATTRIBUTE_free** (PKI_X509_ATTRIBUTE *a)

Frees the memory associated with a PKI_X509_ATTRIBUTE.

- void **PKI_X509_ATTRIBUTE_free_null** (void *a)
- const char * **PKI_X509_ATTRIBUTE_get_descr** (PKI_X509_ATTRIBUTE *a)
- char * **PKI_X509_ATTRIBUTE_get_parsed** (PKI_X509_ATTRIBUTE *a)
- PKI_STRING * **PKI_X509_ATTRIBUTE_get_value** (PKI_X509_ATTRIBUTE *a)
- PKI_X509_ATTRIBUTE * **PKI_X509_ATTRIBUTE_new** (PKI_ID attribute_id, int data_type, unsigned char *value, size_t size)
- PKI_X509_ATTRIBUTE * **PKI_X509_ATTRIBUTE_new_name** (char *name, int data_type, char *value, size_t size)

Returns a `PKI_X509_ATTRIBUTE` from a string description.

- `PKI_X509_ATTRIBUTE * PKI_X509_ATTRIBUTE_new_null` (void)

Returns an empty `PKI_X509_ATTRIBUTE`.

1.35.1 Function Documentation

1.35.1.1 `int PKI_STACK_X509_ATTRIBUTE_add (PKI_X509_ATTRIBUTE_STACK * a_sk, PKI_X509_ATTRIBUTE * a)`

1.35.1.2 `int PKI_STACK_X509_ATTRIBUTE_delete (PKI_X509_ATTRIBUTE_STACK * a_sk, PKI_ID attr)`

1.35.1.3 `int PKI_STACK_X509_ATTRIBUTE_delete_by_name (PKI_X509_ATTRIBUTE_STACK * a_sk, char * name)`

1.35.1.4 `int PKI_STACK_X509_ATTRIBUTE_delete_by_num (PKI_X509_ATTRIBUTE_STACK * a_sk, int num)`

1.35.1.5 `void PKI_STACK_X509_ATTRIBUTE_free (PKI_X509_ATTRIBUTE_STACK * sk)`

Frees the memory associated with a stack of `PKI_X509_ATTRIBUTE`.

1.35.1.6 `void PKI_STACK_X509_ATTRIBUTE_free_all (PKI_X509_ATTRIBUTE_STACK * sk)`

1.35.1.7 `PKI_X509_ATTRIBUTE* PKI_STACK_X509_ATTRIBUTE_get (PKI_X509_ATTRIBUTE_STACK * a_sk, PKI_ID attribute_id)`

1.35.1.8 `PKI_X509_ATTRIBUTE* PKI_STACK_X509_ATTRIBUTE_get_by_name (PKI_X509_ATTRIBUTE_STACK * a_sk, char * name)`

1.35.1.9 `PKI_X509_ATTRIBUTE* PKI_STACK_X509_ATTRIBUTE_get_by_num (PKI_X509_ATTRIBUTE_STACK * a_sk, int num)`

1.35.1.10 `int PKI_STACK_X509_ATTRIBUTE_num (PKI_X509_ATTRIBUTE_STACK * a_sk)`

1.35.1.11 `int PKI_STACK_X509_ATTRIBUTE_replace (PKI_X509_ATTRIBUTE_STACK * a_sk, PKI_X509_ATTRIBUTE * a)`

1.35.1.12 `void PKI_X509_ATTRIBUTE_free (PKI_X509_ATTRIBUTE * a)`

Frees the memory associated with a `PKI_X509_ATTRIBUTE`.

1.35.1.13 void `PKI_X509_ATTRIBUTE_free_null` (void * *a*)

1.35.1.14 const char* `PKI_X509_ATTRIBUTE_get_descr` (PKI_X509_ATTRIBUTE * *a*)

1.35.1.15 char* `PKI_X509_ATTRIBUTE_get_parsed` (PKI_X509_ATTRIBUTE * *a*)

1.35.1.16 PKI_STRING* `PKI_X509_ATTRIBUTE_get_value` (PKI_X509_ATTRIBUTE * *a*)

1.35.1.17 PKI_X509_ATTRIBUTE* `PKI_X509_ATTRIBUTE_new` (PKI_ID *attribute_id*, int *data_type*, unsigned char * *value*, size_t *size*)

1.35.1.18 PKI_X509_ATTRIBUTE* `PKI_X509_ATTRIBUTE_new_name` (char * *name*, int *data_type*, char * *value*, size_t *size*)

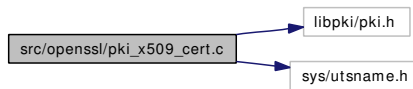
Returns a PKI_X509_ATTRIBUTE from a string description.

1.35.1.19 PKI_X509_ATTRIBUTE* `PKI_X509_ATTRIBUTE_new_null` (void)

Returns an empty PKI_X509_ATTRIBUTE.

1.36 src/openssl/pki_x509_cert.c File Reference

Include dependency graph for pki_x509_cert.c:



Functions

- int `PKI_X509_CERT_add_extension` (PKI_X509_CERT **x*, PKI_X509_EXTENSION **ext*)
Adds a specific extension to a certificate.
- int `PKI_X509_CERT_add_extension_stack` (PKI_X509_CERT **x*, PKI_X509_EXTENSION_STACK **ext*)
Adds a stack of extensions to a certificate object.
- int `PKI_X509_CERT_check_domain` (PKI_X509_CERT **x*, char **domain*)
Returns PKI_OK if a certificate is allowed to be used with the passed domain name.
- PKI_X509_CERT * `PKI_X509_CERT_dup` (PKI_X509_CERT **x*)
Returns a copy of the PKI_X509_CERT structure.
- PKI_DIGEST * `PKI_X509_CERT_fingerprint` (PKI_X509_CERT **x*, PKI_DIGEST_ALG **alg*)
Calculates the fingerprint over a certificate by using the passed digest algorithm identifier.
- PKI_DIGEST * `PKI_X509_CERT_fingerprint_by_name` (PKI_X509_CERT **x*, char **alg*)

*Calculates the fingerprint over a certificate by using the passed digest string (char *) identifier.*

- void [PKI_X509_CERT_free](#) (PKI_X509_CERT *x)
Frees the memory associated with a certificate.
- void [PKI_X509_CERT_free_void](#) (void *x)
- PKI_STACK * [PKI_X509_CERT_get_cdp](#) (PKI_X509_CERT *x)
Returns the stack of URL for the CRL Distribution Point(s).
- void * [PKI_X509_CERT_get_data](#) (PKI_X509_CERT *x, PKI_X509_DATA type)
Returns a pointer to a specified data field in a certificate.
- char ** [PKI_X509_CERT_get_email](#) (PKI_X509_CERT *x)
Returns the list of subject's email addresses embedded in the cert.
- PKI_X509_EXTENSION * [PKI_X509_CERT_get_extension_by_id](#) (PKI_X509_CERT *x, PKI_ID num)
- PKI_X509_EXTENSION * [PKI_X509_CERT_get_extension_by_name](#) (PKI_X509_CERT *x, char *name)
- PKI_X509_EXTENSION * [PKI_X509_CERT_get_extension_by_oid](#) (PKI_X509_CERT *x, PKI_OID *id)
- PKI_X509_EXTENSION_STACK * [PKI_X509_CERT_get_extensions](#) (PKI_X509_CERT *x)
- int [PKI_X509_CERT_get_keysize](#) (PKI_X509_CERT *x)
Returns the size of the certificate public key.
- char * [PKI_X509_CERT_get_parsed](#) (PKI_X509_CERT *x, PKI_X509_DATA type)
*Returns a parsed (char *) representation of the requested data type (type).*
- PKI_X509_CERT_TYPE [PKI_X509_CERT_get_type](#) (PKI_X509_CERT *x)
Returns PKI_X509_CERT_TYPE (an int) with the type of certificate.
- int [PKI_X509_CERT_is_ca](#) (PKI_X509_CERT *x)
Returns PKI_OK if a certificate is allowed to sign certs.
- int [PKI_X509_CERT_is_proxy](#) (PKI_X509_CERT *x)
Returns PKI_OK if a certificate is a Proxy Certificate.
- int [PKI_X509_CERT_is_selfsigned](#) (PKI_X509_CERT *x)
Returns PKI_OK if a certificate is self-signed, PKI_ERR otherwise.
- PKI_DIGEST * [PKI_X509_CERT_key_hash](#) (PKI_X509_CERT *x, PKI_DIGEST_ALG *alg)
Calculates the Hash of the Public Key of the certificate.
- PKI_DIGEST * [PKI_X509_CERT_key_hash_by_name](#) (PKI_X509_CERT *x, char *alg)
*Calculates the Hash of the Public Key of the certificate by using the hash algorithm passed as a (char *).*
- PKI_X509_CERT * [PKI_X509_CERT_new](#) (PKI_X509_CERT *ca_cert, PKI_X509_KEYPAIR *k, PKI_X509_REQ *req, char *subj_s, char *serial_s, unsigned long validity, PKI_X509_PROFILE *conf, PKI_ALGOR *algor, PKI_CONFIG *oids, HSM *hsm)
Generates a new certificate.

- `PKI_X509_CERT * PKI_X509_CERT_new_null` (void)
Returns an empty PKI_X509_CERT data structure.
- `int PKI_X509_CERT_print_parsed` (PKI_X509_CERT **x*, PKI_X509_DATA type, int *fd*)
Print the contents of a certificate in a text format to the file descriptor (fd).

Variables

- `int NID_proxyCertInfo`

1.36.1 Function Documentation

1.36.1.1 `int PKI_X509_CERT_add_extension` (PKI_X509_CERT **x*, PKI_X509_EXTENSION **ext*)

Adds a specific extension to a certificate.

1.36.1.2 `int PKI_X509_CERT_add_extension_stack` (PKI_X509_CERT **x*, PKI_X509_EXTENSION_STACK **ext*)

Adds a stack of extensions to a certificate object.

1.36.1.3 `int PKI_X509_CERT_check_domain` (PKI_X509_CERT **x*, char **domain*)

Returns PKI_OK if a certificate is allowed to be used with the passed domain name.

1.36.1.4 `PKI_X509_CERT* PKI_X509_CERT_dup` (PKI_X509_CERT **x*)

Returns a copy of the PKI_X509_CERT structure.

1.36.1.5 `PKI_DIGEST* PKI_X509_CERT_fingerprint` (PKI_X509_CERT **x*, PKI_DIGEST_ALG **alg*)

Calculates the fingerprint over a certificate by using the passed digest algorithm identifier.

1.36.1.6 `PKI_DIGEST* PKI_X509_CERT_fingerprint_by_name` (PKI_X509_CERT **x*, char **alg*)

Calculates the fingerprint over a certificate by using the passed digest string (char *) identifier.

1.36.1.7 `void PKI_X509_CERT_free` (PKI_X509_CERT **x*)

Frees the memory associated with a certificate.

1.36.1.8 `void PKI_X509_CERT_free_void` (void **x*)

1.36.1.9 `PKI_STACK* PKI_X509_CERT_get_cdp` (PKI_X509_CERT **x*)

Returns the stack of URL for the CRL Distribution Point(s).

1.36.1.10 void* PKI_X509_CERT_get_data (PKI_X509_CERT * *x*, PKI_X509_DATA *type*)

Returns a pointer to a specified data field in a certificate.

1.36.1.11 char PKI_X509_CERT_get_email (PKI_X509_CERT * *x*)**

Returns the list of subject's email addresses embedded in the cert.

1.36.1.12 PKI_X509_EXTENSION* PKI_X509_CERT_get_extension_by_id (PKI_X509_CERT * *x*, PKI_ID *num*)**1.36.1.13 PKI_X509_EXTENSION* PKI_X509_CERT_get_extension_by_name (PKI_X509_CERT * *x*, char * *name*)****1.36.1.14 PKI_X509_EXTENSION* PKI_X509_CERT_get_extension_by_oid (PKI_X509_CERT * *x*, PKI_OID * *id*)****1.36.1.15 PKI_X509_EXTENSION_STACK* PKI_X509_CERT_get_extensions (PKI_X509_CERT * *x*)****1.36.1.16 int PKI_X509_CERT_get_keysize (PKI_X509_CERT * *x*)**

Returns the size of the certificate public key.

1.36.1.17 char* PKI_X509_CERT_get_parsed (PKI_X509_CERT * *x*, PKI_X509_DATA *type*)

Returns a parsed (char *) representation of the requested data type (type).

1.36.1.18 PKI_X509_CERT_TYPE PKI_X509_CERT_get_type (PKI_X509_CERT * *x*)

Returns PKI_X509_CERT_TYPE (an int) with the type of certificate.

1.36.1.19 int PKI_X509_CERT_is_ca (PKI_X509_CERT * *x*)

Returns PKI_OK if a certificate is allowed to sign certs.

1.36.1.20 int PKI_X509_CERT_is_proxy (PKI_X509_CERT * *x*)

Returns PKI_OK if a certificate is a Proxy Certificate.

1.36.1.21 int PKI_X509_CERT_is_selfsigned (PKI_X509_CERT * *x*)

Returns PKI_OK if a certificate is self-signed, PKI_ERR otherwise.

1.36.1.22 PKI_DIGEST* PKI_X509_CERT_key_hash (PKI_X509_CERT * *x*, PKI_DIGEST_ALG * *alg*)

Calculates the Hash of the Public Key of the certificate.

1.36.1.23 `PKI_DIGEST*` `PKI_X509_CERT_key_hash_by_name` (`PKI_X509_CERT * x`, `char * alg`)

Calculates the Hash of the Public Key of the certificate by using the hash algorithm passed as a (`char *`).

1.36.1.24 `PKI_X509_CERT*` `PKI_X509_CERT_new` (`PKI_X509_CERT * ca_cert`, `PKI_X509_KEYPAIR * k`, `PKI_X509_REQ * req`, `char * subj_s`, `char * serial_s`, unsigned long *validity*, `PKI_X509_PROFILE * conf`, `PKI_ALGOR * algor`, `PKI_CONFIG * oids`, `HSM * hsm`)

Generates a new certificate.

1.36.1.25 `PKI_X509_CERT*` `PKI_X509_CERT_new_null` (`void`)

Returns an empty `PKI_X509_CERT` data structure.

1.36.1.26 `int` `PKI_X509_CERT_print_parsed` (`PKI_X509_CERT * x`, `PKI_X509_DATA type`, `int fd`)

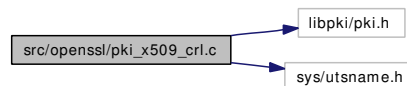
Print the contents of a certificate in a text format to the file descriptor (`fd`).

1.36.2 Variable Documentation

1.36.2.1 `int` `NID_proxyCertInfo`

1.37 src/openssl/pki_x509_crl.c File Reference

Include dependency graph for `pki_x509_crl.c`:



Functions

- `int` `PKI_X509_CRL_add_extension` (`PKI_X509_CRL *x`, `PKI_X509_EXTENSION *ext`)
Adds an Extension to a CRL object.
- `int` `PKI_X509_CRL_add_extension_stack` (`PKI_X509_CRL *x`, `PKI_X509_EXTENSION_STACK *ext`)
Adds a stack of extensions to a CRL object.
- `int` `PKI_X509_CRL_ENTRY_free` (`PKI_X509_CRL_ENTRY *entry`)
Frees a `PKI_X509_CRL_ENTRY`.
- `void` `PKI_X509_CRL_ENTRY_free_void` (`void *entry`)
- `PKI_X509_CRL_ENTRY *` `PKI_X509_CRL_ENTRY_new` (`PKI_X509_CERT *cert`, `PKI_X509_CRL_REASON reason`, `PKI_TIME *revDate`, `PKI_X509_PROFILE *profile`)
Generates a new `PKI_X509_CRL_ENTRY` from a certificate.

- `PKI_X509_CRL_ENTRY * PKI_X509_CRL_ENTRY_new_serial` (`char *serial`, `PKI_X509_CRL_REASON reason`, `PKI_TIME *revDate`, `PKI_X509_PROFILE *profile`)
Generates a new PKI_X509_CRL_ENTRY from a serial number (string).
- `int PKI_X509_CRL_free` (`PKI_X509_CRL *x`)
Frees memory associated with a PKI_CRL.
- `void PKI_X509_CRL_free_void` (`void *x`)
- `void * PKI_X509_CRL_get_data` (`PKI_X509_CRL *x`, `PKI_X509_DATA type`)
Get Data from a CRL object.
- `char * PKI_X509_CRL_get_parsed` (`PKI_X509_CRL *x`, `PKI_X509_DATA type`)
- `PKI_X509_CRL_ENTRY * PKI_X509_CRL_lookup` (`PKI_X509_CRL *x`, `PKI_INTEGER *s`)
Find an entry within a CRL by using the PKI_INTEGER serial number of the certificate.
- `PKI_X509_CRL_ENTRY * PKI_X509_CRL_lookup_cert` (`PKI_X509_CRL *x`, `PKI_X509_CERT *cert`)
Find an entry in a CRL by using a certificate.
- `PKI_X509_CRL_ENTRY * PKI_X509_CRL_lookup_long` (`PKI_X509_CRL *x`, `long long s`)
Lookup for a serial (long long) in a PKI_X509_CRL and returns the PKI_X509_CRL_ENTRY entry if found.
- `PKI_X509_CRL_ENTRY * PKI_X509_CRL_lookup_serial` (`PKI_X509_CRL *x`, `char *serial`)
Find an entry within a CRL.
- `PKI_X509_CRL * PKI_X509_CRL_new` (`PKI_X509_KEYPAIR *k`, `PKI_X509_CERT *cert`, `char *serial_s`, `unsigned long validity`, `PKI_X509_CRL_ENTRY_STACK *sk`, `PKI_X509_PROFILE *profile`, `PKI_CONFIG *oids`, `HSM *hsm`)
Generate a new CRL from a stack of revoked entries.
- `int PKI_X509_CRL_print_parsed` (`PKI_X509_CRL *x`, `PKI_X509_DATA type`, `int fd`)

1.37.1 Function Documentation

1.37.1.1 `int PKI_X509_CRL_add_extension` (`PKI_X509_CRL *x`, `PKI_X509_EXTENSION *ext`)

Adds an Extension to a CRL object.

1.37.1.2 `int PKI_X509_CRL_add_extension_stack` (`PKI_X509_CRL *x`, `PKI_X509_EXTENSION_STACK *ext`)

Adds a stack of extensions to a CRL object.

1.37.1.3 `int PKI_X509_CRL_ENTRY_free` (`PKI_X509_CRL_ENTRY *entry`)

Frees a `PKI_X509_CRL_ENTRY`.

Frees memory associated to a `PKI_X509_CRL_ENTRY`

1.37.1.4 `void PKI_X509_CRL_ENTRY_free_void` (`void *entry`)

1.37.1.5 PKI_X509_CRL_ENTRY* PKI_X509_CRL_ENTRY_new (PKI_X509_CERT * *cert*, PKI_X509_CRL_REASON *reason*, PKI_TIME * *revDate*, PKI_X509_PROFILE * *profile*)

Generates a new PKI_X509_CRL_ENTRY from a certificate.

This function generates a new PKI_X509_CRL_ENTRY starting from a certificate. The new structure can be added to a PKI_X509_CRL_ENTRY_STACK structure in order to generate a new CRL.

1.37.1.6 PKI_X509_CRL_ENTRY* PKI_X509_CRL_ENTRY_new_serial (char * *serial*, PKI_X509_CRL_REASON *reason*, PKI_TIME * *revDate*, PKI_X509_PROFILE * *profile*)

Generates a new PKI_X509_CRL_ENTRY from a serial number (string).

This function generates a new PKI_X509_CRL_ENTRY starting from a string representing the serial number of the entry. The optional profile is used to add extensions to the entry (when needed)

1.37.1.7 int PKI_X509_CRL_free (PKI_X509_CRL * *x*)

Frees memory associated with a PKI_CRL.

This function frees memory associated with a PKI_X509_CRL data structure. Returns PKI_OK if successful or PKI_ERR in case of an error (or if the passed pointer was NULL).

1.37.1.8 void PKI_X509_CRL_free_void (void * *x*)**1.37.1.9 void* PKI_X509_CRL_get_data (PKI_X509_CRL * *x*, PKI_X509_DATA *type*)**

Get Data from a CRL object.

1.37.1.10 char* PKI_X509_CRL_get_parsed (PKI_X509_CRL * *x*, PKI_X509_DATA *type*)**1.37.1.11 PKI_X509_CRL_ENTRY* PKI_X509_CRL_lookup (PKI_X509_CRL * *x*, PKI_INTEGER * *s*)**

Find an entry within a CRL by using the PKI_INTEGER serial number of the certificate.

This function look inside a CRL and returns the PKI_X509_CRL_ENTRY if the passed serial is found. The returned pointer refers to the internal CRL structure, when the CRL is freed the pointer will no more point to a valid memory area.

1.37.1.12 PKI_X509_CRL_ENTRY* PKI_X509_CRL_lookup_cert (PKI_X509_CRL * *x*, PKI_X509_CERT * *cert*)

Find an entry in a CRL by using a certificate.

Use the information within the certificate to look for a corresponding entry in the CRL. If found, the PKI_X509_CRL_ENTRY structure is copied and returned. The returned pointer refers to the internal CRL structure, when the CRL is freed the pointer will no more point to a valid memory area.

1.37.1.13 PKI_X509_CRL_ENTRY* PKI_X509_CRL_lookup_long (PKI_X509_CRL * *x*, long *s*)

Lookup for a serial (long long) in a PKI_X509_CRL and returns the PKI_X509_CRL_ENTRY entry if found.

1.37.1.14 `PKI_X509_CRL_ENTRY*` `PKI_X509_CRL_lookup_serial` (`PKI_X509_CRL * x`, `char * serial`)

Find an entry within a CRL.

This function look inside a CRL and returns the `PKI_X509_CRL_ENTRY` if the passed serial is found. The returned pointer refers to the internal CRL structure, when the CRL is freed the pointer will no more point to a valid memory area.

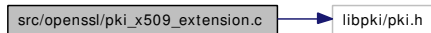
1.37.1.15 `PKI_X509_CRL*` `PKI_X509_CRL_new` (`PKI_X509_KEYPAIR * k`, `PKI_X509_CERT * cert`, `char * serial_s`, `unsigned long validity`, `PKI_X509_CRL_ENTRY_STACK * sk`, `PKI_X509_PROFILE * profile`, `PKI_CONFIG * oids`, `HSM * hsm`)

Generate a new CRL from a stack of revoked entries.

Generates a new signed CRL from a stack of revoked entries. A profile is used to set the right extensions in the CRL. To generate a new revoked entry the `PKI_X509_CRL_ENTRY_new()` function has to be used.

1.37.1.16 `int` `PKI_X509_CRL_print_parsed` (`PKI_X509_CRL * x`, `PKI_X509_DATA type`, `int fd`)**1.38** `src/openssl/pki_x509_extension.c` File Reference

Include dependency graph for `pki_x509_extension.c`:

**Functions**

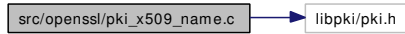
- void `PKI_X509_EXTENSION_free` (`PKI_X509_EXTENSION *ext`)
- void `PKI_X509_EXTENSION_free_void` (`void *ext`)
- `PKI_X509_EXTENSION_STACK *` `PKI_X509_EXTENSION_get_list` (`void *x`, `PKI_X509_DATA type`)
- `PKI_X509_EXTENSION *` `PKI_X509_EXTENSION_new` (`void`)
- `PKI_X509_EXTENSION *` `PKI_X509_EXTENSION_value_new_profile` (`PKI_X509_PROFILE *profile`, `PKI_CONFIG *oids`, `PKI_CONFIG_ELEMENT *extNode`)

1.38.1 **Function Documentation****1.38.1.1** `void` `PKI_X509_EXTENSION_free` (`PKI_X509_EXTENSION * ext`)**1.38.1.2** `void` `PKI_X509_EXTENSION_free_void` (`void * ext`)**1.38.1.3** `PKI_X509_EXTENSION_STACK*` `PKI_X509_EXTENSION_get_list` (`void * x`, `PKI_X509_DATA type`)**1.38.1.4** `PKI_X509_EXTENSION*` `PKI_X509_EXTENSION_new` (`void`)

1.38.1.5 `PKI_X509_EXTENSION*` `PKI_X509_EXTENSION_value_new_profile` (`PKI_X509_PROFILE *` *profile*, `PKI_CONFIG *` *oids*, `PKI_CONFIG_ELEMENT *` *extNode*)

1.39 src/openssl/pki_x509_name.c File Reference

Include dependency graph for `pki_x509_name.c`:



Functions

- `PKI_X509_NAME *` `PKI_X509_NAME_add` (`PKI_X509_NAME *` *name*, `const char *` *entry*)
Adds a new entry to the passed PKI_X509_NAME.
- `int` `PKI_X509_NAME_cmp` (`PKI_X509_NAME *` *a*, `PKI_X509_NAME *` *b*)
Returns 0 if the two names are the same, non-zero otherwise.
- `PKI_X509_NAME *` `PKI_X509_NAME_dup` (`PKI_X509_NAME *` *name*)
Returns a pointer to a duplicate of the passed name.
- `int` `PKI_X509_NAME_free` (`PKI_X509_NAME *` *name*)
- `PKI_DIGEST *` `PKI_X509_NAME_get_digest` (`PKI_X509_NAME *` *name*, `PKI_DIGEST_ALG *` *alg*)
Returns the digest of a PKI_X509_NAME.
- `PKI_X509_NAME_RDN **` `PKI_X509_NAME_get_list` (`PKI_X509_NAME *` *name*, `PKI_X509_NAME_TYPE` *filter*)
Returns a NULL terminated list of PKI_X509_NAME_RDN from the name.
- `char *` `PKI_X509_NAME_get_parsed` (`PKI_X509_NAME *` *name*)
*Returns a char * (utf8) representation of the name.*
- `void` `PKI_X509_NAME_list_free` (`PKI_X509_NAME_RDN **` *list*)
Frees the memory associated with a PKI_X509_NAME_RDN data st.
- `PKI_X509_NAME *` `PKI_X509_NAME_new` (`char *` *name*)
- `PKI_X509_NAME *` `PKI_X509_NAME_new_null` (`void`)
- `const char *` `PKI_X509_NAME_RDN_type_descr` (`PKI_X509_NAME_RDN *` *rdn*)
Returns the description of the type of an RDN.
- `PKI_X509_NAME_TYPE` `PKI_X509_NAME_RDN_type_id` (`PKI_X509_NAME_RDN *` *rdn*)
Returns the PKI_ID of a type of an RDN.
- `const char *` `PKI_X509_NAME_RDN_type_text` (`PKI_X509_NAME_RDN *` *rdn*)
Returns the text representation of the type of an RDN.
- `char *` `PKI_X509_NAME_RDN_value` (`PKI_X509_NAME_RDN *` *rdn*)
*Returns the value (char *) of an RDN.*

1.39.1 Function Documentation

1.39.1.1 **PKI_X509_NAME*** **PKI_X509_NAME_add** (**PKI_X509_NAME** * *name*, **const char** * *entry*)

Adds a new entry to the passed PKI_X509_NAME.

1.39.1.2 **int** **PKI_X509_NAME_cmp** (**PKI_X509_NAME** * *a*, **PKI_X509_NAME** * *b*)

Returns 0 if the two names are the same, non-zero otherwise.

1.39.1.3 **PKI_X509_NAME*** **PKI_X509_NAME_dup** (**PKI_X509_NAME** * *name*)

Returns a pointer to a duplicate of the passed name.

1.39.1.4 **int** **PKI_X509_NAME_free** (**PKI_X509_NAME** * *name*)

1.39.1.5 **PKI_DIGEST*** **PKI_X509_NAME_get_digest** (**PKI_X509_NAME** * *name*, **PKI_DIGEST_ALG** * *alg*)

Returns the digest of a PKI_X509_NAME.

1.39.1.6 **PKI_X509_NAME_RDN**** **PKI_X509_NAME_get_list** (**PKI_X509_NAME** * *name*, **PKI_X509_NAME_TYPE** *filter*)

Returns a NULL terminated list of PKI_X509_NAME_RDN from the name.

1.39.1.7 **char*** **PKI_X509_NAME_get_parsed** (**PKI_X509_NAME** * *name*)

Returns a char * (utf8) representation of the name.

1.39.1.8 **void** **PKI_X509_NAME_list_free** (**PKI_X509_NAME_RDN** ** *list*)

Frees the memory associated with a PKI_X509_NAME_RDN data st.

1.39.1.9 **PKI_X509_NAME*** **PKI_X509_NAME_new** (**char** * *name*)

1.39.1.10 **PKI_X509_NAME*** **PKI_X509_NAME_new_null** (**void**)

1.39.1.11 **const char*** **PKI_X509_NAME_RDN_type_descr** (**PKI_X509_NAME_RDN** * *rdn*)

Returns the description of the type of an RDN.

1.39.1.12 **PKI_X509_NAME_TYPE** **PKI_X509_NAME_RDN_type_id** (**PKI_X509_NAME_RDN** * *rdn*)

Returns the PKI_ID of a type of an RDN.

1.39.1.13 **const char*** **PKI_X509_NAME_RDN_type_text** (**PKI_X509_NAME_RDN** * *rdn*)

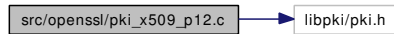
Returns the text representation of the type of an RDN.

1.39.1.14 char* PKI_X509_NAME_RDN_value (PKI_X509_NAME_RDN * rdn)

Returns the value (char *) of an RDN.

1.40 src/openssl/pki_x509_p12.c File Reference

Include dependency graph for pki_x509_p12.c:

**Enumerations**

- enum [bag_datatype_st](#) {
[BAG_DATATYPE_ALL](#) = 0, [BAG_DATATYPE_KEYPAIR](#), [BAG_DATATYPE_CERT](#), [BAG_DATATYPE_CACERT](#),
[BAG_DATATYPE_OTHERCERTS](#), [BAG_DATATYPE_UNKNOWN](#) }

Functions

- PKI_X509_PKCS12_VALUE * [PEM_read_bio_PKCS12](#) (PKI_IO *bp)
- int [PEM_write_bio_PKCS12](#) (PKI_IO *bp, PKI_X509_PKCS12_VALUE *o)
- int [PKI_X509_PKCS12_DATA_add_certs](#) (PKI_X509_PKCS12_DATA *data, PKI_X509_CERT *cert, PKI_X509_CERT *cacert, PKI_X509_CERT_STACK *trusted, PKI_CRED *cred)
Adds user certificate, cacertificate and trusted certs to P12.
- int [PKI_X509_PKCS12_DATA_add_keypair](#) (PKI_X509_PKCS12_DATA *data, PKI_X509_KEYPAIR *keypair, PKI_CRED *cred)
Adds a Keypair (LocalKey) to the PKCS12.
- int [PKI_X509_PKCS12_DATA_add_other_certs](#) (PKI_X509_PKCS12_DATA *data, PKI_X509_CERT_STACK *sk, PKI_CRED *cred)
Adds a 'generic' list of certs to P12.
- void [PKI_X509_PKCS12_DATA_free](#) (PKI_X509_PKCS12_DATA *p12_data)
- PKI_X509_PKCS12_DATA * [PKI_X509_PKCS12_DATA_new](#) (void)
Generates an empty PKI_X509_PKCS12_DATA object to be populated before using it to create a PKCS12.
- void [PKI_X509_PKCS12_free](#) (PKI_X509_PKCS12 *p12)
Releases the memory associated with a PKI_X509_PKCS12 object.
- void [PKI_X509_PKCS12_free_void](#) (void *p12)
- PKI_X509_CERT * [PKI_X509_PKCS12_get_cacert](#) (PKI_X509_PKCS12 *p12, PKI_CRED *cred)
Returns the CA cert present (if) in a PKI_X509_PKCS12 object.
- PKI_X509_CERT * [PKI_X509_PKCS12_get_cert](#) (PKI_X509_PKCS12 *p12, PKI_CRED *cred)
Returns the client (user) cert present in a PKI_X509_PKCS12 object.

- `PKI_X509_KEYPAIR * PKI_X509_PKCS12_get_keypair` (`PKI_X509_PKCS12 *p12`, `PKI_CRED *cred`)
Returns the keypair present in a PKI_X509_PKCS12 object.
- `PKI_X509_CERT_STACK * PKI_X509_PKCS12_get_otherCerts` (`PKI_X509_PKCS12 *p12`, `PKI_CRED *cred`)
Returns all the certs besides the CA and the user cert present (if) in a PKI_X509_PKCS12 object.
- `PKI_X509_PKCS12 * PKI_X509_PKCS12_new` (`PKI_X509_PKCS12_DATA *p12_data`, `PKI_CRED *cred`)
Generates a new PKI_X509_PKCS12 object from a PKI_X509_PKCS12_DATA obj.
- `PKI_X509_PKCS12 * PKI_X509_PKCS12_new_null` (`void`)
Allocates memory for a new PKI_X509_PKCS12 object.
- `int PKI_X509_PKCS12_TOKEN_export` (`PKI_TOKEN *tk`, `URL *url`, `int format`, `HSM *hsm`)
- `int PKI_X509_PKCS12_verify_cred` (`PKI_X509_PKCS12 *p12`, `PKI_CRED *cred`)
Verifies the MAC against the passed credentials.

1.40.1 Enumeration Type Documentation

1.40.1.1 enum `bag_datatype_st`

Enumerator:

`BAG_DATATYPE_ALL`
`BAG_DATATYPE_KEYPAIR`
`BAG_DATATYPE_CERT`
`BAG_DATATYPE_CACERT`
`BAG_DATATYPE_OTHERCERTS`
`BAG_DATATYPE_UNKNOWN`

1.40.2 Function Documentation

1.40.2.1 `PKI_X509_PKCS12_VALUE* PEM_read_bio_PKCS12` (`PKI_IO *bp`)

1.40.2.2 `int PEM_write_bio_PKCS12` (`PKI_IO *bp`, `PKI_X509_PKCS12_VALUE *o`)

1.40.2.3 `int PKI_X509_PKCS12_DATA_add_certs` (`PKI_X509_PKCS12_DATA *data`, `PKI_X509_CERT *cert`, `PKI_X509_CERT *cacert`, `PKI_X509_CERT_STACK *trusted`, `PKI_CRED *cred`)

Adds user certificate, cacertificate and trusted certs to P12.

1.40.2.4 `int PKI_X509_PKCS12_DATA_add_keypair` (`PKI_X509_PKCS12_DATA *data`, `PKI_X509_KEYPAIR *keypair`, `PKI_CRED *cred`)

Adds a Keypair (LocalKey) to the PKCS12.

1.40.2.5 `int PKI_X509_PKCS12_DATA_add_other_certs (PKI_X509_PKCS12_DATA * data, PKI_X509_CERT_STACK * sk, PKI_CRED * cred)`

Adds a 'generic' list of certs to P12.

1.40.2.6 `void PKI_X509_PKCS12_DATA_free (PKI_X509_PKCS12_DATA * p12_data)`

1.40.2.7 `PKI_X509_PKCS12_DATA* PKI_X509_PKCS12_DATA_new (void)`

Generates an empty PKI_X509_PKCS12_DATA object to be populated before using it to create a PKCS12.

1.40.2.8 `void PKI_X509_PKCS12_free (PKI_X509_PKCS12 * p12)`

Releases the memory associated with a PKI_X509_PKCS12 object.

1.40.2.9 `void PKI_X509_PKCS12_free_void (void * p12)`

1.40.2.10 `PKI_X509_CERT* PKI_X509_PKCS12_get_cacert (PKI_X509_PKCS12 * p12, PKI_CRED * cred)`

Returns the CA cert present (if) in a PKI_X509_PKCS12 object.

1.40.2.11 `PKI_X509_CERT* PKI_X509_PKCS12_get_cert (PKI_X509_PKCS12 * p12, PKI_CRED * cred)`

Returns the client (user) cert present in a PKI_X509_PKCS12 object.

1.40.2.12 `PKI_X509_KEYPAIR* PKI_X509_PKCS12_get_keypair (PKI_X509_PKCS12 * p12, PKI_CRED * cred)`

Returns the keypair present in a PKI_X509_PKCS12 object.

1.40.2.13 `PKI_X509_CERT_STACK* PKI_X509_PKCS12_get_otherCerts (PKI_X509_PKCS12 * p12, PKI_CRED * cred)`

Returns all the certs besides the CA and the user cert present (if) in a PKI_X509_PKCS12 object.

1.40.2.14 `PKI_X509_PKCS12* PKI_X509_PKCS12_new (PKI_X509_PKCS12_DATA * p12_data, PKI_CRED * cred)`

Generates a new PKI_X509_PKCS12 object from a PKI_X509_PKCS12_DATA obj.

1.40.2.15 `PKI_X509_PKCS12* PKI_X509_PKCS12_new_null (void)`

Allocates memory for a new PKI_X509_PKCS12 object.

1.40.2.16 `int PKI_X509_PKCS12_TOKEN_export (PKI_TOKEN * tk, URL * url, int format, HSM * hsm)`

1.40.2.17 int PKI_X509_PKCS12_verify_cred (PKI_X509_PKCS12 *p12, PKI_CRED *cred)

Verifies the MAC against the passed credentials.

1.41 src/openssl/pki_x509_pkcs7.c File Reference

Include dependency graph for pki_x509_pkcs7.c:



Functions

- int [PKI_X509_PKCS7_add_attribute](#) (PKI_X509_PKCS7 *p7, PKI_X509_ATTRIBUTE *a)
- int [PKI_X509_PKCS7_add_cert](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT *x)
Adds a certificate to the signature's certificate chain.
- int [PKI_X509_PKCS7_add_cert_stack](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT_STACK *x_sk)
Adds a stack of certificates to the signature's certificate chain.
- int [PKI_X509_PKCS7_add_crl](#) (PKI_X509_PKCS7 *p7, PKI_X509_CRL *crl)
- int [PKI_X509_PKCS7_add_crl_stack](#) (PKI_X509_PKCS7 *p7, PKI_X509_CRL_STACK *crl_sk)
- int [PKI_X509_PKCS7_add_recipient](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT *x)
Adds a new recipient for the PKI_X509_PKCS7.
- int [PKI_X509_PKCS7_add_signed_attribute](#) (PKI_X509_PKCS7 *p7, PKI_X509_ATTRIBUTE *a)
- int [PKI_X509_PKCS7_add_signer](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT *signer, PKI_X509_KEYPAIR *k, PKI_DIGEST_ALG *md)
Signs a PKI_X509_PKCS7 (must be of SIGNED type).
- int [PKI_X509_PKCS7_add_signer_tk](#) (PKI_X509_PKCS7 *p7, PKI_TOKEN *tk, PKI_DIGEST_ALG *md)
Returns a signed version of the PKI_X509_PKCS7 by using the passed token.
- int [PKI_X509_PKCS7_clear_certs](#) (PKI_X509_PKCS7 *p7)
Clears the chain of certificate for the signer.
- PKI_MEM * [PKI_X509_PKCS7_decode](#) (PKI_X509_PKCS7 *p7, PKI_X509_KEYPAIR *k, PKI_X509_CERT *x)
Decrypts the data from a (must) encrypted PKI_X509_PKCS7.
- int [PKI_X509_PKCS7_delete_attribute](#) (PKI_X509_PKCS7 *p7, PKI_ID id)
Deletes an attribute (id) from a PKI_X509_PKCS7.
- int [PKI_X509_PKCS7_delete_signed_attribute](#) (PKI_X509_PKCS7 *p7, PKI_ID id)
Deletes a signed attribute (id) from a PKI_X509_PKCS7.
- int [PKI_X509_PKCS7_encode](#) (PKI_X509_PKCS7 *p7, unsigned char *data, size_t size)

Encode a PKI_X509_PKCS7 by performing sign/encrypt operation.

- void [PKI_X509_PKCS7_free](#) (PKI_X509_PKCS7 *p7)
- void [PKI_X509_PKCS7_free_void](#) (void *p7)
- PKI_X509_ATTRIBUTE * [PKI_X509_PKCS7_get_attribute](#) (PKI_X509_PKCS7 *p7, PKI_ID id)
- PKI_X509_ATTRIBUTE * [PKI_X509_PKCS7_get_attribute_by_name](#) (PKI_X509_PKCS7 *p7, char *name)
- PKI_X509_CERT * [PKI_X509_PKCS7_get_cert](#) (PKI_X509_PKCS7 *p7, int idx)

Returns the n-th cert from a signed/signed&enc PKCS7.

- int [PKI_X509_PKCS7_get_certs_num](#) (PKI_X509_PKCS7 *p7)

Returns the number of certificates present in the signature chain.

- PKI_X509_CERT * [PKI_X509_PKCS7_get_crl](#) (PKI_X509_PKCS7 *p7, int idx)

Returns a copy of the n-th CRL from the signature.

- int [PKI_X509_PKCS7_get_crls_num](#) (PKI_X509_PKCS7 *p7)

Returns the number of CRLs present in the signature.

- PKI_MEM * [PKI_X509_PKCS7_get_data](#) (PKI_X509_PKCS7 *p7, PKI_X509_KEYPAIR *k, PKI_X509_CERT *x)

Decrypts (if needed) and returns the data from a PKI_X509_PKCS7.

- PKI_MEM * [PKI_X509_PKCS7_get_data_tk](#) (PKI_X509_PKCS7 *p7, PKI_TOKEN *tk)

Decrypts (if needed) and returns the idata from a PKI_X509_PKCS7 by using keypair and, if present, cert of the PKI_TOKEN argument.

- PKI_ALGOR * [PKI_X509_PKCS7_get_encode_alg](#) (PKI_X509_PKCS7 *p7)

Returns the encryption algorithm.

- PKI_MEM * [PKI_X509_PKCS7_get_raw_data](#) (PKI_X509_PKCS7 *p7)

Returns the raw data contained in a PKI_X509_PKCS7 (any type).

- PKI_X509_CERT * [PKI_X509_PKCS7_get_recipient_cert](#) (PKI_X509_PKCS7 *p7, int idx)

Returns a copy of the n-th recipient certificate.

- PKCS7_RECIP_INFO * [PKI_X509_PKCS7_get_recipient_info](#) (PKI_X509_PKCS7 *p7, int idx)

- int [PKI_X509_PKCS7_get_recipients_num](#) (PKI_X509_PKCS7 *p7)

Returns the number of recipients.

- PKI_X509_ATTRIBUTE * [PKI_X509_PKCS7_get_signed_attribute](#) (PKI_X509_PKCS7 *p7, PKI_ID id)

- PKI_X509_ATTRIBUTE * [PKI_X509_PKCS7_get_signed_attribute_by_name](#) (PKI_X509_PKCS7 *p7, char *name)

- PKCS7_SIGNER_INFO * [PKI_X509_PKCS7_get_signer_info](#) (PKI_X509_PKCS7 *p7, int idx)

- int [PKI_X509_PKCS7_get_signers_num](#) (PKI_X509_PKCS7 *p7)

Returns the number of signers.

- PKI_X509_PKCS7_TYPE [PKI_X509_PKCS7_get_type](#) (PKI_X509_PKCS7 *p7)

Returns the type of the PKI_X509_PKCS7 data (see PKI_X509_PKCS7_TYPE).

- int [PKI_X509_PKCS7_has_recipients](#) (PKI_X509_PKCS7 *p7)
Returns PKI_OK if the p7 has recipients already set, PKI_ERR otherwise.
- int [PKI_X509_PKCS7_has_signers](#) (PKI_X509_PKCS7 *p7)
Returns PKI_OK if the p7 has signers already set, PKI_ERR otherwise.
- PKI_X509_PKCS7 * [PKI_X509_PKCS7_new](#) (PKI_X509_PKCS7_TYPE type)
- int [PKI_X509_PKCS7_set_cipher](#) (PKI_X509_PKCS7 *p7, PKI_CIPHER *cipher)
Set the cipher in a encrypted (or signed and encrypted) PKCS7.
- int [PKI_X509_PKCS7_set_recipients](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT_STACK *x_sk)
Sets the recipients for a PKI_X509_PKCS7.
- int [PKI_X509_PKCS7_VALUE_print_bio](#) (PKI_IO *bio, PKI_X509_PKCS7_VALUE *p7val)

1.41.1 Function Documentation

1.41.1.1 int [PKI_X509_PKCS7_add_attribute](#) (PKI_X509_PKCS7 *p7, PKI_X509_ATTRIBUTE *a)

1.41.1.2 int [PKI_X509_PKCS7_add_cert](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT *x)

Adds a certificate to the signature's certificate chain.

1.41.1.3 int [PKI_X509_PKCS7_add_cert_stack](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT_STACK *x_sk)

Adds a stack of certificates to the signature's certificate chain.

1.41.1.4 int [PKI_X509_PKCS7_add_crl](#) (PKI_X509_PKCS7 *p7, PKI_X509_CRL *crl)

1.41.1.5 int [PKI_X509_PKCS7_add_crl_stack](#) (PKI_X509_PKCS7 *p7, PKI_X509_CRL_STACK *crl_sk)

1.41.1.6 int [PKI_X509_PKCS7_add_recipient](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT *x)

Adds a new recipient for the PKI_X509_PKCS7.

1.41.1.7 int [PKI_X509_PKCS7_add_signed_attribute](#) (PKI_X509_PKCS7 *p7, PKI_X509_ATTRIBUTE *a)

1.41.1.8 int [PKI_X509_PKCS7_add_signer](#) (PKI_X509_PKCS7 *p7, PKI_X509_CERT *signer, PKI_X509_KEYPAIR *k, PKI_DIGEST_ALG *md)

Signs a PKI_X509_PKCS7 (must be of SIGNED type).

1.41.1.9 int PKI_X509_PKCS7_add_signer_tk (PKI_X509_PKCS7 * *p7*, PKI_TOKEN * *tk*, PKI_DIGEST_ALG * *md*)

Returns a signed version of the PKI_X509_PKCS7 by using the passed token.

1.41.1.10 int PKI_X509_PKCS7_clear_certs (PKI_X509_PKCS7 * *p7*)

Clears the chain of certificate for the signer.

1.41.1.11 PKI_MEM* PKI_X509_PKCS7_decode (PKI_X509_PKCS7 * *p7*, PKI_X509_KEYPAIR * *k*, PKI_X509_CERT * *x*)

Decrypts the data from a (must) encrypted PKI_X509_PKCS7.

1.41.1.12 int PKI_X509_PKCS7_delete_attribute (PKI_X509_PKCS7 * *p7*, PKI_ID *id*)

Deletes an attribute (*id*) from a PKI_X509_PKCS7.

1.41.1.13 int PKI_X509_PKCS7_delete_signed_attribute (PKI_X509_PKCS7 * *p7*, PKI_ID *id*)

Deletes a signed attribute (*id*) from a PKI_X509_PKCS7.

1.41.1.14 int PKI_X509_PKCS7_encode (PKI_X509_PKCS7 * *p7*, unsigned char * *data*, size_t *size*)

Encode a PKI_X509_PKCS7 by performing sign/encrypt operation.

1.41.1.15 void PKI_X509_PKCS7_free (PKI_X509_PKCS7 * *p7*)**1.41.1.16 void PKI_X509_PKCS7_free_void (void * *p7*)****1.41.1.17 PKI_X509_ATTRIBUTE* PKI_X509_PKCS7_get_attribute (PKI_X509_PKCS7 * *p7*, PKI_ID *id*)****1.41.1.18 PKI_X509_ATTRIBUTE* PKI_X509_PKCS7_get_attribute_by_name (PKI_X509_PKCS7 * *p7*, char * *name*)****1.41.1.19 PKI_X509_CERT* PKI_X509_PKCS7_get_cert (PKI_X509_PKCS7 * *p7*, int *idx*)**

Returns the *n*-th cert from a signed/sign&enc PKCS7.

1.41.1.20 int PKI_X509_PKCS7_get_certs_num (PKI_X509_PKCS7 * *p7*)

Returns the number of certificates present in the signature chain.

1.41.1.21 PKI_X509_CERT* PKI_X509_PKCS7_get_crl (PKI_X509_PKCS7 * *p7*, int *idx*)

Returns a copy of the *n*-th CRL from the signature.

1.41.1.22 int PKI_X509_PKCS7_get_crls_num (PKI_X509_PKCS7 * *p7*)

Returns the number of CRLs present in the signature.

1.41.1.23 PKI_MEM* PKI_X509_PKCS7_get_data (PKI_X509_PKCS7 * *p7*, PKI_X509_KEYPAIR * *k*, PKI_X509_CERT * *x*)

Decrypts (if needed) and returns the data from a PKI_X509_PKCS7.

1.41.1.24 PKI_MEM* PKI_X509_PKCS7_get_data_tk (PKI_X509_PKCS7 * *p7*, PKI_TOKEN * *tk*)

Decrypts (if needed) and returns the idata from a PKI_X509_PKCS7 by using keypair and, if present, cert of the PKI_TOKEN argument.

1.41.1.25 PKI_ALGOR* PKI_X509_PKCS7_get_encode_alg (PKI_X509_PKCS7 * *p7*)

Returns the encryption algorithm.

1.41.1.26 PKI_MEM* PKI_X509_PKCS7_get_raw_data (PKI_X509_PKCS7 * *p7*)

Returns the raw data contained in a PKI_X509_PKCS7 (any type).

1.41.1.27 PKI_X509_CERT* PKI_X509_PKCS7_get_recipient_cert (PKI_X509_PKCS7 * *p7*, int *idx*)

Returns a copy of the n-th recipient certificate.

1.41.1.28 PKCS7_RECIP_INFO* PKI_X509_PKCS7_get_recipient_info (PKI_X509_PKCS7 * *p7*, int *idx*)**1.41.1.29 int PKI_X509_PKCS7_get_recipients_num (PKI_X509_PKCS7 * *p7*)**

Returns the number of recipients.

1.41.1.30 PKI_X509_ATTRIBUTE* PKI_X509_PKCS7_get_signed_attribute (PKI_X509_PKCS7 * *p7*, PKI_ID *id*)**1.41.1.31 PKI_X509_ATTRIBUTE* PKI_X509_PKCS7_get_signed_attribute_by_name (PKI_X509_PKCS7 * *p7*, char * *name*)****1.41.1.32 PKCS7_SIGNER_INFO* PKI_X509_PKCS7_get_signer_info (PKI_X509_PKCS7 * *p7*, int *idx*)****1.41.1.33 int PKI_X509_PKCS7_get_signers_num (PKI_X509_PKCS7 * *p7*)**

Returns the number of signers.

1.41.1.34 PKI_X509_PKCS7_TYPE PKI_X509_PKCS7_get_type (PKI_X509_PKCS7 * *p7*)

Returns the type of the PKI_X509_PKCS7 data (see PKI_X509_PKCS7_TYPE).

1.41.1.35 int PKI_X509_PKCS7_has_recipients (PKI_X509_PKCS7 * p7)

Returns PKI_OK if the p7 has recipients already set, PKI_ERR otherwise.

1.41.1.36 int PKI_X509_PKCS7_has_signers (PKI_X509_PKCS7 * p7)

Returns PKI_OK if the p7 has signers already set, PKI_ERR otherwise.

1.41.1.37 PKI_X509_PKCS7* PKI_X509_PKCS7_new (PKI_X509_PKCS7_TYPE type)**1.41.1.38 int PKI_X509_PKCS7_set_cipher (PKI_X509_PKCS7 * p7, PKI_CIPHER * cipher)**

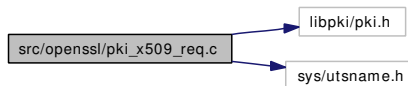
Set the cipher in a encrypted (or signed and encrypted) PKCS7.

1.41.1.39 int PKI_X509_PKCS7_set_recipients (PKI_X509_PKCS7 * p7, PKI_X509_CERT_STACK * x_sk)

Sets the recipients for a PKI_X509_PKCS7.

1.41.1.40 int PKI_X509_PKCS7_VALUE_print_bio (PKI_IO * bio, PKI_X509_PKCS7_VALUE * p7val)**1.42 src/openssl/pki_x509_req.c File Reference**

Include dependency graph for pki_x509_req.c:

**Functions**

- int [PKI_X509_REQ_add_attribute](#) (PKI_X509_REQ *req, PKI_X509_ATTRIBUTE *attr)
- int [PKI_X509_REQ_add_extension](#) (PKI_X509_REQ *x, PKI_X509_EXTENSION *ext)
Adds an extension to a Certificate Request.
- int [PKI_X509_REQ_add_extension_stack](#) (PKI_X509_REQ *x, PKI_X509_EXTENSION_STACK *ext)
Adds a stack of extensions to a Certificate Request.
- int [PKI_X509_REQ_clear_attributes](#) (PKI_X509_REQ *req)
Clears the stack of attributes from a PKI_X509_REQ.
- int [PKI_X509_REQ_delete_attribute](#) (PKI_X509_REQ *req, PKI_ID id)
Deletes an attribute by using its PKI_ID.
- int [PKI_X509_REQ_delete_attribute_by_name](#) (PKI_X509_REQ *req, char *name)
Deletes an attribute by using its name.

- int [PKI_X509_REQ_delete_attribute_by_num](#) (PKI_X509_REQ *req, int pos)
Deletes an attribute by using its number (position).
- void [PKI_X509_REQ_free](#) (PKI_X509_REQ *x)
Frees the memory associated with a Certificate Request object.
- void [PKI_X509_REQ_free_void](#) (void *x)
- PKI_X509_ATTRIBUTE * [PKI_X509_REQ_get_attribute](#) (PKI_X509_REQ *req, PKI_ID type)
Returns an attribute from a PKI_X509_REQ by its type (PKI_ID).
- PKI_X509_ATTRIBUTE * [PKI_X509_REQ_get_attribute_by_name](#) (PKI_X509_REQ *req, char *name)
Returns an attribute from a PKI_X509_REQ by its name.
- PKI_X509_ATTRIBUTE * [PKI_X509_REQ_get_attribute_by_num](#) (PKI_X509_REQ *req, int num)
Returns an attribute from a PKI_X509_REQ by its number (position).
- int [PKI_X509_REQ_get_attributes_num](#) (PKI_X509_REQ *req)
Gets the number of attributes present in a PKI_X509_REQ.
- void * [PKI_X509_REQ_get_data](#) (PKI_X509_REQ *req, PKI_X509_DATA type)
Returns an attribute of the Certificate Request.
- int [PKI_X509_REQ_get_extension_by_name](#) (PKI_X509_REQ *req, char *name)
- int [PKI_X509_REQ_get_extension_by_num](#) (PKI_X509_REQ *req, int num)
- int [PKI_X509_REQ_get_extension_by_oid](#) (PKI_X509_REQ *req, PKI_OID *oid)
- int [PKI_X509_REQ_get_keysize](#) (PKI_X509_REQ *x)
Returns the size of the public key in the Certificate Request.
- const char * [PKI_X509_REQ_get_parsed](#) (PKI_X509_REQ *req, PKI_X509_DATA type)
*Returns a char * representation of the data present in the request.*
- PKI_X509_REQ * [PKI_X509_REQ_new](#) (PKI_X509_KEYPAIR *k, char *subj_s, PKI_X509_PROFILE *req_cnf, PKI_CONFIG *oids, PKI_DIGEST_ALG *digest, HSM *hsm)
Generates a signed Certificate Request Object.
- PKI_X509_REQ * [PKI_X509_REQ_new_null](#) (void)
- int [PKI_X509_REQ_print_parsed](#) (PKI_X509_REQ *req, PKI_X509_DATA type, int fd)
Prints the requested data from the request to the file descriptor passed as an argument.

1.42.1 Function Documentation

1.42.1.1 int [PKI_X509_REQ_add_attribute](#) (PKI_X509_REQ * req, PKI_X509_ATTRIBUTE * attr)

Adds an Attribute to a PKI_X509_REQ

1.42.1.2 int PKI_X509_REQ_add_extension (PKI_X509_REQ * *x*, PKI_X509_EXTENSION * *ext*)

Adds an extension to a Certificate Request.

1.42.1.3 int PKI_X509_REQ_add_extension_stack (PKI_X509_REQ * *x*, PKI_X509_EXTENSION_STACK * *ext*)

Adds a stack of extensions to a Certificate Request.

1.42.1.4 int PKI_X509_REQ_clear_attributes (PKI_X509_REQ * *req*)

Clears the stack of attributes from a PKI_X509_REQ.

1.42.1.5 int PKI_X509_REQ_delete_attribute (PKI_X509_REQ * *req*, PKI_ID *id*)

Deletes an attribute by using its PKI_ID.

1.42.1.6 int PKI_X509_REQ_delete_attribute_by_name (PKI_X509_REQ * *req*, char * *name*)

Deletes an attribute by using its name.

1.42.1.7 int PKI_X509_REQ_delete_attribute_by_num (PKI_X509_REQ * *req*, int *pos*)

Deletes an attribute by using its number (position).

1.42.1.8 void PKI_X509_REQ_free (PKI_X509_REQ * *x*)

Frees the memory associated with a Certificate Request object.

1.42.1.9 void PKI_X509_REQ_free_void (void * *x*)**1.42.1.10 PKI_X509_ATTRIBUTE* PKI_X509_REQ_get_attribute (PKI_X509_REQ * *req*, PKI_ID *type*)**

Returns an attribute from a PKI_X509_REQ by its type (PKI_ID).

1.42.1.11 PKI_X509_ATTRIBUTE* PKI_X509_REQ_get_attribute_by_name (PKI_X509_REQ * *req*, char * *name*)

Returns an attribute from a PKI_X509_REQ by its name.

1.42.1.12 PKI_X509_ATTRIBUTE* PKI_X509_REQ_get_attribute_by_num (PKI_X509_REQ * *req*, int *num*)

Returns an attribute from a PKI_X509_REQ by its number (position).

1.42.1.13 int PKI_X509_REQ_get_attributes_num (PKI_X509_REQ * *req*)

Gets the number of attributes present in a PKI_X509_REQ.

1.42.1.14 void* **PKI_X509_REQ_get_data** (PKI_X509_REQ * *req*, PKI_X509_DATA *type*)

Returns an attribute of the Certificate Request.

1.42.1.15 int **PKI_X509_REQ_get_extension_by_name** (PKI_X509_REQ * *req*, char * *name*)

1.42.1.16 int **PKI_X509_REQ_get_extension_by_num** (PKI_X509_REQ * *req*, int *num*)

1.42.1.17 int **PKI_X509_REQ_get_extension_by_oid** (PKI_X509_REQ * *req*, PKI_OID * *oid*)

1.42.1.18 int **PKI_X509_REQ_get_keysize** (PKI_X509_REQ * *x*)

Returns the size of the public key in the Certificate Request.

1.42.1.19 const char* **PKI_X509_REQ_get_parsed** (PKI_X509_REQ * *req*, PKI_X509_DATA *type*)

Returns a char * representation of the data present in the request.

1.42.1.20 PKI_X509_REQ* **PKI_X509_REQ_new** (PKI_X509_KEYPAIR * *k*, char * *subj_s*, PKI_X509_PROFILE * *req_cnf*, PKI_CONFIG * *oids*, PKI_DIGEST_ALG * *digest*, HSM * *hsm*)

Generates a signed Certificate Request Object.

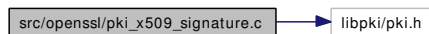
1.42.1.21 PKI_X509_REQ* **PKI_X509_REQ_new_null** (void)

1.42.1.22 int **PKI_X509_REQ_print_parsed** (PKI_X509_REQ * *req*, PKI_X509_DATA *type*, int *fd*)

Prints the requested data from the request to the file descriptor passed as an argument.

1.43 src/openssl/pki_x509_signature.c File Reference

Include dependency graph for pki_x509_signature.c:



Functions

- char * **PKI_X509_SIGNATURE_get_parsed** (PKI_X509_SIGNATURE * *sig*)

Returns a char * with a string representation of the Signature.

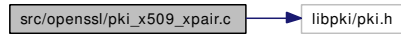
1.43.1 Function Documentation

1.43.1.1 char* **PKI_X509_SIGNATURE_get_parsed** (PKI_X509_SIGNATURE * *sig*)

Returns a char * with a string representation of the Signature.

1.44 src/openssl/pki_x509_xpair.c File Reference

Include dependency graph for pki_x509_xpair.c:



Functions

- void [PKI_X509_XPAIR_free](#) (PKI_X509_XPAIR *x)
- void [PKI_X509_XPAIR_free_void](#) (void *x)
- PKI_X509_CERT * [PKI_X509_XPAIR_get_forward](#) (PKI_X509_XPAIR *xp)
Returns the forward cert pointer present in a cross cert pair.
- PKI_X509_CERT * [PKI_X509_XPAIR_get_reverse](#) (PKI_X509_XPAIR *xp)
Returns the reverse cert pointer present in a cross cert pair.
- PKI_X509_XPAIR * [PKI_X509_XPAIR_new_certs](#) (PKI_X509_CERT *forward, PKI_X509_CERT *reverse)
Creates a X-Certificate data structure and set the appropriate values for the forward and reverse certificate.
- PKI_X509_XPAIR * [PKI_X509_XPAIR_new_null](#) (void)
- int [PKI_X509_XPAIR_set_forward](#) (PKI_X509_XPAIR *xp, PKI_X509_CERT *cert)
Sets the forward certificate in a Cross Cert data structure.
- int [PKI_X509_XPAIR_set_reverse](#) (PKI_X509_XPAIR *xp, PKI_X509_CERT *cert)
Sets the reverse certificate in a Cross Cert data structure.
- PKI_XPAIR * [PKI_XPAIR_new_null](#) (void)
Create an empty cross certificate data structure.

1.44.1 Function Documentation

1.44.1.1 void PKI_X509_XPAIR_free (PKI_X509_XPAIR * x)

1.44.1.2 void PKI_X509_XPAIR_free_void (void * x)

1.44.1.3 PKI_X509_CERT* PKI_X509_XPAIR_get_forward (PKI_X509_XPAIR * xp)

Returns the forward cert pointer present in a cross cert pair.

1.44.1.4 PKI_X509_CERT* PKI_X509_XPAIR_get_reverse (PKI_X509_XPAIR * xp)

Returns the reverse cert pointer present in a cross cert pair.

1.44.1.5 PKI_X509_XPAIR* PKI_X509_XPAIR_new_certs (PKI_X509_CERT * forward, PKI_X509_CERT * reverse)

Creates a X-Certificate data structure and set the appropriate values for the forward and reverse certificate.

1.44.1.6 PKI_X509_XPAIR* PKI_X509_XPAIR_new_null (void)**1.44.1.7 int PKI_X509_XPAIR_set_forward (PKI_X509_XPAIR * *xp*, PKI_X509_CERT * *cert*)**

Sets the forward certificate in a Cross Cert data structure.

1.44.1.8 int PKI_X509_XPAIR_set_reverse (PKI_X509_XPAIR * *xp*, PKI_X509_CERT * *cert*)

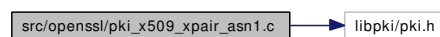
Sets the reverse certificate in a Cross Cert data structure.

1.44.1.9 PKI_XPAIR* PKI_XPAIR_new_null (void)

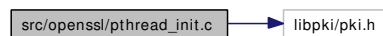
Create an empty cross certificate data structure.

1.45 src/openssl/pki_x509_xpair_asn1.c File Reference

Include dependency graph for pki_x509_xpair_asn1.c:

**1.46 src/openssl/pthread_init.c File Reference**

Include dependency graph for pthread_init.c:

**Defines**

- #define [CRYPTO_LOCK](#) 0x01
- #define [CRYPTO_READ](#) 0x04
- #define [CRYPTO_UNLOCK](#) 0x02
- #define [CRYPTO_WRITE](#) 0x08

Functions

- CRYPTO_dynlock_value * [_dyn_create_callback](#) (const char *file, int line)
- void [_dyn_destroy_callback](#) (struct CRYPTO_dynlock_value *l, const char *file, int line)
- void [_dyn_lock_callback](#) (int mode, struct CRYPTO_dynlock_value *l, const char *file, int line)
- void [thread_cleanup](#) (void)
- void [win32_locking_callback](#) (int mode, int type, char *file, int line)

Variables

- CRYPTO_dynlock_value * [lock_cs](#)

1.46.1 Define Documentation

1.46.1.1 `#define CRYPTO_LOCK 0x01`

1.46.1.2 `#define CRYPTO_READ 0x04`

1.46.1.3 `#define CRYPTO_UNLOCK 0x02`

1.46.1.4 `#define CRYPTO_WRITE 0x08`

1.46.2 Function Documentation

1.46.2.1 `struct CRYPTO_dynlock_value* _dyn_create_callback (const char *file, int line)`

1.46.2.2 `void _dyn_destroy_callback (struct CRYPTO_dynlock_value *l, const char *file, int line)`

1.46.2.3 `void _dyn_lock_callback (int mode, struct CRYPTO_dynlock_value *l, const char *file, int line)`

1.46.2.4 `void thread_cleanup (void)`

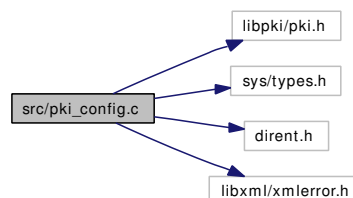
1.46.2.5 `void win32_locking_callback (int mode, int type, char *file, int line)`

1.46.3 Variable Documentation

1.46.3.1 `struct CRYPTO_dynlock_value* lock_cs`

1.47 src/pki_config.c File Reference

Include dependency graph for pki_config.c:



Defines

- `#define LIBXML_MIN_VERSION 20600`
- `#define PKI_DEF_CONF_DIRS_SIZE 2`
- `#define xmlErrorPtr void *`

Functions

- void [logXmlMessages](#) (void *userData, xmlErrorPtr error)
- PKI_CONFIG_ELEMENT * [PKI_CONFIG_add_node](#) (PKI_CONFIG *doc, char *parent, char *name, char *value)
- int [PKI_CONFIG_ELEMENT_add_attribute](#) (PKI_CONFIG *doc, PKI_CONFIG_ELEMENT *node, char *name, char *value)
Adds an attribute to an existing profile element.
- PKI_CONFIG_ELEMENT * [PKI_CONFIG_ELEMENT_add_child](#) (PKI_CONFIG *doc, PKI_CONFIG_ELEMENT *node, char *name, char *value)
Add a child element to an existing node.
- PKI_CONFIG_ELEMENT * [PKI_CONFIG_ELEMENT_add_child_el](#) (PKI_CONFIG *doc, PKI_CONFIG_ELEMENT *node, PKI_CONFIG_ELEMENT *el)
Add a child element to an existing node.
- PKI_CONFIG_ELEMENT * [PKI_CONFIG_ELEMENT_new](#) (char *name, char *value)
Create a new Node for a PKI_X509_PROFILE.
- char * [PKI_CONFIG_find](#) (char *dir, char *name)
Returns a pointer to the filename of the configuration file that contains the configuration named 'name'.
- char * [PKI_CONFIG_find_all](#) (char *dir, char *name, char *subdir)
Returns a pointer to the filename of the configuration file that contains the configuration named 'name'.
- int [PKI_CONFIG_free](#) (PKI_CONFIG *doc)
Frees the memory associated with a PKI_CONFIG object.
- void [PKI_CONFIG_free_void](#) (void *doc)
- char * [PKI_CONFIG_get_attribute_value](#) (PKI_CONFIG *doc, char *search, char *attr_name)
Returns the value of the named attribute in the searched item.
- PKI_CONFIG_ELEMENT * [PKI_CONFIG_get_element](#) (PKI_CONFIG *doc, char *search, int num)
Returns the n-th PKI_CONFIG_ELEMENT identified by the search path.
- PKI_CONFIG_ELEMENT * [PKI_CONFIG_get_element_child](#) (PKI_CONFIG_ELEMENT *e)
Returns the child of a PKI_CONFIG_ELEMENT.
- PKI_CONFIG_ELEMENT_STACK * [PKI_CONFIG_get_element_children](#) (PKI_CONFIG_ELEMENT *e)
Returns the stack of a PKI_CONFIG_ELEMENT's children.
- char * [PKI_CONFIG_get_element_name](#) (PKI_CONFIG_ELEMENT *e)
Returns the name of a PKI_CONFIG_ELEMENT.
- PKI_CONFIG_ELEMENT * [PKI_CONFIG_get_element_next](#) (PKI_CONFIG_ELEMENT *e)
Returns the next PKI_CONFIG_ELEMENT.
- PKI_CONFIG_ELEMENT * [PKI_CONFIG_get_element_prev](#) (PKI_CONFIG_ELEMENT *e)

Returns the previous `PKI_CONFIG_ELEMENT`.

- `PKI_CONFIG_ELEMENT_STACK * PKI_CONFIG_get_element_stack (PKI_CONFIG *doc, char *search)`

Returns the stack of elements identified by the search path.

- `char * PKI_CONFIG_get_element_value (PKI_CONFIG_ELEMENT *e)`

Returns the value of a `PKI_CONFIG_ELEMENT`.

- `int PKI_CONFIG_get_elements_num (PKI_CONFIG *doc, char *search)`

Returns the number of items identified by the search path.

- `PKI_CONFIG_ELEMENT * PKI_CONFIG_get_root (PKI_CONFIG *doc)`

Gets the root element of a `PKI_CONFIG` document.

- `PKI_STACK * PKI_CONFIG_get_search_paths (char *dir)`

Returns a `PKI_STACK` of directories (useful to search in default dirs for config files).

- `PKI_STACK * PKI_CONFIG_get_stack_value (PKI_CONFIG *doc, char *search)`

Returns a stack of values for the selected search path.

- `char * PKI_CONFIG_get_value (PKI_CONFIG *doc, char *search)`

Returns the first value found via the provided search path.

- `PKI_CONFIG * PKI_CONFIG_load (char *urlPath)`

Loads a `PKI_CONFIG` object (XML config file).

- `PKI_CONFIG_STACK * PKI_CONFIG_load_all (char *dir)`

Returns a stack of all configurations files found in the passed directory plush the default search paths.

- `PKI_CONFIG_STACK * PKI_CONFIG_load_dir (char *dir, PKI_CONFIG_STACK *sk)`

Returns a stack of all configuration files found in the passed directory.

- `PKI_OID * PKI_CONFIG_OID_search (PKI_CONFIG *doc, char *searchName)`

Searches for a specific `OID` inside a `PKI_CONFIG` object.

1.47.1 Define Documentation

1.47.1.1 `#define LIBXML_MIN_VERSION 20600`

1.47.1.2 `#define PKI_DEF_CONF_DIRS_SIZE 2`

1.47.1.3 `#define xmlErrorPtr void *`

1.47.2 Function Documentation

1.47.2.1 `void logXmlMessages (void * userData, xmlErrorPtr error)`

1.47.2.2 `PKI_CONFIG_ELEMENT* PKI_CONFIG_add_node (PKI_CONFIG * doc, char * parent, char * name, char * value)`

1.47.2.3 `int PKI_CONFIG_ELEMENT_add_attribute (PKI_CONFIG * doc, PKI_CONFIG_ELEMENT * node, char * name, char * value)`

Adds an attribute to an existing profile element.

1.47.2.4 `PKI_CONFIG_ELEMENT* PKI_CONFIG_ELEMENT_add_child (PKI_CONFIG * doc, PKI_CONFIG_ELEMENT * node, char * name, char * value)`

Add a child element to an existing node.

1.47.2.5 `PKI_CONFIG_ELEMENT* PKI_CONFIG_ELEMENT_add_child_el (PKI_CONFIG * doc, PKI_CONFIG_ELEMENT * node, PKI_CONFIG_ELEMENT * el)`

Add a child element to an existing node.

1.47.2.6 `PKI_CONFIG_ELEMENT* PKI_CONFIG_ELEMENT_new (char * name, char * value)`

Create a new Node for a PKI_X509_PROFILE.

1.47.2.7 `char* PKI_CONFIG_find (char * dir, char * name)`

Returns a pointer to the filename of the configuration file that contains the configuration named 'name'.

1.47.2.8 `char* PKI_CONFIG_find_all (char * dir, char * name, char * subdir)`

Returns a pointer to the filename of the configuration file that contains the configuration named 'name'.

1.47.2.9 `int PKI_CONFIG_free (PKI_CONFIG * doc)`

Frees the memory associated with a PKI_CONFIG object.

1.47.2.10 `void PKI_CONFIG_free_void (void * doc)`

1.47.2.11 `char* PKI_CONFIG_get_attribute_value (PKI_CONFIG * doc, char * search, char * attr_name)`

Returns the value of the named attribute in the searched item.

1.47.2.12 `PKI_CONFIG_ELEMENT* PKI_CONFIG_get_element (PKI_CONFIG * doc, char * search, int num)`

Returns the n-th PKI_CONFIG_ELEMENT identified by the search path.

1.47.2.13 `PKI_CONFIG_ELEMENT* PKI_CONFIG_get_element_child (PKI_CONFIG_ELEMENT * e)`

Returns the child of a PKI_CONFIG_ELEMENT.

1.47.2.14 **PKI_CONFIG_ELEMENT_STACK*** **PKI_CONFIG_get_element_children** (**PKI_CONFIG_ELEMENT * *e***)

Returns the stack of a PKI_CONFIG_ELEMENT's children.

1.47.2.15 **char*** **PKI_CONFIG_get_element_name** (**PKI_CONFIG_ELEMENT * *e***)

Returns the name of a PKI_CONFIG_ELEMENT.

1.47.2.16 **PKI_CONFIG_ELEMENT*** **PKI_CONFIG_get_element_next** (**PKI_CONFIG_ELEMENT * *e***)

Returns the next PKI_CONFIG_ELEMENT.

1.47.2.17 **PKI_CONFIG_ELEMENT*** **PKI_CONFIG_get_element_prev** (**PKI_CONFIG_ELEMENT * *e***)

Returns the previous PKI_CONFIG_ELEMENT.

1.47.2.18 **PKI_CONFIG_ELEMENT_STACK*** **PKI_CONFIG_get_element_stack** (**PKI_CONFIG * *doc*, char * *search***)

Returns the stack of elements identified by the search path.

1.47.2.19 **char*** **PKI_CONFIG_get_element_value** (**PKI_CONFIG_ELEMENT * *e***)

Returns the value of a PKI_CONFIG_ELEMENT.

1.47.2.20 **int** **PKI_CONFIG_get_elements_num** (**PKI_CONFIG * *doc*, char * *search***)

Returns the number of items identified by the search path.

1.47.2.21 **PKI_CONFIG_ELEMENT*** **PKI_CONFIG_get_root** (**PKI_CONFIG * *doc***)

Gets the root element of a PKI_CONFIG document.

1.47.2.22 **PKI_STACK*** **PKI_CONFIG_get_search_paths** (**char * *dir***)

Returns a PKI_STACK of directories (useful to search in default dirs for config files).

1.47.2.23 **PKI_STACK*** **PKI_CONFIG_get_stack_value** (**PKI_CONFIG * *doc*, char * *search***)

Returns a stack of values for the selected search path.

1.47.2.24 **char*** **PKI_CONFIG_get_value** (**PKI_CONFIG * *doc*, char * *search***)

Returns the first value found via the provided search path.

1.47.2.25 **PKI_CONFIG*** **PKI_CONFIG_load** (**char * *urlPath***)

Loads a PKI_CONFIG object (XML config file).

1.47.2.26 PKI_CONFIG_STACK* PKI_CONFIG_load_all (char * dir)

Returns a stack of all configurations files found in the passed directory plus the default search paths.

1.47.2.27 PKI_CONFIG_STACK* PKI_CONFIG_load_dir (char * dir, PKI_CONFIG_STACK * sk)

Returns a stack of all configuration files found in the passed directory.

1.47.2.28 PKI_OID* PKI_CONFIG_OID_search (PKI_CONFIG * doc, char * searchName)

Searches for a specific OID inside a PKI_CONFIG object.

1.48 src/pki_cred.c File Reference

Include dependency graph for pki_cred.c:

**Functions**

- PKI_CRED * [PKI_CRED_dup](#) (PKI_CRED *cred)
Duplicates a PKI_CRED data structure.
- void [PKI_CRED_free](#) (PKI_CRED *cred)
Free a PKI_CRED memory region.
- pki_ssl_t * [PKI_CRED_get_ssl](#) (PKI_CRED *cred)
Gets the pointer to the PKI_SSL structure inside a CRED.
- PKI_CRED * [PKI_CRED_new](#) (char *user, char *pwd)
Allocates a new PKI_CRED structure.
- PKI_CRED * [PKI_CRED_new_null](#) (void)
Allocates a new PKI_CRED structure.
- int [PKI_CRED_set_ssl](#) (PKI_CRED *cred, struct pki_ssl_t *ssl)
Sets the SSL configuration for Creds.

1.48.1 Function Documentation**1.48.1.1 PKI_CRED* PKI_CRED_dup (PKI_CRED * cred)**

Duplicates a PKI_CRED data structure.

1.48.1.2 void PKI_CRED_free (PKI_CRED * cred)

Free a PKI_CRED memory region.

This function frees a PKI_CRED data structure. The internal data is also freed (no need to free the internal structures before calling this function).

No value is returned (void).

1.48.1.3 struct pki_ssl_t* PKI_CRED_get_ssl (PKI_CRED * cred)

Gets the pointer to the PKI_SSL structure inside a CRED.

1.48.1.4 PKI_CRED* PKI_CRED_new (char * user, char * pwd)

Allocates a new PKI_CRED structure.

Allocates memory for a new PKI_CRED structure. The returned data structure contains a copy (strdup) of the passed user and pwd strings.

The function returns a pointer to a PKI_CRED structure in case of success, otherwise it returns NULL.

1.48.1.5 PKI_CRED* PKI_CRED_new_null (void)

Allocates a new PKI_CRED structure.

Allocates memory for a new PKI_CRED structure. The returned data structure is already zeroized.

The function returns a pointer to a PKI_CRED structure in case of success, otherwise it returns NULL.

1.48.1.6 int PKI_CRED_set_ssl (PKI_CRED * cred, struct pki_ssl_t * ssl)

Sets the SSL configuration for Creds.

1.49 src/pki_err.c File Reference

Include dependency graph for pki_err.c:

**Defines**

- `#define __LIBPKI_ERR__`

Functions

- `int PKI_get_err (void)`
Get the current error number.
- `const char * PKI_get_strerror (void)`
Returns the string pointer to the error value.
- `int PKI_set_err (int err)`

Set PKI err (internally used by library functions).

Variables

- PKI_STACK * [pki_err_stack](#) = NULL

1.49.1 Define Documentation

1.49.1.1 #define __LIBPKI_ERR__

1.49.2 Function Documentation

1.49.2.1 int PKI_get_err (void)

Get the current error number.

1.49.2.2 const char* PKI_get_strerror (void)

Returns the string pointer to the error value.

1.49.2.3 int PKI_set_err (int *err*)

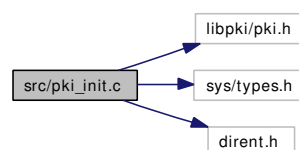
Set PKI err (internally used by library functions).

1.49.3 Variable Documentation

1.49.3.1 PKI_STACK* [pki_err_stack](#) = NULL

1.50 src/pki_init.c File Reference

Include dependency graph for pki_init.c:



Functions

- PKI_TOKEN_STACK * [PKI_get_all_tokens](#) (char *dir)
- PKI_TOKEN_STACK * [PKI_get_all_tokens_dir](#) (char *dir, PKI_TOKEN_STACK *list)
- int [PKI_get_init_status](#) (void)

Returns the status of the library (check for initialization).

- int [PKI_init_all](#) (void)

Initialize libpki internal structures.

- `PKI_ID_INFO_STACK * PKI_list_all_id` (void)
Returns a stack of all the available Identities.
- `PKI_STACK * PKI_list_all_tokens` (char *dir)
- `PKI_STACK * PKI_list_all_tokens_dir` (char *dir, PKI_STACK *list)

Variables

- const long `LIBPKI_OS_DETAILS`
- int `NID_proxyCertInfo` = -1

1.50.1 Function Documentation

1.50.1.1 `PKI_TOKEN_STACK* PKI_get_all_tokens` (char * *dir*)

1.50.1.2 `PKI_TOKEN_STACK* PKI_get_all_tokens_dir` (char * *dir*, PKI_TOKEN_STACK * *list*)

1.50.1.3 `int PKI_get_init_status` (void)

Returns the status of the library (check for initialization).

1.50.1.4 `int PKI_init_all` (void)

Initialize libpki internal structures.

1.50.1.5 `PKI_ID_INFO_STACK* PKI_list_all_id` (void)

Returns a stack of all the available Identities.

1.50.1.6 `PKI_STACK* PKI_list_all_tokens` (char * *dir*)

1.50.1.7 `PKI_STACK* PKI_list_all_tokens_dir` (char * *dir*, PKI_STACK * *list*)

1.50.2 Variable Documentation

1.50.2.1 const long `LIBPKI_OS_DETAILS`

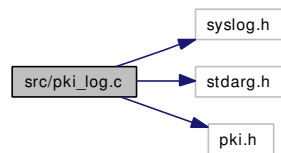
Initial value:

```
LIBPKI_OS_CLASS |
    LIBPKI_OS_BITS | LIBPKI_OS_VENDOR
```

1.50.2.2 `int NID_proxyCertInfo` = -1

1.51 src/pki_log.c File Reference

Include dependency graph for pki_log.c:



Functions

- void [PKI_log](#) (int level, const char *fmt,...)
Add an entry in the log.
- void [PKI_log_debug_simple](#) (const char *fmt,...)
Add an entry in the Debug log.
- int [PKI_log_end](#) (void)
Finalize the log subsystem.
- void [PKI_log_err_simple](#) (const char *fmt,...)
Add an entry in the Debug log.
- int [PKI_log_init](#) (PKI_LOG_TYPE type, PKI_LOG_LEVEL level, char *resource, PKI_LOG_FLAGS flags, PKI_TOKEN *tk)
Initialize the log subsystem.

1.51.1 Function Documentation

1.51.1.1 void PKI_log (int level, const char *fmt, ...)

Add an entry in the log.

1.51.1.2 void PKI_log_debug_simple (const char *fmt, ...)

Add an entry in the Debug log.

1.51.1.3 int PKI_log_end (void)

Finalize the log subsystem.

1.51.1.4 void PKI_log_err_simple (const char *fmt, ...)

Add an entry in the Debug log.

1.51.1.5 int PKI_log_init (PKI_LOG_TYPE type, PKI_LOG_LEVEL level, char *resource, PKI_LOG_FLAGS flags, PKI_TOKEN *tk)

Initialize the log subsystem.

1.52 src/pki_mem.c File Reference

Include dependency graph for pki_mem.c:



Functions

- void [PKI_Free](#) (void *ret)
Frees memory associated with a pointer (allocated with PKI_Malloc).
- void * [PKI_Malloc](#) (size_t size)
Allocates size bytes of memory, zeroize it, and returns the pointer to the beginning of the memory region.
- int [PKI_MEM_add](#) (PKI_MEM *buf, char *data, size_t data_size)
Adds the passed data to a PKI_MEM.
- int [PKI_MEM_B64_decode](#) (PKI_MEM *b64_mem)
Decodes a PKI_MEM from B64.
- int [PKI_MEM_B64_encode](#) (PKI_MEM *der)
Encodes the contents of a PKI_MEM in B64 format.
- PKI_MEM * [PKI_MEM_dup](#) (PKI_MEM *mem)
Duplicates a PKI_MEM.
- ssize_t [PKI_MEM_fprintf](#) (FILE *file, PKI_MEM *buf)
Prints the content of a PKI_MEM to a FILE pointer.
- int [PKI_MEM_free](#) (PKI_MEM *buf)
Frees the memory associated with a PKI_MEM, returns 1.
- void [PKI_MEM_free_void](#) (void *buf)
- unsigned char * [PKI_MEM_get_data](#) (PKI_MEM *buf)
Returns the pointer to the data within a PKI_MEM datastructure.
- size_t [PKI_MEM_get_size](#) (PKI_MEM *buf)
Returns the size of the data within a PKI_MEM datastructure.
- int [PKI_MEM_grow](#) (PKI_MEM *buf, size_t data_size)
Grows the allocated size of data_size bytes.
- PKI_MEM * [PKI_MEM_new](#) (size_t size)
Returns a new PKI_MEM object with size allocated data.
- PKI_MEM * [PKI_MEM_new_bio](#) (PKI_IO *io, PKI_MEM **mem)
Returns a PKI_MEM with the contents read from the PKI_IO.
- PKI_MEM * [PKI_MEM_new_data](#) (size_t size, unsigned char *data)

Returns a new PKI_MEM object with a copy of the data passed as arg.

- PKI_MEM * [PKI_MEM_new_func](#) (void *obj, int(*func)(void *, unsigned char **p))

Returns a PKI_MEM with the contents decoded via a function.

- PKI_MEM * [PKI_MEM_new_func_bio](#) (void *obj, int(*func)(BIO *, void *))

Returns the content from a BIO after dec it with func(BIO,void*).*

- PKI_MEM * [PKI_MEM_new_membio](#) (PKI_IO *io)

Returns a PKI_MEM with the contents of a memory PKI_IO.

- PKI_MEM * [PKI_MEM_new_null](#) (void)

Returns a new PKI_MEM object with no data associated with it.

- ssize_t [PKI_MEM_printf](#) (PKI_MEM *buf)

Prints the content of a PKI_MEM to stdout.

- PKI_MEM * [PKI_MEM_url_decode](#) (PKI_MEM *mem, int skip_newlines)

Decodes a URL-encoded version of a PKI_MEM.

- PKI_MEM * [PKI_MEM_url_encode](#) (PKI_MEM *mem, int skip_newlines)

Returns a URL-safe encoded version of a PKI_MEM.

- void [PKI_ZFree](#) (void *pnt, size_t size)

Frees and Zeroizes memory associated with a pointer.

- void [PKI_ZFree_str](#) (char *str)

Frees and Zeroizes memory associated with a string.

1.52.1 Function Documentation

1.52.1.1 void PKI_Free (void *ret)

Frees memory associated with a pointer (allocated with PKI_Malloc).

1.52.1.2 void* PKI_Malloc (size_t size)

Allocates size bytes of memory, zeroize it, and returns the pointer to the beginning of the memory region.

1.52.1.3 int PKI_MEM_add (PKI_MEM *buf, char *data, size_t data_size)

Adds the passed data to a PKI_MEM.

1.52.1.4 int PKI_MEM_B64_decode (PKI_MEM *b64_mem)

Decodes a PKI_MEM from B64.

1.52.1.5 int PKI_MEM_B64_encode (PKI_MEM *der)

Encodes the contents of a PKI_MEM in B64 format.

1.52.1.6 PKI_MEM* PKI_MEM_dup (PKI_MEM * *mem*)

Duplicates a PKI_MEM.

1.52.1.7 ssize_t PKI_MEM_fprintf (FILE * *file*, PKI_MEM * *buf*)

Prints the content of a PKI_MEM to a FILE pointer.

1.52.1.8 int PKI_MEM_free (PKI_MEM * *buf*)

Frees the memory associated with a PKI_MEM, returns 1.

1.52.1.9 void PKI_MEM_free_void (void * *buf*)**1.52.1.10 unsigned char* PKI_MEM_get_data (PKI_MEM * *buf*)**

Returns the pointer to the data within a PKI_MEM datastructure.

1.52.1.11 size_t PKI_MEM_get_size (PKI_MEM * *buf*)

Returns the size of the data within a PKI_MEM datastructure.

1.52.1.12 int PKI_MEM_grow (PKI_MEM * *buf*, size_t *data_size*)

Grows the allocated size of *data_size* bytes.

1.52.1.13 PKI_MEM* PKI_MEM_new (size_t *size*)

Returns a new PKI_MEM object with *size* allocated data.

1.52.1.14 PKI_MEM* PKI_MEM_new_bio (PKI_IO * *io*, PKI_MEM ** *mem*)

Returns a PKI_MEM with the contents read from the PKI_IO.

1.52.1.15 PKI_MEM* PKI_MEM_new_data (size_t *size*, unsigned char * *data*)

Returns a new PKI_MEM object with a copy of the data passed as arg.

1.52.1.16 PKI_MEM* PKI_MEM_new_func (void * *obj*, int(*) (void *, unsigned char **p) *func*)

Returns a PKI_MEM with the contents decoded via a function.

Returns a new PKI_MEM object filled with the data from an object and its render function from *func* which accepts BIO * and data * as its input.

1.52.1.17 PKI_MEM* PKI_MEM_new_func_bio (void * *obj*, int(*) (BIO *, void *) *func*)

Returns the content from a BIO after dec it with *func*(BIO*,void*).

1.52.1.18 PKI_MEM* PKI_MEM_new_membio (PKI_IO * *io*)

Returns a PKI_MEM with the contents of a memory PKI_IO.

1.52.1.19 PKI_MEM* PKI_MEM_new_null (void)

Returns a new PKI_MEM object with no data associated with it.

1.52.1.20 ssize_t PKI_MEM_printf (PKI_MEM * buf)

Prints the content of a PKI_MEM to stdout.

1.52.1.21 PKI_MEM* PKI_MEM_url_decode (PKI_MEM * mem, int skip_newlines)

Decodes a URL-encoded version of a PKI_MEM.

1.52.1.22 PKI_MEM* PKI_MEM_url_encode (PKI_MEM * mem, int skip_newlines)

Returns a URL-safe encoded version of a PKI_MEM.

1.52.1.23 void PKI_ZFree (void * pnt, size_t size)

Frees and Zeroizes memory associated with a pointer.

1.52.1.24 void PKI_ZFree_str (char * str)

Frees and Zeroizes memory associated with a string.

1.53 src/pki_msg_req.c File Reference

Include dependency graph for pki_msg_req.c:

**Functions**

- int [PKI_MSG_REQ_add_data](#) (PKI_MSG_REQ *msg, unsigned char *data, size_t size)
Adds data to the Message body.
- int [PKI_MSG_REQ_add_recipient](#) (PKI_MSG_REQ *msg, PKI_X509_CERT *x)
Adds a certificate to the list of recipients.
- int [PKI_MSG_REQ_clear_data](#) (PKI_MSG_REQ *msg)
Clears the data of the Message body.
- int [PKI_MSG_REQ_clear_recipients](#) (PKI_MSG_REQ *msg)
Clears the list of recipients in a PKI_MSG_REQ.
- int [PKI_MSG_REQ_encode](#) (PKI_MSG_REQ *msg, PKI_MSG_PROTO proto)
Encodes the message according to the selected PKI_MSG_PROTO.
- void [PKI_MSG_REQ_free](#) (PKI_MSG_REQ *msg)
Free a PKI_MSG_REQ data structure.

- PKI_MSG_REQ_ACTION [PKI_MSG_REQ_get_action](#) (PKI_MSG_REQ *msg)
Gets the basic action in a PKI_MSG_REQ message.
- PKI_X509_CERT * [PKI_MSG_REQ_get_cacert](#) (PKI_MSG_REQ *msg)
Gets the Certificate of the CA the request is intended for.
- void * [PKI_MSG_REQ_get_encoded](#) (PKI_MSG_REQ *msg)
Gets the encoded version of the message.
- PKI_X509_KEYPAIR * [PKI_MSG_REQ_get_keypair](#) (PKI_MSG_REQ *msg)
Gets the Keypair to be used when generating the request.
- char * [PKI_MSG_REQ_get_loa](#) (PKI_MSG_REQ *msg)
Gets the requested certificate template.
- PKI_MSG_PROTO [PKI_MSG_REQ_get_proto](#) (PKI_MSG_REQ *msg)
Returns the PKI_MSG_PROTO from a PKI_MSG_REQUEST object.
- PKI_X509_CERT_STACK * [PKI_MSG_REQ_get_recipients](#) (PKI_MSG_REQ *msg)
Gets the list of recipients in a PKI_MSG_REQ.
- PKI_X509_CERT * [PKI_MSG_REQ_get_signer](#) (PKI_MSG_REQ *msg)
Gets the Signer Certificate.
- char * [PKI_MSG_REQ_get_subject](#) (PKI_MSG_REQ *msg)
Gets the Subject to be used in the certificate request.
- char * [PKI_MSG_REQ_get_template](#) (PKI_MSG_REQ *msg)
Gets the requested certificate template.
- PKI_MSG_REQ * [PKI_MSG_REQ_new](#) (PKI_MSG_REQ_ACTION action, char *subject, char *template_name, PKI_X509_KEYPAIR *sign_key, PKI_X509_CERT *signer, PKI_X509_CERT *cacert)
Builds a new message.
- PKI_MSG_REQ * [PKI_MSG_REQ_new_null](#) (void)
Returns an empty generic PKI request message.
- PKI_MSG_REQ * [PKI_MSG_REQ_new_tk](#) (PKI_MSG_REQ_ACTION action, char *subject, char *template_name, PKI_TOKEN *tk)
- int [PKI_MSG_REQ_replace_data](#) (PKI_MSG_REQ *msg, unsigned char *data, size_t size)
Replaces data in a PKI_MSG_REQ message.
- int [PKI_MSG_REQ_SCEP_new](#) (PKI_MSG_REQ *msg)
- PKI_MSG_RESP * [PKI_MSG_REQ_SCEP_send](#) (PKI_MSG_REQ *msg, PKI_STACK *sk, PKI_TOKEN *tk)
- PKI_MSG_RESP * [PKI_MSG_REQ_send](#) (PKI_MSG_REQ *msg, PKI_TOKEN *tk, char *url_s)
Sends a message and retrieves the response.

- int [PKI_MSG_REQ_set_action](#) (PKI_MSG_REQ *msg, PKI_MSG_REQ_ACTION action)
Sets the basic action in a PKI_MSG_REQ message.
- int [PKI_MSG_REQ_set_cacert](#) (PKI_MSG_REQ *msg, PKI_X509_CERT *cacert)
Sets the Certificate of the CA the request is intended for.
- int [PKI_MSG_REQ_set_keypair](#) (PKI_MSG_REQ *msg, PKI_X509_KEYPAIR *pkey)
Sets the Keypair to be used when generating the request.
- int [PKI_MSG_REQ_set_loa](#) (PKI_MSG_REQ *msg, char *loa)
Sets the requested certificate level of assurance (LOA).
- int [PKI_MSG_REQ_set_proto](#) (PKI_MSG_REQ *msg, PKI_MSG_PROTO proto)
Sets the messaging protocol to be used.
- int [PKI_MSG_REQ_set_recipients](#) (PKI_MSG_REQ *msg, PKI_X509_CERT_STACK *x_sk)
Sets the list of recipients in a PKI_MSG_REQ.
- int [PKI_MSG_REQ_set_signer](#) (PKI_MSG_REQ *msg, PKI_X509_CERT *signer)
Sets the Signer Certificate.
- int [PKI_MSG_REQ_set_subject](#) (PKI_MSG_REQ *msg, char *subject)
Sets the Subject to be used in the certificate request.
- int [PKI_MSG_REQ_set_template](#) (PKI_MSG_REQ *msg, char *name)
Sets the requested certificate template.

1.53.1 Function Documentation

1.53.1.1 int PKI_MSG_REQ_add_data (PKI_MSG_REQ * msg, unsigned char * data, size_t size)

Adds data to the Message body.

1.53.1.2 int PKI_MSG_REQ_add_recipient (PKI_MSG_REQ * msg, PKI_X509_CERT * x)

Adds a certificate to the list of recipients.

1.53.1.3 int PKI_MSG_REQ_clear_data (PKI_MSG_REQ * msg)

Clears the data of the Message body.

1.53.1.4 int PKI_MSG_REQ_clear_recipients (PKI_MSG_REQ * msg)

Clears the list of recipients in a PKI_MSG_REQ.

1.53.1.5 int PKI_MSG_REQ_encode (PKI_MSG_REQ * msg, PKI_MSG_PROTO proto)

Encodes the message according to the selected PKI_MSG_PROTO.

1.53.1.6 void PKI_MSG_REQ_free (PKI_MSG_REQ * msg)

Free a PKI_MSG_REQ data structure.

1.53.1.7 PKI_MSG_REQ_ACTION PKI_MSG_REQ_get_action (PKI_MSG_REQ * msg)

Gets the basic action in a PKI_MSG_REQ message.

1.53.1.8 PKI_X509_CERT* PKI_MSG_REQ_get_cacert (PKI_MSG_REQ * msg)

Gets the Certificate of the CA the request is intended for.

1.53.1.9 void* PKI_MSG_REQ_get_encoded (PKI_MSG_REQ * msg)

Gets the encoded version of the message.

1.53.1.10 PKI_X509_KEYPAIR* PKI_MSG_REQ_get_keypair (PKI_MSG_REQ * msg)

Gets the Keypair to be used when generating the request.

1.53.1.11 char* PKI_MSG_REQ_get_loa (PKI_MSG_REQ * msg)

Gets the requested certificate template.

1.53.1.12 PKI_MSG_PROTO PKI_MSG_REQ_get_proto (PKI_MSG_REQ * msg)

Returns the PKI_MSG_PROTO from a PKI_MSG_REQUEST object.

1.53.1.13 PKI_X509_CERT_STACK* PKI_MSG_REQ_get_recipients (PKI_MSG_REQ * msg)

Gets the list of recipients in a PKI_MSG_REQ.

1.53.1.14 PKI_X509_CERT* PKI_MSG_REQ_get_signer (PKI_MSG_REQ * msg)

Gets the Signer Certificate.

1.53.1.15 char* PKI_MSG_REQ_get_subject (PKI_MSG_REQ * msg)

Gets the Subject to be used in the certificate request.

1.53.1.16 char* PKI_MSG_REQ_get_template (PKI_MSG_REQ * msg)

Gets the requested certificate template.

1.53.1.17 PKI_MSG_REQ* PKI_MSG_REQ_new (PKI_MSG_REQ_ACTION action, char * subject, char * template_name, PKI_X509_KEYPAIR * sign_key, PKI_X509_CERT * signer, PKI_X509_CERT * cacert)

Builds a new message.

1.53.1.18 PKI_MSG_REQ* PKI_MSG_REQ_new_null (void)

Returns an empty generic PKI request message.

1.53.1.19 `PKI_MSG_REQ* PKI_MSG_REQ_new_tk (PKI_MSG_REQ_ACTION action, char * subject, char * template_name, PKI_TOKEN * tk)`

Builds a new PKI message by using a PKI_TOKEN

1.53.1.20 `int PKI_MSG_REQ_replace_data (PKI_MSG_REQ * msg, unsigned char * data, size_t size)`

Replaces data in a PKI_MSG_REQ message.

1.53.1.21 `int PKI_MSG_REQ_SCEP_new (PKI_MSG_REQ * msg)`

Returns a SCEP_MSG from the passed PKI_MSG_REQ

1.53.1.22 `PKI_MSG_RESP* PKI_MSG_REQ_SCEP_send (PKI_MSG_REQ * msg, PKI_STACK * sk, PKI_TOKEN * tk)`

1.53.1.23 `PKI_MSG_RESP* PKI_MSG_REQ_send (PKI_MSG_REQ * msg, PKI_TOKEN * tk, char * url_s)`

Sends a message and retrieves the response.

1.53.1.24 `int PKI_MSG_REQ_set_action (PKI_MSG_REQ * msg, PKI_MSG_REQ_ACTION action)`

Sets the basic action in a PKI_MSG_REQ message.

1.53.1.25 `int PKI_MSG_REQ_set_cacert (PKI_MSG_REQ * msg, PKI_X509_CERT * cacert)`

Sets the Certificate of the CA the request is intended for.

1.53.1.26 `int PKI_MSG_REQ_set_keypair (PKI_MSG_REQ * msg, PKI_X509_KEYPAIR * pkey)`

Sets the Keypair to be used when generating the request.

1.53.1.27 `int PKI_MSG_REQ_set_loa (PKI_MSG_REQ * msg, char * loa)`

Sets the requested certificate level of assurance (LOA).

1.53.1.28 `int PKI_MSG_REQ_set_proto (PKI_MSG_REQ * msg, PKI_MSG_PROTO proto)`

Sets the messaging protocol to be used.

1.53.1.29 `int PKI_MSG_REQ_set_recipients (PKI_MSG_REQ * msg, PKI_X509_CERT_STACK * x_sk)`

Sets the list of recipients in a PKI_MSG_REQ.

1.53.1.30 `int PKI_MSG_REQ_set_signer (PKI_MSG_REQ * msg, PKI_X509_CERT * signer)`

Sets the Signer Certificate.

1.53.1.31 int PKI_MSG_REQ_set_subject (PKI_MSG_REQ * msg, char * subject)

Sets the Subject to be used in the certificate request.

1.53.1.32 int PKI_MSG_REQ_set_template (PKI_MSG_REQ * msg, char * name)

Sets the requested certificate template.

1.54 src/pki_msg_resp.c File Reference

Include dependency graph for pki_msg_resp.c:

**Functions**

- int [PKI_MSG_RESP_add_data](#) (PKI_MSG_RESP *msg, unsigned char *data, size_t size)
Adds data to the Message body.
- int [PKI_MSG_RESP_add_recipient](#) (PKI_MSG_RESP *msg, PKI_X509_CERT *x)
Adds a certificate to the list of recipients.
- int [PKI_MSG_RESP_clear_data](#) (PKI_MSG_RESP *msg)
Clears the data of the Message body.
- int [PKI_MSG_RESP_clear_recipients](#) (PKI_MSG_RESP *msg)
Clears the list of recipients in a PKI_MSG_RESP.
- void * [PKI_MSG_RESP_encode](#) (PKI_MSG_RESP *msg, PKI_MSG_PROTO proto)
Encodes the message according to the selected PKI_MSG_PROTO.
- void [PKI_MSG_RESP_free](#) (PKI_MSG_RESP *msg)
Free a PKI_MSG_REQ data structure.
- PKI_MSG_RESP_ACTION [PKI_MSG_RESP_get_action](#) (PKI_MSG_RESP *msg)
Gets the basic action in a PKI_MSG_RESP message.
- PKI_X509_CERT * [PKI_MSG_RESP_get_cacert](#) (PKI_MSG_RESP *msg)
Gets the CA certificate from the Response.
- void * [PKI_MSG_RESP_get_encoded](#) (PKI_MSG_RESP *msg)
Gets the encoded version of the Response message.
- PKI_X509_CERT * [PKI_MSG_RESP_get_issued_cert](#) (PKI_MSG_RESP *msg)
Gets the certificate from the Response.
- PKI_X509_KEYPAIR * [PKI_MSG_RESP_get_keypair](#) (PKI_MSG_RESP *msg)
Gets the Keypair to be used when generating the response.

- `PKI_MSG_PROTO` [PKI_MSG_RESP_get_proto](#) (`PKI_MSG_RESP *msg`)
Returns the `PKI_MSG_PROTO` from a `PKI_MSG_RESP` object.
- `PKI_X509_CERT_STACK *` [PKI_MSG_RESP_get_recipients](#) (`PKI_MSG_RESP *msg`)
Gets the list of recipients in a `PKI_MSG_RESP`.
- `PKI_X509_CERT *` [PKI_MSG_RESP_get_signer](#) (`PKI_MSG_RESP *msg`)
Gets the Signer Certificate.
- `PKI_MSG_STATUS` [PKI_MSG_RESP_get_status](#) (`PKI_MSG_RESP *msg`)
Gets the status in a `PKI_MSG_RESP` message.
- `PKI_MSG_RESP *` [PKI_MSG_RESP_new](#) (`PKI_MSG_RESP_ACTION` action, `PKI_MSG_STATUS` status, `PKI_X509_KEYPAIR *`sign_key, `PKI_X509_CERT *`signer, `PKI_X509_CERT *`cacert)
Builds a new message.
- `PKI_MSG_RESP *` [PKI_MSG_RESP_new_null](#) (void)
Returns an empty generic PKI response message.
- `PKI_MSG_RESP *` [PKI_MSG_RESP_new_tk](#) (`PKI_MSG_RESP_ACTION` action, `PKI_MSG_STATUS` status, `PKI_TOKEN *`tk)
- `int` [PKI_MSG_RESP_replace_data](#) (`PKI_MSG_RESP *msg`, unsigned char *data, size_t size)
Replaces data in a `PKI_MSG_RESP` message.
- `PKI_X509_SCEP_MSG *` [PKI_MSG_RESP_SCEP_new](#) (`PKI_MSG_RESP *msg`)
- `int` [PKI_MSG_RESP_set_action](#) (`PKI_MSG_RESP *msg`, `PKI_MSG_RESP_ACTION` action)
Sets the basic action in a `PKI_MSG_RESP` message.
- `int` [PKI_MSG_RESP_set_cacert](#) (`PKI_MSG_RESP *msg`, `PKI_X509_CERT *`x)
Sets the CA certificate in the Response.
- `int` [PKI_MSG_RESP_set_issued_cert](#) (`PKI_MSG_RESP *msg`, `PKI_X509_CERT *`x)
Sets the certificate from the Response.
- `int` [PKI_MSG_RESP_set_keypair](#) (`PKI_MSG_RESP *msg`, `PKI_X509_KEYPAIR *`pkey)
Sets the Keypair to be used when generating the response.
- `int` [PKI_MSG_RESP_set_proto](#) (`PKI_MSG_RESP *msg`, `PKI_MSG_PROTO` proto)
Sets the messaging protocol to be used.
- `int` [PKI_MSG_RESP_set_recipients](#) (`PKI_MSG_RESP *msg`, `PKI_X509_CERT_STACK *`x_sk)
Sets the list of recipients in a `PKI_MSG_RESP`.
- `int` [PKI_MSG_RESP_set_signer](#) (`PKI_MSG_RESP *msg`, `PKI_X509_CERT *`signer)
Sets the Signer Certificate.
- `int` [PKI_MSG_RESP_set_status](#) (`PKI_MSG_RESP *msg`, `PKI_MSG_STATUS` status)
Sets the status in a `PKI_MSG_RESP` message.

1.54.1 Function Documentation

1.54.1.1 int PKI_MSG_RESP_add_data (PKI_MSG_RESP * *msg*, unsigned char * *data*, size_t *size*)

Adds data to the Message body.

1.54.1.2 int PKI_MSG_RESP_add_recipient (PKI_MSG_RESP * *msg*, PKI_X509_CERT * *x*)

Adds a certificate to the list of recipients.

1.54.1.3 int PKI_MSG_RESP_clear_data (PKI_MSG_RESP * *msg*)

Clears the data of the Message body.

1.54.1.4 int PKI_MSG_RESP_clear_recipients (PKI_MSG_RESP * *msg*)

Clears the list of recipients in a PKI_MSG_RESP.

1.54.1.5 void* PKI_MSG_RESP_encode (PKI_MSG_RESP * *msg*, PKI_MSG_PROTO *proto*)

Encodes the message according to the selected PKI_MSG_PROTO.

1.54.1.6 void PKI_MSG_RESP_free (PKI_MSG_RESP * *msg*)

Free a PKI_MSG_REQ data structure.

1.54.1.7 PKI_MSG_RESP_ACTION PKI_MSG_RESP_get_action (PKI_MSG_RESP * *msg*)

Gets the basic action in a PKI_MSG_RESP message.

1.54.1.8 PKI_X509_CERT* PKI_MSG_RESP_get_cacert (PKI_MSG_RESP * *msg*)

Gets the CA certificate from the Response.

1.54.1.9 void* PKI_MSG_RESP_get_encoded (PKI_MSG_RESP * *msg*)

Gets the encoded version of the Response message.

1.54.1.10 PKI_X509_CERT* PKI_MSG_RESP_get_issued_cert (PKI_MSG_RESP * *msg*)

Gets the certificate from the Response.

1.54.1.11 PKI_X509_KEYPAIR* PKI_MSG_RESP_get_keypair (PKI_MSG_RESP * *msg*)

Gets the Keypair to be used when generating the response.

1.54.1.12 PKI_MSG_PROTO PKI_MSG_RESP_get_proto (PKI_MSG_RESP * *msg*)

Returns the PKI_MSG_PROTO from a PKI_MSG_RESP object.

1.54.1.13 PKI_X509_CERT_STACK* PKI_MSG_RESP_get_recipients (PKI_MSG_RESP * msg)

Gets the list of recipients in a PKI_MSG_RESP.

1.54.1.14 PKI_X509_CERT* PKI_MSG_RESP_get_signer (PKI_MSG_RESP * msg)

Gets the Signer Certificate.

1.54.1.15 PKI_MSG_STATUS PKI_MSG_RESP_get_status (PKI_MSG_RESP * msg)

Gets the status in a PKI_MSG_RESP message.

1.54.1.16 PKI_MSG_RESP* PKI_MSG_RESP_new (PKI_MSG_RESP_ACTION action, PKI_MSG_STATUS status, PKI_X509_KEYPAIR * sign_key, PKI_X509_CERT * signer, PKI_X509_CERT * cacert)

Builds a new message.

1.54.1.17 PKI_MSG_RESP* PKI_MSG_RESP_new_null (void)

Returns an empty generic PKI response message.

1.54.1.18 PKI_MSG_RESP* PKI_MSG_RESP_new_tk (PKI_MSG_RESP_ACTION action, PKI_MSG_STATUS status, PKI_TOKEN * tk)

Builds a new PKI Response message by using a PKI_TOKEN

1.54.1.19 int PKI_MSG_RESP_replace_data (PKI_MSG_RESP * msg, unsigned char * data, size_t size)

Replaces data in a PKI_MSG_RESP message.

1.54.1.20 PKI_X509_SCEP_MSG* PKI_MSG_RESP_SCEP_new (PKI_MSG_RESP * msg)

Returns a SCEP_MSG from the passed PKI_MSG_RESP

1.54.1.21 int PKI_MSG_RESP_set_action (PKI_MSG_RESP * msg, PKI_MSG_RESP_ACTION action)

Sets the basic action in a PKI_MSG_RESP message.

1.54.1.22 int PKI_MSG_RESP_set_cacert (PKI_MSG_RESP * msg, PKI_X509_CERT * x)

Sets the CA certificate in the Response.

1.54.1.23 int PKI_MSG_RESP_set_issued_cert (PKI_MSG_RESP * msg, PKI_X509_CERT * x)

Sets the certificate from the Response.

1.54.1.24 `int PKI_MSG_RESP_set_keypair (PKI_MSG_RESP * msg, PKI_X509_KEYPAIR * pkey)`

Sets the Keypair to be used when generating the response.

1.54.1.25 `int PKI_MSG_RESP_set_proto (PKI_MSG_RESP * msg, PKI_MSG_PROTO proto)`

Sets the messaging protocol to be used.

1.54.1.26 `int PKI_MSG_RESP_set_recipients (PKI_MSG_RESP * msg, PKI_X509_CERT_STACK * x_sk)`

Sets the list of recipients in a PKI_MSG_RESP.

1.54.1.27 `int PKI_MSG_RESP_set_signer (PKI_MSG_RESP * msg, PKI_X509_CERT * signer)`

Sets the Signer Certificate.

1.54.1.28 `int PKI_MSG_RESP_set_status (PKI_MSG_RESP * msg, PKI_MSG_STATUS status)`

Sets the status in a PKI_MSG_RESP message.

1.55 src/pki_threads.c File Reference

Include dependency graph for pki_threads.c:



Functions

- `int PKI_THREAD_create (PKI_THREAD *th, PKI_THREAD_ATTR *attr, void *(*func)(void *), void *arg)`
Spawns a new Thread.
- `PKI_THREAD * PKI_THREAD_new (void *(*func)(void *arg), void *arg)`
Creates and Spawns a new thread.
- `PKI_THREAD_ID PKI_THREAD_self (void)`
Returns the identifier for current thread.

1.55.1 Function Documentation

1.55.1.1 `int PKI_THREAD_create (PKI_THREAD * th, PKI_THREAD_ATTR * attr, void *(*)(void *) func, void * arg)`

Spawns a new Thread.

1.55.1.2 PKI_THREAD* PKI_THREAD_new (void (*)(void *arg) func, void * arg)

Creates and Spawns a new thread.

1.55.1.3 PKI_THREAD_ID PKI_THREAD_self (void)

Returns the identifier for current thread.

1.56 src/pki_threads_vars.c File Reference

Include dependency graph for pki_threads_vars.c:

**Functions**

- timespec * [PKI_clock_gettime](#) (void)
Returns the current time in a timespec structure.
- int [PKI_COND_broadcast](#) (PKI_COND *var)
Broadcasts on a condition variable (wakes up all waiting threads).
- int [PKI_COND_destroy](#) (PKI_COND *var)
Destroys (but do not free memory) a condition variable.
- void [PKI_COND_free](#) (PKI_COND *var)
Frees the memory associated with a Condition Variable.
- int [PKI_COND_init](#) (PKI_COND *var)
Initializes an already allocated (or destroyed) condition variable.
- PKI_COND * [PKI_COND_new](#) ()
Creates a new Condition Variable used for thread management.
- int [PKI_COND_signal](#) (PKI_COND *var)
Signal on a condition variable (wakes up the first waiting thread).
- int [PKI_COND_timedwait](#) (PKI_COND *var, PKI_MUTEX *mutex, struct timespec *t)
Waits on a condition variable for timespec time only.
- int [PKI_COND_wait](#) (PKI_COND *var, PKI_MUTEX *mutex)
Waits on a condition variable.
- int [PKI_MUTEX_acquire](#) (PKI_MUTEX *var)
Acquires access for a mutex.
- int [PKI_MUTEX_destroy](#) (PKI_MUTEX *var)
Destroys (but does not free the memory) a mutex structure.

- void [PKI_MUTEX_free](#) (PKI_MUTEX *var)
Frees the memory associated with a PKI_MUTEX variable.
- int [PKI_MUTEX_init](#) (PKI_MUTEX *var)
Initializes an already allocated mutex structure.
- PKI_MUTEX * [PKI_MUTEX_new](#) ()
Creates a mutex variable to be used for critical sections.
- int [PKI_MUTEX_release](#) (PKI_MUTEX *var)
Releases a mutex.
- int [PKI_RWLOCK_destroy](#) (PKI_RWLOCK *l)
Destroys a R/W Lock (needed before re-initialization).
- void [PKI_RWLOCK_free](#) (PKI_RWLOCK *l)
Destroys a R/W lock and Frees its memory.
- int [PKI_RWLOCK_init](#) (PKI_RWLOCK *l)
Initializes a R/W Lock.
- PKI_RWLOCK * [PKI_RWLOCK_new](#) ()
Allocates a new R/W Lock and Initializes it.
- int [PKI_RWLOCK_read_lock](#) (PKI_RWLOCK *l)
Locks a R/W lock in READ mode (SHARED).
- int [PKI_RWLOCK_release](#) (PKI_RWLOCK *l)
Release a PKI_RWLOCK.
- int [PKI_RWLOCK_try_read_lock](#) (PKI_RWLOCK *l)
- int [PKI_RWLOCK_try_write_lock](#) (PKI_RWLOCK *l)
- int [PKI_RWLOCK_write_lock](#) (PKI_RWLOCK *l)
Locks a R/W lock in WRITE mode (Exclusive).

1.56.1 Function Documentation

1.56.1.1 struct timespec* [PKI_clock_gettime](#) (void)

Returns the current time in a timespec structure.

1.56.1.2 int [PKI_COND_broadcast](#) (PKI_COND * var)

Broadcasts on a condition variable (wakes up all waiting threads).

1.56.1.3 int [PKI_COND_destroy](#) (PKI_COND * var)

Destroys (but do not free memory) a condition variable.

1.56.1.4 void PKI_COND_free (PKI_COND * var)

Frees the memory associated with a Condition Variable.

1.56.1.5 int PKI_COND_init (PKI_COND * var)

Initializes an already allocated (or destroyed) condition variable.

1.56.1.6 PKI_COND* PKI_COND_new ()

Creates a new Condition Variable used for thread management.

1.56.1.7 int PKI_COND_signal (PKI_COND * var)

Signal on a condition variable (wakes up the first waiting thread).

1.56.1.8 int PKI_COND_timedwait (PKI_COND * var, PKI_MUTEX * mutex, struct timespec * t)

Waits on a condition variable for timespec time only.

1.56.1.9 int PKI_COND_wait (PKI_COND * var, PKI_MUTEX * mutex)

Waits on a condition variable.

1.56.1.10 int PKI_MUTEX_acquire (PKI_MUTEX * var)

Acquires access for a mutex.

1.56.1.11 int PKI_MUTEX_destroy (PKI_MUTEX * var)

Destroys (but does not free the memory) a mutex structure.

1.56.1.12 void PKI_MUTEX_free (PKI_MUTEX * var)

Frees the memory associated with a PKI_MUTEX variable.

1.56.1.13 int PKI_MUTEX_init (PKI_MUTEX * var)

Initializes an already allocated mutex structure.

1.56.1.14 PKI_MUTEX* PKI_MUTEX_new ()

Creates a mutex variable to be used for critical sections.

1.56.1.15 int PKI_MUTEX_release (PKI_MUTEX * var)

Releases a mutex.

1.56.1.16 int PKI_RWLOCK_destroy (PKI_RWLOCK * l)

Destroys a R/W Lock (needed before re-initialization).

1.56.1.17 void PKI_RWLOCK_free (PKI_RWLOCK * l)

Destroys a R/W lock and Frees its memory.

1.56.1.18 int PKI_RWLOCK_init (PKI_RWLOCK * l)

Initializes a R/W Lock.

1.56.1.19 PKI_RWLOCK* PKI_RWLOCK_new ()

Allocates a new R/W Lock and Initializes it.

1.56.1.20 int PKI_RWLOCK_read_lock (PKI_RWLOCK * l)

Locks a R/W lock in READ mode (SHARED).

1.56.1.21 int PKI_RWLOCK_release (PKI_RWLOCK * l)

Release a PKI_RWLOCK.

1.56.1.22 int PKI_RWLOCK_try_read_lock (PKI_RWLOCK * l)**1.56.1.23 int PKI_RWLOCK_try_write_lock (PKI_RWLOCK * l)****1.56.1.24 int PKI_RWLOCK_write_lock (PKI_RWLOCK * l)**

Locks a R/W lock in WRITE mode (Exclusive).

1.57 src/pki_x509.c File Reference

Include dependency graph for pki_x509.c:

**Functions**

- const PKI_X509_CALLBACKS * [PKI_X509_CALLBACKS_get](#) (PKI_DATATYPE type, struct hsm_st *hsm)
Returns the callbacks for a specific PKI_DATATYPE.
- int [PKI_X509_delete](#) (PKI_X509 *x)
Deletes the hard copy (eg., file, hsm file, etc.) of the PKI_X509 object.
- PKI_X509 * [PKI_X509_dup](#) (PKI_X509 *x)
Duplicates a PKI_X509 object.
- void * [PKI_X509_dup_value](#) (PKI_X509 *x)
Duplicates the PKI_X509_XXX_VALUE from the passed PKI_X509 object.

- void [PKI_X509_free](#) (PKI_X509 *x)
• void [PKI_X509_free_void](#) (void *x)
Frees the memory associated with a PKI_X509 object.
- void * [PKI_X509_get_data](#) (PKI_X509 *x, PKI_X509_DATA type)
Returns a ref to the X509 data (e.g., SUBJECT) within the passed PKI_X509 object.
- hsm_st * [PKI_X509_get_hsm](#) (PKI_X509 *x)
Retrieves the HSM reference from a PKI_X509 object.
- void * [PKI_X509_get_parsed](#) (PKI_X509 *x, PKI_X509_DATA type)
*Returns the parsed (char *, int *, etc.) version of the data in a PKI_X509 object.*
- URL * [PKI_X509_get_reference](#) (PKI_X509 *x)
Retrieves the reference URL from a PKI_X509 object.
- PKI_DATATYPE [PKI_X509_get_type](#) (PKI_X509 *x)
Returns the type of a PKI_X509 object.
- void * [PKI_X509_get_value](#) (PKI_X509 *x)
Returns the reference to the PKI_X509_XXX_VALUE withing a PKI_X509 object.
- int [PKI_X509_is_signed](#) (PKI_X509 *obj)
Returns PKI_OK if the PKI_X509 object is signed.
- PKI_X509 * [PKI_X509_new](#) (PKI_DATATYPE type, struct hsm_st *hsm)
Allocs the memory associated with an empty PKI_X509 object.
- PKI_X509 * [PKI_X509_new_dup_value](#) (PKI_DATATYPE type, void *value, struct hsm_st *hsm)
Allocates the memory for a new PKI_X509 and duplicates the data.
- PKI_X509 * [PKI_X509_new_value](#) (PKI_DATATYPE type, void *value, struct hsm_st *hsm)
Allocates the memory for a new PKI_X509 and sets the data.
- int [PKI_X509_print_parsed](#) (PKI_X509 *x, PKI_X509_DATA type, int fd)
Prints the parsed data from a PKI_X509 object to a file descriptor.
- int [PKI_X509_set_hsm](#) (PKI_X509 *x, struct hsm_st *hsm)
Sets the HSM reference in a PKI_X509 object.
- int [PKI_X509_set_reference](#) (PKI_X509 *x, URL *url)
Sets (duplicates) the reference URL of a PKI_X509 object.
- int [PKI_X509_set_value](#) (PKI_X509 *x, void *data)
Sets the pointer to the internal value in a PKI_X509.

1.57.1 Function Documentation

1.57.1.1 `const PKI_X509_CALLBACKS* PKI_X509_CALLBACKS_get (PKI_DATATYPE type, struct hsm_st * hsm)`

Returns the callbacks for a specific PKI_DATATYPE.

1.57.1.2 `int PKI_X509_delete (PKI_X509 * x)`

Deletes the hard copy (eg., file, hsm file, etc.) of the PKI_X509 object.

1.57.1.3 `PKI_X509* PKI_X509_dup (PKI_X509 * x)`

Duplicates a PKI_X509 object.

1.57.1.4 `void* PKI_X509_dup_value (PKI_X509 * x)`

Duplicates the PKI_X509_XXX_VALUE from the passed PKI_X509 object.

1.57.1.5 `void PKI_X509_free (PKI_X509 * x)`

1.57.1.6 `void PKI_X509_free_void (void * x)`

Frees the memory associated with a PKI_X509 object.

1.57.1.7 `void* PKI_X509_get_data (PKI_X509 * x, PKI_X509_DATA type)`

Returns a ref to the X509 data (e.g., SUBJECT) within the passed PKI_X509 object.

1.57.1.8 `struct hsm_st* PKI_X509_get_hsm (PKI_X509 * x)`

Retrieves the HSM reference from a PKI_X509 object.

1.57.1.9 `void* PKI_X509_get_parsed (PKI_X509 * x, PKI_X509_DATA type)`

Returns the parsed (char *, int *, etc.) version of the data in a PKI_X509 object.

1.57.1.10 `URL* PKI_X509_get_reference (PKI_X509 * x)`

Retrieves the reference URL from a PKI_X509 object.

1.57.1.11 `PKI_DATATYPE PKI_X509_get_type (PKI_X509 * x)`

Returns the type of a PKI_X509 object.

1.57.1.12 `void* PKI_X509_get_value (PKI_X509 * x)`

Returns the reference to the PKI_X509_XXX_VALUE withing a PKI_X509 object.

1.57.1.13 `int PKI_X509_is_signed (PKI_X509 * obj)`

Returns PKI_OK if the PKI_X509 object is signed.

1.57.1.14 `PKI_X509* PKI_X509_new (PKI_DATATYPE type, struct hsm_st * hsm)`

Allocates the memory associated with an empty PKI_X509 object.

1.57.1.15 `PKI_X509* PKI_X509_new_dup_value (PKI_DATATYPE type, void * value, struct hsm_st * hsm)`

Allocates the memory for a new PKI_X509 and duplicates the data.

1.57.1.16 `PKI_X509* PKI_X509_new_value (PKI_DATATYPE type, void * value, struct hsm_st * hsm)`

Allocates the memory for a new PKI_X509 and sets the data.

1.57.1.17 `int PKI_X509_print_parsed (PKI_X509 * x, PKI_X509_DATA type, int fd)`

Prints the parsed data from a PKI_X509 object to a file descriptor.

1.57.1.18 `int PKI_X509_set_hsm (PKI_X509 * x, struct hsm_st * hsm)`

Sets the HSM reference in a PKI_X509 object.

1.57.1.19 `int PKI_X509_set_reference (PKI_X509 * x, URL * url)`

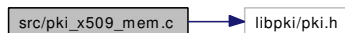
Sets (duplicates) the reference URL of a PKI_X509 object.

1.57.1.20 `int PKI_X509_set_value (PKI_X509 * x, void * data)`

Sets the pointer to the internal value in a PKI_X509.

1.58 src/pki_x509_mem.c File Reference

Include dependency graph for pki_x509_mem.c:

**Functions**

- `PKI_X509 * PKI_X509_get_mem (PKI_MEM *mem, PKI_DATATYPE type, PKI_CRED *cred, HSM *hsm)`

Reads a PKI_X509 object from a PKI_MEM.

- `void * PKI_X509_get_mem_value (PKI_MEM *mem, PKI_DATATYPE type, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_put_mem (PKI_X509 *x, PKI_DATA_FORMAT format, PKI_MEM **mem, PKI_CRED *cred)`

Writes a PKI_X509 object to a PKI_MEM structure.

- `PKI_MEM * PKI_X509_put_mem_value (void *x, PKI_DATATYPE type, PKI_MEM **pki_mem, PKI_DATA_FORMAT format, PKI_CRED *cred, HSM *hsm)`

Writes a PKI_X509_XXX_VALUE to a PKI_MEM structure.

- PKI_X509_STACK * [PKI_X509_STACK_get_mem](#) (PKI_MEM *mem, PKI_DATATYPE type, PKI_CRED *cred, HSM *hsm)

Returns a stack of objects read from the passed PKI_MEM.

- PKI_MEM * [PKI_X509_STACK_put_mem](#) (PKI_X509_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)

Writes a stack of PKI_X509 to a PKI_MEM.

1.58.1 Function Documentation

1.58.1.1 PKI_X509* PKI_X509_get_mem (PKI_MEM * mem, PKI_DATATYPE type, PKI_CRED * cred, HSM * hsm)

Reads a PKI_X509 object from a PKI_MEM.

1.58.1.2 void* PKI_X509_get_mem_value (PKI_MEM * mem, PKI_DATATYPE type, PKI_CRED * cred, HSM * hsm)

1.58.1.3 PKI_MEM* PKI_X509_put_mem (PKI_X509 * x, PKI_DATA_FORMAT format, PKI_MEM ** mem, PKI_CRED * cred)

Writes a PKI_X509 object to a PKI_MEM structure.

1.58.1.4 PKI_MEM* PKI_X509_put_mem_value (void * x, PKI_DATATYPE type, PKI_MEM ** pki_mem, PKI_DATA_FORMAT format, PKI_CRED * cred, HSM * hsm)

Writes a PKI_X509_XXX_VALUE to a PKI_MEM structure.

1.58.1.5 PKI_X509_STACK* PKI_X509_STACK_get_mem (PKI_MEM * mem, PKI_DATATYPE type, PKI_CRED * cred, HSM * hsm)

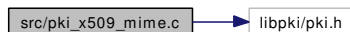
Returns a stack of objects read from the passed PKI_MEM.

1.58.1.6 PKI_MEM* PKI_X509_STACK_put_mem (PKI_X509_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)

Writes a stack of PKI_X509 to a PKI_MEM.

1.59 src/pki_x509_mime.c File Reference

Include dependency graph for pki_x509_mime.c:



Functions

- const char * [PKI_X509_get_mimetype](#) (PKI_DATATYPE type)

1.59.1 Function Documentation

1.59.1.1 `const char* PKI_X509_get_mimetype (PKI_DATATYPE type)`

1.60 src/profile.c File Reference

Include dependency graph for profile.c:



Functions

- PKI_CONFIG_ELEMENT * [PKI_X509_PROFILE_add_extension](#) (PKI_X509_PROFILE *doc, char *name, char *value, char *type, int crit)
- int [PKI_X509_PROFILE_free](#) (PKI_X509_PROFILE *doc)
- void [PKI_X509_PROFILE_free_void](#) (void *doc)
- PKI_X509_PROFILE * [PKI_X509_PROFILE_get_default](#) (PKI_X509_PROFILE_TYPE profile_id)
- PKI_X509_EXTENSION * [PKI_X509_PROFILE_get_ext_by_num](#) (PKI_X509_PROFILE *doc, int num)
- PKI_CONFIG_ELEMENT * [PKI_X509_PROFILE_get_extensions](#) (PKI_X509_PROFILE *doc)
- int [PKI_X509_PROFILE_get_exts_num](#) (PKI_X509_PROFILE *doc)
- char * [PKI_X509_PROFILE_get_name](#) (PKI_X509_PROFILE *doc)
- char * [PKI_X509_PROFILE_get_value](#) (PKI_X509_PROFILE *doc, char *path)
- PKI_X509_PROFILE * [PKI_X509_PROFILE_load](#) (char *urlPath)
- PKI_X509_PROFILE * [PKI_X509_PROFILE_new](#) (char *name)

Create a new PKI_X509_PROFILE.

- int [PKI_X509_PROFILE_put_file](#) (PKI_X509_PROFILE *doc, char *url)

1.60.1 Function Documentation

1.60.1.1 `PKI_CONFIG_ELEMENT* PKI_X509_PROFILE_add_extension (PKI_X509_PROFILE * doc, char * name, char * value, char * type, int crit)`

1.60.1.2 `int PKI_X509_PROFILE_free (PKI_X509_PROFILE * doc)`

1.60.1.3 `void PKI_X509_PROFILE_free_void (void * doc)`

1.60.1.4 `PKI_X509_PROFILE* PKI_X509_PROFILE_get_default (PKI_X509_PROFILE_TYPE profile_id)`

1.60.1.5 `PKI_X509_EXTENSION* PKI_X509_PROFILE_get_ext_by_num (PKI_X509_PROFILE * doc, int num)`

1.60.1.6 `PKI_CONFIG_ELEMENT* PKI_X509_PROFILE_get_extensions (PKI_X509_PROFILE * doc)`

1.60.1.7 `int PKI_X509_PROFILE_get_exts_num (PKI_X509_PROFILE * doc)`

1.60.1.8 `char* PKI_X509_PROFILE_get_name (PKI_X509_PROFILE * doc)`

1.60.1.9 `char* PKI_X509_PROFILE_get_value (PKI_X509_PROFILE * doc, char * path)`

1.60.1.10 `PKI_X509_PROFILE* PKI_X509_PROFILE_load (char * urlPath)`

1.60.1.11 `PKI_X509_PROFILE* PKI_X509_PROFILE_new (char * name)`

Create a new PKI_X509_PROFILE.

1.60.1.12 `int PKI_X509_PROFILE_put_file (PKI_X509_PROFILE * doc, char * url)`

1.61 src/prqp/asn1_req.c File Reference

Include dependency graph for `asn1_req.c`:



1.62 src/prqp/asn1_res.c File Reference

Include dependency graph for `asn1_res.c`:



1.63 src/prqp/http_client.c File Reference

Include dependency graph for `http_client.c`:



Functions

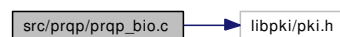
- `PKI_X509_PRQP_RESP * PKI_X509_PRQP_RESP_get_http (URL *url, PKI_X509_PRQP_REQ *req, unsigned long max_size)`

1.63.1 Function Documentation

1.63.1.1 `PKI_X509_PRQP_RESP* PKI_X509_PRQP_RESP_get_http` (URL * *url*, PKI_X509_PRQP_REQ * *req*, unsigned long *max_size*)

1.64 src/prqp/prqp_bio.c File Reference

Include dependency graph for prqp_bio.c:



Functions

- `PKI_PRQP_REQ * d2i_PRQP_REQ_bio` (BIO **bp*, PKI_PRQP_REQ **p*)
- `PKI_PRQP_RESP * d2i_PRQP_RESP_bio` (BIO **bp*, PKI_PRQP_RESP **p*)
- `int i2d_PRQP_REQ_bio` (BIO **bp*, PKI_PRQP_REQ **o*)
- `int i2d_PRQP_RESP_bio` (BIO **bp*, PKI_PRQP_RESP **o*)
- `PKI_PRQP_REQ * PEM_read_bio_PRQP_REQ` (BIO **bp*)
- `PKI_PRQP_RESP * PEM_read_bio_PRQP_RESP` (BIO **bp*)
- `int PEM_write_bio_PRQP_REQ` (BIO **bp*, PKI_PRQP_REQ **o*)
- `int PEM_write_bio_PRQP_RESP` (BIO **bp*, PKI_PRQP_RESP **o*)

1.64.1 Function Documentation

1.64.1.1 `PKI_PRQP_REQ* d2i_PRQP_REQ_bio` (BIO * *bp*, PKI_PRQP_REQ * *p*)

1.64.1.2 `PKI_PRQP_RESP* d2i_PRQP_RESP_bio` (BIO * *bp*, PKI_PRQP_RESP * *p*)

1.64.1.3 `int i2d_PRQP_REQ_bio` (BIO * *bp*, PKI_PRQP_REQ * *o*)

1.64.1.4 `int i2d_PRQP_RESP_bio` (BIO * *bp*, PKI_PRQP_RESP * *o*)

1.64.1.5 `PKI_PRQP_REQ* PEM_read_bio_PRQP_REQ` (BIO * *bp*)

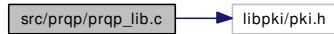
1.64.1.6 `PKI_PRQP_RESP* PEM_read_bio_PRQP_RESP` (BIO * *bp*)

1.64.1.7 `int PEM_write_bio_PRQP_REQ` (BIO * *bp*, PKI_PRQP_REQ * *o*)

1.64.1.8 `int PEM_write_bio_PRQP_RESP` (BIO * *bp*, PKI_PRQP_RESP * *o*)

1.65 src/prqp/prqp_lib.c File Reference

Include dependency graph for prqp_lib.c:



Defines

- `#define __PKI_PRQP_LIB_C__`

Functions

- `int CERT_IDENTIFIER_cmp (CERT_IDENTIFIER *a, CERT_IDENTIFIER *b)`
- `CERT_IDENTIFIER * PKI_PRQP_CERTID_new (PKI_X509_NAME *caName, PKI_X509_NAME *caIssuerName, PKI_INTEGER *serial, PKI_STRING *caCertHash, PKI_STRING *caKeyHash, PKI_STRING *caKeyId, PKI_STRING *issKeyId, PKI_DIGEST_ALG *dgst)`
- `CERT_IDENTIFIER * PKI_PRQP_CERTID_new_cert (PKI_X509_CERT *caCert, PKI_X509_CERT *issuerCert, PKI_X509_CERT *issuedCert, char *subject_s, char *serial_s, PKI_DIGEST_ALG *dgst)`

Generates a new CERT_IDENTIFIER to be used in a PRQP request.

- `PKI_X509_PRQP_REQ * PKI_PRQP_REQ_new_cert (PKI_X509_CERT *caCert, PKI_X509_CERT *caIssuerCert, PKI_X509_CERT *issuedCert, char *subject_s, char *serial_s, PKI_DIGEST_ALG *md)`
- `PKI_INTEGER * PKI_X509_PRQP_NONCE_new (int bits)`
- `int PKI_X509_PRQP_REQ_add_service (PKI_X509_PRQP_REQ *p, char *ss)`

Adds a service identifier to a PRQP REQUEST.

- `int PKI_X509_PRQP_REQ_add_service_stack (PKI_X509_PRQP_REQ *p, PKI_STACK *sk_services)`

Adds a stack of services to a PRQP REQUEST.

- `void PKI_X509_PRQP_REQ_free (PKI_X509_PRQP_REQ *x)`
- `void PKI_X509_PRQP_REQ_free_void (void *x)`
- `void * PKI_X509_PRQP_REQ_get_data (PKI_X509_PRQP_REQ *obj, PKI_X509_DATA type)`

Returns data pointers from a PRQP request.

- `PKI_X509_PRQP_REQ * PKI_X509_PRQP_REQ_new_certs_res (PKI_X509_CERT *caCert, PKI_X509_CERT *caIssuerCert, PKI_X509_CERT *issuedCert, PKI_STACK *sk_services)`
- `void * PKI_X509_PRQP_REQ_new_null (void)`
- `PKI_X509_PRQP_REQ * PKI_X509_PRQP_REQ_new_url (char *ca_cert_s, char *ca_issuer_cert_s, char *issued_cert_s, char *subject_s, char *serial_s, EVP_MD *md)`
- `int PKI_X509_PRQP_REQ_print (PKI_X509_PRQP_REQ *req)`

Prints out the contents of a PRQP_REQ on stdout.

- `int PKI_X509_PRQP_REQ_print_fp (FILE *fp, PKI_X509_PRQP_REQ *req)`

Writes a PRQP_REQ in text format in the passed file pointer.

- `int PKI_X509_PRQP_REQ_VALUE_print_bio (PKI_PRQP_REQ *req, BIO *bio)`

- int [PKI_X509_PRQP_REQ_verify](#) (PKI_X509_PRQP_REQ *r)
Verify Signature on the PRQP REQUEST.
- int [PKI_X509_PRQP_RESP_add_referrals](#) (PKI_X509_PRQP_RESP *resp, PKI_STACK *referrals)
Adds a stack of referrals (PKI_STACK) to a PRQP_RESP object.
- int [PKI_X509_PRQP_RESP_add_service](#) (PKI_X509_PRQP_RESP *r, PKI_OID *resId, char *url, long long version, char *comment, PKI_OID *oid)
Adds a new service (single URL) to the stack of services in a PRQP_RESP.
- int [PKI_X509_PRQP_RESP_add_service_stack](#) (PKI_X509_PRQP_RESP *r, PKI_OID *resId, PKI_STACK *url_stack, long long version, char *comment, PKI_OID *oid)
Adds a new service (stack of URLs) to the stack of services in a PRQP_RESP.
- void [PKI_X509_PRQP_RESP_free](#) (PKI_X509_PRQP_RESP *x)
Releases the memory associated with a PRQP_RESP object.
- void [PKI_X509_PRQP_RESP_free_void](#) (void *x)
- void * [PKI_X509_PRQP_RESP_get_data](#) (PKI_X509_PRQP_RESP *obj, PKI_X509_DATA type)
Returns a pointer to the specified PKI_X509_DATA field of a PRQP response.
- int [PKI_X509_PRQP_RESP_get_status](#) (PKI_X509_PRQP_RESP *obj)
Returns the PKI_X509_PRQP_STATUS associated to a PRQP response.
- void * [PKI_X509_PRQP_RESP_new_null](#) (void)
Creates a new PRQP_RESP empty object.
- PKI_X509_PRQP_RESP * [PKI_X509_PRQP_RESP_new_req](#) (PKI_X509_PRQP_RESP **resp_pnt, PKI_X509_PRQP_REQ *x_req, int status, long secs)
Created a new PRQP_RESP from the contents of a PRQP_REQ.
- int [PKI_X509_PRQP_RESP_nonce_dup](#) (PKI_X509_PRQP_RESP *resp, PKI_X509_PRQP_REQ *req)
Duplicates the NONCE from a PRQP_REQ to a PRQP_RESP.
- int [PKI_X509_PRQP_RESP_pkistatus_set](#) (PKI_X509_PRQP_RESP *resp, long v, char *info)
Sets the status of a PKI_X509_PRQP_RESP object.
- int [PKI_X509_PRQP_RESP_print](#) (PKI_X509_PRQP_RESP *resp)
Prints out the contents of a PRQP_RESP to stdout.
- int [PKI_X509_PRQP_RESP_print_fp](#) (FILE *fp, PKI_X509_PRQP_RESP *resp)
Writes a PRQP_RESP in text format to the passed file pointer.
- PKI_STACK * [PKI_X509_PRQP_RESP_url_sk](#) (PKI_X509_PRQP_RESP *r)
Returns a PKI_STACK of URLs from a PRQP_RESP object.
- int [PKI_X509_PRQP_RESP_VALUE_print_bio](#) (PKI_X509_PRQP_RESP_VALUE *resp, BIO *bio)

- int [PKI_X509_PRQP_RESP_verify](#) (PKI_X509_PRQP_RESP *r)
- int [PKI_X509_PRQP_RESP_version_set](#) (PKI_X509_PRQP_RESP *resp, int ver)
Sets the protocol version of a PRQP_RESP object.
- int [PKI_X509_PRQP_sign](#) (PKI_X509 *obj, PKI_X509_KEYPAIR *k, PKI_X509_CERT *x, PKI_DIGEST_ALG *dgst, PKI_X509_CERT_STACK *certs)
Signs a PRQP object.
- int [PKI_X509_PRQP_sign_tk](#) (PKI_X509_PRQP_RESP *resp, PKI_TOKEN *tk, PKI_DIGEST_ALG *dgst)
Signs a PRQP object by using a provided TOKEN object.
- int [PKI_X509_PRQP_verify](#) (PKI_X509 *r)
Verifies that the signature on a PRQP object is correct.
- int [PRQP_init_all_services](#) (void)
- PKI_OID * [PRQP_RESOURCE_RESPONSE_TOKEN_get_oid](#) (RESOURCE_RESPONSE_TOKEN *rrt)
- PKI_STACK * [PRQP_RESOURCE_RESPONSE_TOKEN_get_services](#) (RESOURCE_RESPONSE_TOKEN *rrt)

Variables

- char * [PKI_X509_PRQP_STATUS_STRING](#) []

1.65.1 Define Documentation

1.65.1.1 #define __PKI_PRQP_LIB_C__

1.65.2 Function Documentation

1.65.2.1 int CERT_IDENTIFIER_cmp (CERT_IDENTIFIER *a, CERT_IDENTIFIER *b)

1.65.2.2 CERT_IDENTIFIER* PKI_PRQP_CERTID_new (PKI_X509_NAME *caName, PKI_X509_NAME *caIssuerName, PKI_INTEGER *serial, PKI_STRING *caCertHash, PKI_STRING *caKeyHash, PKI_STRING *caKeyId, PKI_STRING *issKeyId, PKI_DIGEST_ALG *dgst)

1.65.2.3 CERT_IDENTIFIER* PKI_PRQP_CERTID_new_cert (PKI_X509_CERT *caCert, PKI_X509_CERT *issuerCert, PKI_X509_CERT *issuedCert, char *subject_s, char *serial_s, PKI_DIGEST_ALG *dgst)

Generates a new CERT_IDENTIFIER to be used in a PRQP request.

1.65.2.4 PKI_X509_PRQP_REQ* PKI_PRQP_REQ_new_cert (PKI_X509_CERT *caCert, PKI_X509_CERT *caIssuerCert, PKI_X509_CERT *issuedCert, char *subject_s, char *serial_s, PKI_DIGEST_ALG *md)

1.65.2.5 PKI_INTEGER* PKI_X509_PRQP_NONCE_new (int bits)

1.65.2.6 int PKI_X509_PRQP_REQ_add_service (PKI_X509_PRQP_REQ * *p*, char * *ss*)

Adds a service identifier to a PRQP REQUEST.

1.65.2.7 int PKI_X509_PRQP_REQ_add_service_stack (PKI_X509_PRQP_REQ * *p*, PKI_STACK * *sk_services*)

Adds a stack of services to a PRQP REQUEST.

1.65.2.8 void PKI_X509_PRQP_REQ_free (PKI_X509_PRQP_REQ * *x*)**1.65.2.9 void PKI_X509_PRQP_REQ_free_void (void * *x*)****1.65.2.10 void* PKI_X509_PRQP_REQ_get_data (PKI_X509_PRQP_REQ * *obj*, PKI_X509_DATA *type*)**

Returns data pointers from a PRQP request.

1.65.2.11 PKI_X509_PRQP_REQ* PKI_X509_PRQP_REQ_new_certs_res (PKI_X509_CERT * *caCert*, PKI_X509_CERT * *caIssuerCert*, PKI_X509_CERT * *issuedCert*, PKI_STACK * *sk_services*)**1.65.2.12 void* PKI_X509_PRQP_REQ_new_null (void)****1.65.2.13 PKI_X509_PRQP_REQ* PKI_X509_PRQP_REQ_new_url (char * *ca_cert_s*, char * *ca_issuer_cert_s*, char * *issued_cert_s*, char * *subject_s*, char * *serial_s*, EVP_MD * *md*)****1.65.2.14 int PKI_X509_PRQP_REQ_print (PKI_X509_PRQP_REQ * *req*)**

Prints out the contents of a PRQP_REQ on stdout.

1.65.2.15 int PKI_X509_PRQP_REQ_print_fp (FILE * *fp*, PKI_X509_PRQP_REQ * *req*)

Writes a PRQP_REQ in text format in the passed file pointer.

1.65.2.16 int PKI_X509_PRQP_REQ_VALUE_print_bio (PKI_PRQP_REQ * *req*, BIO * *bio*)**1.65.2.17 int PKI_X509_PRQP_REQ_verify (PKI_X509_PRQP_REQ * *r*)**

Verify Signature on the PRQP REQUEST.

1.65.2.18 int PKI_X509_PRQP_RESP_add_referrals (PKI_X509_PRQP_RESP * *resp*, PKI_STACK * *referrals*)

Adds a stack of referrals (PKI_STACK) to a PRQP_RESP object.

1.65.2.19 int PKI_X509_PRQP_RESP_add_service (PKI_X509_PRQP_RESP * *r*, PKI_OID * *res-Id*, char * *url*, long long *version*, char * *comment*, PKI_OID * *oid*)

Adds a new service (single URL) to the stack of services in a PRQP_RESP.

1.65.2.20 `int PKI_X509_PRQP_RESP_add_service_stack (PKI_X509_PRQP_RESP * r, PKI_OID * resId, PKI_STACK * url_stack, long long version, char * comment, PKI_OID * oid)`

Adds a new service (stack of URLs) to the stack of services in a PRQP_RESP.

1.65.2.21 `void PKI_X509_PRQP_RESP_free (PKI_X509_PRQP_RESP * x)`

Releases the memory associated with a PRQP_RESP object.

1.65.2.22 `void PKI_X509_PRQP_RESP_free_void (void * x)`

1.65.2.23 `void* PKI_X509_PRQP_RESP_get_data (PKI_X509_PRQP_RESP * obj, PKI_X509_DATA type)`

Returns a pointer to the specified PKI_X509_DATA field of a PRQP response.

1.65.2.24 `int PKI_X509_PRQP_RESP_get_status (PKI_X509_PRQP_RESP * obj)`

Returns the PKI_X509_PRQP_STATUS associated to a PRQP response.

1.65.2.25 `void* PKI_X509_PRQP_RESP_new_null (void)`

Creates a new PRQP_RESP empty object.

1.65.2.26 `PKI_X509_PRQP_RESP* PKI_X509_PRQP_RESP_new_req (PKI_X509_PRQP_RESP ** resp_pnt, PKI_X509_PRQP_REQ * x_req, int status, long secs)`

Creates a new PRQP_RESP from the contents of a PRQP_REQ.

1.65.2.27 `int PKI_X509_PRQP_RESP_nonce_dup (PKI_X509_PRQP_RESP * resp, PKI_X509_PRQP_REQ * req)`

Duplicates the NONCE from a PRQP_REQ to a PRQP_RESP.

1.65.2.28 `int PKI_X509_PRQP_RESP_pkistatus_set (PKI_X509_PRQP_RESP * resp, long v, char * info)`

Sets the status of a PKI_X509_PRQP_RESP object.

1.65.2.29 `int PKI_X509_PRQP_RESP_print (PKI_X509_PRQP_RESP * resp)`

Prints out the contents of a PRQP_RESP to stdout.

1.65.2.30 `int PKI_X509_PRQP_RESP_print_fp (FILE * fp, PKI_X509_PRQP_RESP * resp)`

Writes a PRQP_RESP in text format to the passed file pointer.

1.65.2.31 `PKI_STACK* PKI_X509_PRQP_RESP_url_sk (PKI_X509_PRQP_RESP * r)`

Returns a PKI_STACK of URLs from a PRQP_RESP object.

1.65.2.32 `int PKI_X509_PRQP_RESP_VALUE_print_bio (PKI_X509_PRQP_RESP_VALUE * resp, BIO * bio)`

1.65.2.33 `int PKI_X509_PRQP_RESP_verify (PKI_X509_PRQP_RESP * r)`

1.65.2.34 `int PKI_X509_PRQP_RESP_version_set (PKI_X509_PRQP_RESP * resp, int ver)`

Sets the protocol version of a PRQP_RESP object.

1.65.2.35 `int PKI_X509_PRQP_sign (PKI_X509 * obj, PKI_X509_KEYPAIR * k, PKI_X509_CERT * x, PKI_DIGEST_ALG * dgst, PKI_X509_CERT_STACK * certs)`

Signs a PRQP object.

1.65.2.36 `int PKI_X509_PRQP_sign_tk (PKI_X509_PRQP_RESP * resp, PKI_TOKEN * tk, PKI_DIGEST_ALG * dgst)`

Signs a PRQP object by using a provided TOKEN object.

1.65.2.37 `int PKI_X509_PRQP_verify (PKI_X509 * r)`

Verifies that the signature on a PRQP object is correct.

1.65.2.38 `int PRQP_init_all_services (void)`

1.65.2.39 `PKI_OID* PRQP_RESOURCE_RESPONSE_TOKEN_get_oid (RESOURCE_RESPONSE_TOKEN * rrt)`

1.65.2.40 `PKI_STACK* PRQP_RESOURCE_RESPONSE_TOKEN_get_services (RESOURCE_RESPONSE_TOKEN * rrt)`

1.65.3 Variable Documentation

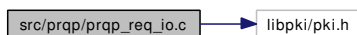
1.65.3.1 `char* PKI_X509_PRQP_STATUS_STRING[]`

Initial value:

```
{
    PKI_X509_PRQP_STATUS_STRING_OK,
    PKI_X509_PRQP_STATUS_STRING_BAD_REQUEST,
    PKI_X509_PRQP_STATUS_STRING_CA_NOT_PRESENT,
    PKI_X509_PRQP_STATUS_STRING_SYS_FAILURE
}
```

1.66 src/prqp/prqp_req_io.c File Reference

Include dependency graph for prqp_req_io.c:



Functions

- PKI_X509_PRQP_REQ * [PKI_X509_PRQP_REQ_get](#) (char *url_s, PKI_CRED *cred, HSM *hsm)
- PKI_X509_PRQP_REQ * [PKI_X509_PRQP_REQ_get_mem](#) (PKI_MEM *mem, PKI_CRED *cred, HSM *hsm)
- PKI_X509_PRQP_REQ * [PKI_X509_PRQP_REQ_get_url](#) (URL *url, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_PRQP_REQ_put](#) (PKI_X509_PRQP_REQ *req, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_PRQP_REQ_put_mem](#) (PKI_X509_PRQP_REQ *req, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_PRQP_REQ_put_url](#) (PKI_X509_PRQP_REQ *req, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_X509_PRQP_REQ_STACK * [PKI_X509_PRQP_REQ_STACK_get](#) (char *url_s, PKI_CRED *cred, HSM *hsm)
- PKI_X509_PRQP_REQ_STACK * [PKI_X509_PRQP_REQ_STACK_get_mem](#) (PKI_MEM *mem, PKI_CRED *cred, HSM *hsm)
- PKI_X509_PRQP_REQ_STACK * [PKI_X509_PRQP_REQ_STACK_get_url](#) (URL *url, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_PRQP_REQ_STACK_put](#) (PKI_X509_PRQP_REQ_STACK *sk, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)
- PKI_MEM * [PKI_X509_PRQP_REQ_STACK_put_mem](#) (PKI_X509_PRQP_REQ_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)
- int [PKI_X509_PRQP_REQ_STACK_put_url](#) (PKI_X509_PRQP_REQ_STACK *sk, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)

1.66.1 Function Documentation

1.66.1.1 PKI_X509_PRQP_REQ* [PKI_X509_PRQP_REQ_get](#) (char * url_s, PKI_CRED * cred, HSM * hsm)

1.66.1.2 PKI_X509_PRQP_REQ* [PKI_X509_PRQP_REQ_get_mem](#) (PKI_MEM * mem, PKI_CRED * cred, HSM * hsm)

1.66.1.3 PKI_X509_PRQP_REQ* [PKI_X509_PRQP_REQ_get_url](#) (URL * url, PKI_CRED * cred, HSM * hsm)

1.66.1.4 int [PKI_X509_PRQP_REQ_put](#) (PKI_X509_PRQP_REQ * req, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)

1.66.1.5 PKI_MEM* [PKI_X509_PRQP_REQ_put_mem](#) (PKI_X509_PRQP_REQ * req, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)

1.66.1.6 int [PKI_X509_PRQP_REQ_put_url](#) (PKI_X509_PRQP_REQ * req, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)

1.66.1.7 PKI_X509_PRQP_REQ_STACK* [PKI_X509_PRQP_REQ_STACK_get](#) (char * url_s, PKI_CRED * cred, HSM * hsm)

1.66.1.8 `PKI_X509_PRQP_REQ_STACK* PKI_X509_PRQP_REQ_STACK_get_mem (PKI_MEM * mem, PKI_CRED * cred, HSM * hsm)`

1.66.1.9 `PKI_X509_PRQP_REQ_STACK* PKI_X509_PRQP_REQ_STACK_get_url (URL * url, PKI_CRED * cred, HSM * hsm)`

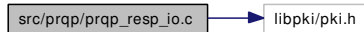
1.66.1.10 `int PKI_X509_PRQP_REQ_STACK_put (PKI_X509_PRQP_REQ_STACK * sk, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.66.1.11 `PKI_MEM* PKI_X509_PRQP_REQ_STACK_put_mem (PKI_X509_PRQP_REQ_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.66.1.12 `int PKI_X509_PRQP_REQ_STACK_put_url (PKI_X509_PRQP_REQ_STACK * sk, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.67 src/prqp/prqp_resp_io.c File Reference

Include dependency graph for prqp_resp_io.c:



Functions

- `PKI_X509_PRQP_RESP * PKI_X509_PRQP_RESP_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_PRQP_RESP * PKI_X509_PRQP_RESP_get_mem (PKI_MEM *mem, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_PRQP_RESP * PKI_X509_PRQP_RESP_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_PRQP_RESP_put (PKI_X509_PRQP_RESP *resp, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_PRQP_RESP_put_mem (PKI_X509_PRQP_RESP *resp, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_PRQP_RESP_put_url (PKI_X509_PRQP_RESP *resp, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_PRQP_RESP_STACK * PKI_X509_PRQP_RESP_STACK_get (char *url_s, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_PRQP_RESP_STACK * PKI_X509_PRQP_RESP_STACK_get_mem (PKI_MEM *mem, PKI_CRED *cred, HSM *hsm)`
- `PKI_X509_PRQP_RESP_STACK * PKI_X509_PRQP_RESP_STACK_get_url (URL *url, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_PRQP_RESP_STACK_put (PKI_X509_PRQP_RESP_STACK *sk, PKI_DATA_FORMAT format, char *url_s, char *mime, PKI_CRED *cred, HSM *hsm)`
- `PKI_MEM * PKI_X509_PRQP_RESP_STACK_put_mem (PKI_X509_PRQP_RESP_STACK *sk, PKI_DATA_FORMAT format, PKI_MEM **pki_mem, PKI_CRED *cred, HSM *hsm)`
- `int PKI_X509_PRQP_RESP_STACK_put_url (PKI_X509_PRQP_RESP_STACK *sk, PKI_DATA_FORMAT format, URL *url, char *mime, PKI_CRED *cred, HSM *hsm)`

1.67.1 Function Documentation

1.67.1.1 `PKI_X509_PRQP_RESP* PKI_X509_PRQP_RESP_get (char * url_s, PKI_CRED * cred, HSM * hsm)`

1.67.1.2 `PKI_X509_PRQP_RESP* PKI_X509_PRQP_RESP_get_mem (PKI_MEM * mem, PKI_CRED * cred, HSM * hsm)`

1.67.1.3 `PKI_X509_PRQP_RESP* PKI_X509_PRQP_RESP_get_url (URL * url, PKI_CRED * cred, HSM * hsm)`

1.67.1.4 `int PKI_X509_PRQP_RESP_put (PKI_X509_PRQP_RESP * resp, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.67.1.5 `PKI_MEM* PKI_X509_PRQP_RESP_put_mem (PKI_X509_PRQP_RESP * resp, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.67.1.6 `int PKI_X509_PRQP_RESP_put_url (PKI_X509_PRQP_RESP * resp, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.67.1.7 `PKI_X509_PRQP_RESP_STACK* PKI_X509_PRQP_RESP_STACK_get (char * url_s, PKI_CRED * cred, HSM * hsm)`

1.67.1.8 `PKI_X509_PRQP_RESP_STACK* PKI_X509_PRQP_RESP_STACK_get_mem (PKI_MEM * mem, PKI_CRED * cred, HSM * hsm)`

1.67.1.9 `PKI_X509_PRQP_RESP_STACK* PKI_X509_PRQP_RESP_STACK_get_url (URL * url, PKI_CRED * cred, HSM * hsm)`

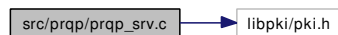
1.67.1.10 `int PKI_X509_PRQP_RESP_STACK_put (PKI_X509_PRQP_RESP_STACK * sk, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

1.67.1.11 `PKI_MEM* PKI_X509_PRQP_RESP_STACK_put_mem (PKI_X509_PRQP_RESP_STACK * sk, PKI_DATA_FORMAT format, PKI_MEM ** pki_mem, PKI_CRED * cred, HSM * hsm)`

1.67.1.12 `int PKI_X509_PRQP_RESP_STACK_put_url (PKI_X509_PRQP_RESP_STACK * sk, PKI_DATA_FORMAT format, URL * url, char * mime, PKI_CRED * cred, HSM * hsm)`

1.68 src/prqp/prqp_srv.c File Reference

Include dependency graph for prqp_srv.c:



Functions

- `PKI_X509_PRQP_RESP * PKI_DISCOVER_get_resp (PKI_X509_PRQP_REQ *p, char *url_s)`
Retireve a PRQP response from the passed url or from one of the configured RQAs in /etc/pki.conf.
- `PKI_X509_PRQP_RESP * PKI_DISCOVER_get_resp_url (PKI_X509_PRQP_REQ *p, URL *url)`
Retrieve a PRQP Response from a server.
- `PKI_STACK * PKI_get_ca_resources (PKI_X509_CERT *caCert, PKI_X509_CERT *caIssuerCert, PKI_X509_CERT *issuedCert, PKI_STACK *sk_services, char *url_s)`
Retireve a PKI_STACK of addresses from a PRQP server.
- `char * PKI_get_ca_service (PKI_X509_CERT *caCert, char *srv, char *url_s)`
Retireve the first configured URL for a PKI service from a PRQP server.
- `PKI_STACK * PKI_get_ca_service_sk (PKI_X509_CERT *caCert, char *srv, char *url_s)`
Retireve a stack of configured URL for a PKI service from a PRQP server.

1.68.1 Function Documentation

1.68.1.1 `PKI_X509_PRQP_RESP* PKI_DISCOVER_get_resp (PKI_X509_PRQP_REQ * p, char * url_s)`

Retireve a PRQP response from the passed url or from one of the configured RQAs in /etc/pki.conf.

1.68.1.2 `PKI_X509_PRQP_RESP* PKI_DISCOVER_get_resp_url (PKI_X509_PRQP_REQ * p, URL * url)`

Retrieve a PRQP Response from a server.

The function returns a `PKI_X509_PRQP_RESP` if succesful or `NULL` if an error occurs when contacting the PRQP server.

1.68.1.3 `PKI_STACK* PKI_get_ca_resources (PKI_X509_CERT * caCert, PKI_X509_CERT * ca-IssuerCert, PKI_X509_CERT * issuedCert, PKI_STACK * sk_services, char * url_s)`

Retireve a `PKI_STACK` of addresses from a PRQP server.

Retrieve informations about services provided by a CA from a PRQP server. The function returns a stack of strings containing URLs of the requested services. If no URL (`char *`) is passed, then the library will search for the default config file `/etc/pki.conf` for the configured Resource Query Authority (PRQP Server).

1.68.1.4 `char* PKI_get_ca_service (PKI_X509_CERT * caCert, char * srv, char * url_s)`

Retireve the first configured URL for a PKI service from a PRQP server.

Retrieve informations about a specific service provided by a CA. The function returns a string containing the URL of the requested service (if available).

If no URL (`char *`) is passed, then the library will search for the default config file `/etc/pki.conf` for the configured Resource Query Authority (PRQP Server).

1.68.1.5 PKI_STACK* PKI_get_ca_service_sk (PKI_X509_CERT * caCert, char * srv, char * url_s)

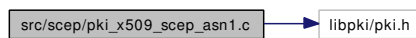
Retrieve a stack of configured URL for a PKI service from a PRQP server.

Retrieve information about a specific service provided by a CA. The function returns a PKI_STACK containing the URLs of the requested service (if available).

If no URL (char *) is passed, then the library will search for the default config file /etc/pki.conf for the configured Resource Query Authority (PRQP Server).

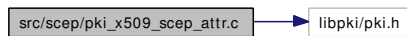
1.69 src/scep/pki_x509_scep_asn1.c File Reference

Include dependency graph for pki_x509_scep_asn1.c:



1.70 src/scep/pki_x509_scep_attr.c File Reference

Include dependency graph for pki_x509_scep_attr.c:



Defines

- #define [SCEP_CONF_LIST_SIZE](#) 8

Functions

- PKI_ID [PKI_X509_SCEP_ATTRIBUTE_get_nid](#) (SCEP_ATTRIBUTE_TYPE num)
Returns the PKI_ID of the specified SCEP_ATTRIBUTE_TYPE.
- SCEP_ATTRIBUTE_TYPE [PKI_X509_SCEP_ATTRIBUTE_get_txt](#) (char *txt)
Returns the SCEP_ATTRIBUTE_TYPE from the attribute name.
- void [PKI_X509_SCEP_init](#) (void)
- PKI_MEM * [PKI_X509_SCEP_MSG_get_attr_value](#) (PKI_X509_SCEP_MSG *msg, SCEP_ATTRIBUTE_TYPE type)
Returns the value of the specified attribute in a PKI_MEM.
- int [PKI_X509_SCEP_MSG_get_attr_value_int](#) (PKI_X509_SCEP_MSG *msg, SCEP_ATTRIBUTE_TYPE type)
- SCEP_FAILURE [PKI_X509_SCEP_MSG_get_failinfo](#) (PKI_X509_PKCS7 *msg)
Returns the failInfo attribute from a SCEP message.
- PKI_OID * [PKI_X509_SCEP_MSG_get_oid](#) (SCEP_ATTRIBUTE_TYPE scep_attribute)
Returns the PKI_OID for the specified SCEP attribute.

- int [PKI_X509_SCEP_MSG_get_proxy](#) (PKI_X509_PKCS7 *msg)
Returns the proxyAuthenticator attribute from a SCEP message.
- PKI_MEM * [PKI_X509_SCEP_MSG_get_recipient_nonce](#) (PKI_X509_PKCS7 *msg)
Returns the recipientNonce attribute from a SCEP message.
- PKI_MEM * [PKI_X509_SCEP_MSG_get_sender_nonce](#) (PKI_X509_PKCS7 *msg)
Returns the senderNonce attribute from a SCEP message.
- SCEP_STATUS [PKI_X509_SCEP_MSG_get_status](#) (PKI_X509_PKCS7 *msg)
Returns the pkiStatus attribute from a SCEP message.
- char * [PKI_X509_SCEP_MSG_get_trans_id](#) (PKI_X509_SCEP_MSG *msg)
- SCEP_MESSAGE_TYPE [PKI_X509_SCEP_MSG_get_type](#) (PKI_X509_PKCS7 *msg)
Returns the messageType attribute from a SCEP message.
- PKI_MEM * [PKI_X509_SCEP_MSG_new_trans_id](#) (PKI_X509_KEYPAIR *key)
Generates a new PKI_MEM suitable for the transId of a SCEP message.
- int [PKI_X509_SCEP_MSG_set_attribute](#) (PKI_X509_PKCS7 *msg, SCEP_ATTRIBUTE_TYPE type, unsigned char *data, size_t size)
Sets the message type attribute in a SCEP message (signed P7).
- int [PKI_X509_SCEP_MSG_set_attribute_by_name](#) (PKI_X509_PKCS7 *msg, char *name, unsigned char *data, size_t size)
Adds an attribute (identified by its name) to a SCEP message.
- int [PKI_X509_SCEP_MSG_set_attribute_int](#) (PKI_X509_PKCS7 *msg, PKI_ID id, int val)
Adds the specified attribute (int) as a string.
- int [PKI_X509_SCEP_MSG_set_failinfo](#) (PKI_X509_PKCS7 *msg, int fail)
Sets the failInfo attribute in a SCEP message.
- int [PKI_X509_SCEP_MSG_set_proxy](#) (PKI_X509_PKCS7 *msg, int auth)
Sets the proxyAuthenticator attribute from a SCEP message.
- int [PKI_X509_SCEP_MSG_set_recipient_nonce](#) (PKI_X509_PKCS7 *msg, PKI_MEM *mem)
Sets the recipientNonce attribute from a SCEP message.
- int [PKI_X509_SCEP_MSG_set_sender_nonce](#) (PKI_X509_PKCS7 *msg, PKI_MEM *mem)
Sets the senderNonce attribute in a SCEP message.
- int [PKI_X509_SCEP_MSG_set_status](#) (PKI_X509_PKCS7 *msg, SCEP_STATUS status)
Sets the pkiStatus attribute in a SCEP message.
- int [PKI_X509_SCEP_MSG_set_trans_id](#) (PKI_X509_PKCS7 *msg, PKI_MEM *mem)
Sets the transactionId attribute in a SCEP message.
- int [PKI_X509_SCEP_MSG_set_type](#) (PKI_X509_PKCS7 *msg, SCEP_MESSAGE_TYPE type)
Sets the messageType attribute in a SCEP message.

Variables

- SCEP_CONF_ATTRIBUTE [SCEP_ATTRIBUTE_list](#) [SCEP_CONF_LIST_SIZE]

1.70.1 Define Documentation

1.70.1.1 #define SCEP_CONF_LIST_SIZE 8

1.70.2 Function Documentation

1.70.2.1 PKI_ID PKI_X509_SCEP_ATTRIBUTE_get_nid (SCEP_ATTRIBUTE_TYPE *num*)

Returns the PKI_ID of the specified SCEP_ATTRIBUTE_TYPE.

1.70.2.2 SCEP_ATTRIBUTE_TYPE PKI_X509_SCEP_ATTRIBUTE_get_txt (char * *txt*)

Returns the SCEP_ATTRIBUTE_TYPE from the attribute name.

1.70.2.3 void PKI_X509_SCEP_init (void)

1.70.2.4 PKI_MEM* PKI_X509_SCEP_MSG_get_attr_value (PKI_X509_SCEP_MSG * *msg*, SCEP_ATTRIBUTE_TYPE *type*)

Returns the value of the specified attribute in a PKI_MEM.

1.70.2.5 int PKI_X509_SCEP_MSG_get_attr_value_int (PKI_X509_SCEP_MSG * *msg*, SCEP_ATTRIBUTE_TYPE *type*)

1.70.2.6 SCEP_FAILURE PKI_X509_SCEP_MSG_get_failinfo (PKI_X509_PKCS7 * *msg*)

Returns the failInfo attribute from a SCEP message.

1.70.2.7 PKI_OID* PKI_X509_SCEP_MSG_get_oid (SCEP_ATTRIBUTE_TYPE *scep_attribute*)

Returns the PKI_OID for the specified SCEP attribute.

1.70.2.8 int PKI_X509_SCEP_MSG_get_proxy (PKI_X509_PKCS7 * *msg*)

Returns the proxyAuthenticator attribute from a SCEP message.

1.70.2.9 PKI_MEM* PKI_X509_SCEP_MSG_get_recipient_nonce (PKI_X509_PKCS7 * *msg*)

Returns the recipientNonce attribute from a SCEP message.

1.70.2.10 PKI_MEM* PKI_X509_SCEP_MSG_get_sender_nonce (PKI_X509_PKCS7 * *msg*)

Returns the senderNonce attribute from a SCEP message.

1.70.2.11 `SCEP_STATUS` `PKI_X509_SCEP_MSG_get_status (PKI_X509_PKCS7 * msg)`

Returns the pkiStatus attribute from a SCEP message.

1.70.2.12 `char*` `PKI_X509_SCEP_MSG_get_trans_id (PKI_X509_SCEP_MSG * msg)`**1.70.2.13** `SCEP_MESSAGE_TYPE` `PKI_X509_SCEP_MSG_get_type (PKI_X509_PKCS7 * msg)`

Returns the messageType attribute from a SCEP message.

1.70.2.14 `PKI_MEM*` `PKI_X509_SCEP_MSG_new_trans_id (PKI_X509_KEYPAIR * key)`

Generates a new PKI_MEM suitable for the transId of a SCEP message.

1.70.2.15 `int` `PKI_X509_SCEP_MSG_set_attribute (PKI_X509_PKCS7 * msg, SCEP_ATTRIBUTE_TYPE type, unsigned char * data, size_t size)`

Sets the message type attribute in a SCEP message (signed P7).

1.70.2.16 `int` `PKI_X509_SCEP_MSG_set_attribute_by_name (PKI_X509_PKCS7 * msg, char * name, unsigned char * data, size_t size)`

Adds an attribute (identified by its name) to a SCEP message.

1.70.2.17 `int` `PKI_X509_SCEP_MSG_set_attribute_int (PKI_X509_PKCS7 * msg, PKI_ID id, int val)`

Adds the specified attribute (int) as a string.

1.70.2.18 `int` `PKI_X509_SCEP_MSG_set_failinfo (PKI_X509_PKCS7 * msg, int fail)`

Sets the failInfo attribute in a SCEP message.

1.70.2.19 `int` `PKI_X509_SCEP_MSG_set_proxy (PKI_X509_PKCS7 * msg, int auth)`

Sets the proxyAuthenticator attribute from a SCEP message.

1.70.2.20 `int` `PKI_X509_SCEP_MSG_set_recipient_nonce (PKI_X509_PKCS7 * msg, PKI_MEM * mem)`

Sets the recipientNonce attribute from a SCEP message.

1.70.2.21 `int` `PKI_X509_SCEP_MSG_set_sender_nonce (PKI_X509_PKCS7 * msg, PKI_MEM * mem)`

Sets the senderNonce attribute in a SCEP message.

1.70.2.22 `int` `PKI_X509_SCEP_MSG_set_status (PKI_X509_PKCS7 * msg, SCEP_STATUS status)`

Sets the pkiStatus attribute in a SCEP message.

1.70.2.23 int PKI_X509_SCEP_MSG_set_trans_id (PKI_X509_PKCS7 * msg, PKI_MEM * mem)

Sets the transactionId attribute in a SCEP message.

1.70.2.24 int PKI_X509_SCEP_MSG_set_type (PKI_X509_PKCS7 * msg, SCEP_MESSAGE_TYPE type)

Sets the messageType attribute in a SCEP message.

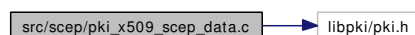
1.70.3 Variable Documentation**1.70.3.1 SCEP_CONF_ATTRIBUTE SCEP_ATTRIBUTE_list[SCEP_CONF_LIST_SIZE]**

Initial value:

```
{
    { SCEP_ATTRIBUTE_MESSAGE_TYPE, "2.16.840.1.113733.1.9.2",
      "scepMessageType", "SCEP Message Type", -1 },
    { SCEP_ATTRIBUTE_PKI_STATUS, "2.16.840.1.113733.1.9.3",
      "pkiStatus", "Status", -1 },
    { SCEP_ATTRIBUTE_FAIL_INFO, "2.16.840.1.113733.1.9.4",
      "failInfo", "Failure Info", -1 },
    { SCEP_ATTRIBUTE_SENDER_NONCE, "2.16.840.1.113733.1.9.5",
      "senderNonce", "Sender Nonce", -1 },
    { SCEP_ATTRIBUTE_RECIPIENT_NONCE, "2.16.840.1.113733.1.9.6",
      "recipientNonce", "Recipient Nonce", -1 },
    { SCEP_ATTRIBUTE_TRANS_ID, "2.16.840.1.113733.1.9.7",
      "transId", "Transaction Identifier", -1 },
    { SCEP_ATTRIBUTE_EXTENSION_REQ, "2.16.840.1.113733.1.9.8",
      "extensionReq", "Extension Request", -1 },
    { SCEP_ATTRIBUTE_PROXY_AUTH, "1.3.6.1.4.1.4263.5.5",
      "proxyAuth", "Proxy Authenticator", -1 },
}
```

1.71 src/scep/pki_x509_scep_data.c File Reference

Include dependency graph for pki_x509_scep_data.c:

**Functions**

- int [PKI_X509_SCEP_DATA_add_recipient](#) (PKI_X509_SCEP_DATA *data, PKI_X509_CERT *recipient)

Adds a recipient to a SCEP_DATA.

- void [PKI_X509_SCEP_DATA_free](#) (PKI_X509_SCEP_DATA *data)

Frees the memory associated with a PKI_X509_SCEP_DATA.

- PKI_X509_SCEP_DATA * [PKI_X509_SCEP_DATA_new](#) (void)

Generates a new SCEP_DATA.

- int [PKI_X509_SCEP_DATA_set_ias](#) (PKI_X509_SCEP_DATA *scep_data, SCEP_ISSUER_AND_SUBJECT *ias)
Sets the content of the SCEP_DATA via a SCEP_ISSUER_AND_SUBJECT.
- int [PKI_X509_SCEP_DATA_set_raw_data](#) (PKI_X509_SCEP_DATA *data, unsigned char *raw_val, ssize_t size)
Sets the content of the SCEP_DATA (raw data).
- int [PKI_X509_SCEP_DATA_set_recipients](#) (PKI_X509_SCEP_DATA *data, PKI_X509_CERT_STACK *sk)
Adds a stack of recipients for a SCEP_DATA.
- int [PKI_X509_SCEP_DATA_set_x509_obj](#) (PKI_X509_SCEP_DATA *data, PKI_X509 *obj)
Sets the content of the SCEP_DATA via a PKI_X509 object.

1.71.1 Function Documentation

1.71.1.1 int PKI_X509_SCEP_DATA_add_recipient (PKI_X509_SCEP_DATA * data, PKI_X509_CERT * recipient)

Adds a recipient to a SCEP_DATA.

1.71.1.2 void PKI_X509_SCEP_DATA_free (PKI_X509_SCEP_DATA * data)

Frees the memory associated with a PKI_X509_SCEP_DATA.

1.71.1.3 PKI_X509_SCEP_DATA* PKI_X509_SCEP_DATA_new (void)

Generates a new SCEP_DATA.

1.71.1.4 int PKI_X509_SCEP_DATA_set_ias (PKI_X509_SCEP_DATA * scep_data, SCEP_ISSUER_AND_SUBJECT * ias)

Sets the content of the SCEP_DATA via a SCEP_ISSUER_AND_SUBJECT.

1.71.1.5 int PKI_X509_SCEP_DATA_set_raw_data (PKI_X509_SCEP_DATA * data, unsigned char * raw_val, ssize_t size)

Sets the content of the SCEP_DATA (raw data).

1.71.1.6 int PKI_X509_SCEP_DATA_set_recipients (PKI_X509_SCEP_DATA * data, PKI_X509_CERT_STACK * sk)

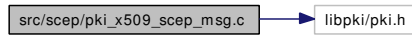
Adds a stack of recipients for a SCEP_DATA.

1.71.1.7 int PKI_X509_SCEP_DATA_set_x509_obj (PKI_X509_SCEP_DATA * data, PKI_X509 * obj)

Sets the content of the SCEP_DATA via a PKI_X509 object.

1.72 src/scep/pki_x509_scep_msg.c File Reference

Include dependency graph for pki_x509_scep_msg.c:



Functions

- [int `PKI_X509_SCEP_MSG_add_signer`](#) (`PKI_X509_SCEP_MSG *msg`, `PKI_X509_CERT *signer`, `PKI_X509_KEYPAIR *key`, `PKI_DIGEST_ALG *md`)
Add a signer to a SCEP_MSG.
- [int `PKI_X509_SCEP_MSG_add_signer_tk`](#) (`PKI_X509_SCEP_MSG *msg`, `PKI_TOKEN *tk`, `PKI_DIGEST_ALG *md`)
Add a signer to a SCEP_MSG by using the passed token data.
- `PKI_MEM *` [PKI_X509_SCEP_MSG_decode](#) (`PKI_X509_SCEP_MSG *msg`, `PKI_X509_KEYPAIR *key`, `PKI_X509_CERT *x`)
Retrieves the decoded data (raw) from a SCEP_MSG.
- [int `PKI_X509_SCEP_MSG_encode`](#) (`PKI_X509_SCEP_MSG *msg`, `PKI_X509_SCEP_DATA *data`)
Encodes a SCEP_MSG to a PKCS7 structure.
- `void` [PKI_X509_SCEP_MSG_free](#) (`PKI_X509_SCEP_MSG *msg`)
Frees the memory associated with a PKI_X509_SCEP_MSG.
- `PKI_X509_SCEP_MSG *` [PKI_X509_SCEP_MSG_new](#) (`SCEP_MESSAGE_TYPE type`)
Generates a new PKI_X509_SCEP message.
- `PKI_X509_SCEP_MSG *` [PKI_X509_SCEP_MSG_new_certreq](#) (`PKI_X509_KEYPAIR *key`, `PKI_X509_REQ *req`, `PKI_X509_CERT *signer`, `PKI_X509_CERT_STACK *recipients`)
Generates a PKCSReq message from a keypair and a subject.

1.72.1 Function Documentation

1.72.1.1 `int PKI_X509_SCEP_MSG_add_signer (PKI_X509_SCEP_MSG * msg, PKI_X509_CERT * signer, PKI_X509_KEYPAIR * key, PKI_DIGEST_ALG * md)`

Add a signer to a SCEP_MSG.

1.72.1.2 `int PKI_X509_SCEP_MSG_add_signer_tk (PKI_X509_SCEP_MSG * msg, PKI_TOKEN * tk, PKI_DIGEST_ALG * md)`

Add a signer to a SCEP_MSG by using the passed token data.

1.72.1.3 `PKI_MEM* PKI_X509_SCEP_MSG_decode (PKI_X509_SCEP_MSG * msg, PKI_X509_KEYPAIR * key, PKI_X509_CERT * x)`

Retrieves the decoded data (raw) from a SCEP_MSG.

1.72.1.4 int PKI_X509_SCEP_MSG_encode (PKI_X509_SCEP_MSG * msg, PKI_X509_SCEP_DATA * data)

Encodes a SCEP_MSG to a PKCS7 structure.

1.72.1.5 void PKI_X509_SCEP_MSG_free (PKI_X509_SCEP_MSG * msg)

Frees the memory associated with a PKI_X509_SCEP_MSG.

1.72.1.6 PKI_X509_SCEP_MSG* PKI_X509_SCEP_MSG_new (SCEP_MESSAGE_TYPE type)

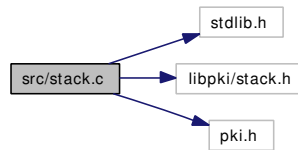
Generates a new PKI_X509_SCEP message.

1.72.1.7 PKI_X509_SCEP_MSG* PKI_X509_SCEP_MSG_new_certreq (PKI_X509_KEYPAIR * key, PKI_X509_REQ * req, PKI_X509_CERT * signer, PKI_X509_CERT_STACK * recipients)

Generates a PKCSReq message from a keypair and a subject.

1.73 src/stack.c File Reference

Include dependency graph for stack.c:



Functions

- void * [PKI_STACK_del_num](#) (PKI_STACK *st, int num)
Pops a specific data element from a PKI_STACK.
- int [PKI_STACK_elements](#) (PKI_STACK *st)
Returns the number of elements in a PKI_STACK.
- int [PKI_STACK_free](#) (PKI_STACK *st)
PKI_STACK free all function.
- int [PKI_STACK_free_all](#) (PKI_STACK *st)
Frees memory associated with a PKI_STACK.
- void * [PKI_STACK_get_num](#) (PKI_STACK *st, int num)
Returns data stored in the n-th element of the PKI_STACK.
- int [PKI_STACK_ins_num](#) (PKI_STACK *st, int num, void *obj)
Inserts a new element in a PKI_STACK at a specific position.
- PKI_STACK * [PKI_STACK_new](#) (void(*free)())

PKI_STACK initialization function.

- `PKI_STACK * PKI_STACK_new_null` (void)
- `PKI_STACK * PKI_STACK_new_type` (int type)
- `void * PKI_STACK_pop` (PKI_STACK *st)

Pops the last element in PKI_STACK.

- `int PKI_STACK_pop_free` (PKI_STACK *st)

Pops the last element in PKI_STACK and frees the object data (when the free function pointer is provided).

- `int PKI_STACK_push` (PKI_STACK *st, void *obj)

Adds a new element to a PKI_STACK.

1.73.1 Function Documentation

1.73.1.1 `void* PKI_STACK_del_num` (PKI_STACK * st, int num)

Pops a specific data element from a PKI_STACK.

This function detatches a specific element of a PKI_STACK and returns the pointer to the associated data. The data is now no more linked in the PKI_STACK and memory management is up to the calling application (i.e., the calling application can now free the memory by calling the appropriate function depending on the type of data structure).

The function returns the pointer to the data. In case of error, it returns NULL.

1.73.1.2 `int PKI_STACK_elements` (PKI_STACK * st)

Returns the number of elements in a PKI_STACK.

This function returns the number of elements stored in a PKI_STACK.

1.73.1.3 `int PKI_STACK_free` (PKI_STACK * st)

PKI_STACK free all function.

This function frees the memory used by a PKI_STACK structure. If the structure is not empty, the pointers to every node are freed, but the pointers to the actualy DATA are not freed. If you want to completely clean up memory, use the `PKI_STACK_free_all()`.

You can also use the `PKI_STACK_pop()` function and free the elements by using the appropriate function (e.g., `PKI_X509_CERT_free()` if it is a stack of certificates).

1.73.1.4 `int PKI_STACK_free_all` (PKI_STACK * st)

Frees memory associated with a PKI_STACK.

This function frees the memory used by a PKI_STACK structure. If the structure is not empty, the pointers to every node are freed, If the type of data within the STACK is not known to the stack itself, it is suggested that you use the `PKI_STACK_pop()` function and free the elements by using the appropriate function.

1.73.1.5 `void* PKI_STACK_get_num` (PKI_STACK * st, int num)

Returns data stored in the n-th element of the PKI_STACK.

Use this function to retrieve data from a specific element of the `PKI_STACK`. The returned pointer points to the data stored in the `PKI_STACK` node, therefore it is not advisable to free the returned memory. You should use the `PKI_STACK_del_num()` function to detach the data from the `PKI_STACK`. The function returns the pointer to the requested data, in case of error it returns `NULL`.

1.73.1.6 `int PKI_STACK_ins_num (PKI_STACK * st, int num, void * obj)`

Inserts a new element in a `PKI_STACK` at a specific position.

Use this function to insert an element into a `PKI_STACK` at a specific position.

If successful the function returns `PKI_STACK_OK`, otherwise it returns `PKI_STACK_ERR` in case of error.

1.73.1.7 `PKI_STACK* PKI_STACK_new (void(*)() free)`

`PKI_STACK` initialization function.

This function allocates the memory for a `PKI_STACK` structure. It also initializes the internal fields. It returns the pointer to the allocated data structure. In case of error the returned value is `NULL`.

1.73.1.8 `PKI_STACK* PKI_STACK_new_null (void)`

1.73.1.9 `PKI_STACK* PKI_STACK_new_type (int type)`

1.73.1.10 `void* PKI_STACK_pop (PKI_STACK * st)`

Pops the last element in `PKI_STACK`.

This function returns the data pointed by the last element of a `PKI_STACK` and frees the internal memory. The calling program will have to free the memory related to the returned pointer.

1.73.1.11 `int PKI_STACK_pop_free (PKI_STACK * st)`

Pops the last element in `PKI_STACK` and frees the object data (when the free function pointer is provided).

This function returns the data pointed by the last element of a `PKI_STACK` and frees the internal memory. If the `PKI_STACK->free` function pointer is provided when created (`PKI_STACK_new`) the associated object is automatically freed, otherwise the calling program will have to free the memory related to the returned pointer.

1.73.1.12 `int PKI_STACK_push (PKI_STACK * st, void * obj)`

Adds a new element to a `PKI_STACK`.

It adds a general pointer to an already initialized `PKI_STACK` structure. In case of success it returns the number of elements in the stack after the new insertion. Otherwise it returns `PKI_STACK_ERR`.

1.74 src/support.c File Reference

Include dependency graph for support.c:



Functions

- char * [get_env_string](#) (const char *str)
- char * [PKI_get_env](#) (char *name)
Returns the value of the ENV variable 'name'.
- int [PKI_set_env](#) (char *name, char *value)
Set the ENV variable 'name' with the value 'value'.
- int [strcmp_nocase](#) (char *st1, char *st2)
- int [strncmp_nocase](#) (char *st1, char *st2, int n)
- char * [strstr_nocase](#) (char *buf, char *string)

1.74.1 Function Documentation

1.74.1.1 char* get_env_string (const char * str)

1.74.1.2 char* PKI_get_env (char * name)

Returns the value of the ENV variable 'name'.

1.74.1.3 int PKI_set_env (char * name, char * value)

Set the ENV variable 'name' with the value 'value'.

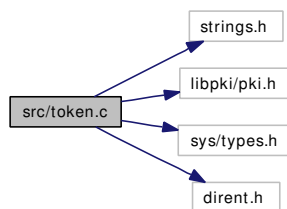
1.74.1.4 int strcmp_nocase (char * st1, char * st2)

1.74.1.5 int strncmp_nocase (char * st1, char * st2, int n)

1.74.1.6 char* strstr_nocase (char * buf, char * string)

1.75 src/token.c File Reference

Include dependency graph for token.c:



Functions

- int [PKI_TOKEN_add_profile](#) (PKI_TOKEN *tk, PKI_X509_PROFILE *profile)
- int [PKI_TOKEN_check](#) (PKI_TOKEN *tk)
Checks the integrity of a PKI_TOKEN.

- PKI_CRED * [PKI_TOKEN_cred_cb_env](#) (char *env)
- PKI_CRED * [PKI_TOKEN_cred_cb_stdin](#) (char *prompt)
- PKI_CRED * [PKI_TOKEN_cred_get](#) (PKI_TOKEN *tk, char *st)
Returns credentials from the registered callback function.
- int [PKI_TOKEN_cred_set_cb](#) (PKI_TOKEN *tk, PKI_CRED *(*cb)(char *), char *prompt)
Register the callback function for asking password to access Token.
- int [PKI_TOKEN_del_url](#) (PKI_TOKEN *tk, URL *url, int datatype)
- int [PKI_TOKEN_export_cert](#) (PKI_TOKEN *tk, char *url_string, int format)
- int [PKI_TOKEN_export_keypair](#) (PKI_TOKEN *tk, char *url_string, int format)
- int [PKI_TOKEN_export_keypair_url](#) (PKI_TOKEN *tk, URL *url, int format)
Export the TOKEN keypair to an external URI (Wrap).
- int [PKI_TOKEN_export_otherCerts](#) (PKI_TOKEN *tk, char *url_string, int format)
Export the TOKEN otherCerts to an external URI.
- int [PKI_TOKEN_export_p12](#) (PKI_TOKEN *tk, int format, char *url_s, PKI_CRED *cred)
Exports a PKI_TOKEN to a URL as PKCS#12 format.
- int [PKI_TOKEN_export_req](#) (PKI_TOKEN *tk, char *url_string, int format)
Export the TOKEN request from the Token to an external URI.
- int [PKI_TOKEN_export_trustedCerts](#) (PKI_TOKEN *tk, char *url_string, int format)
Export the TOKEN trustedCerts to an external URI.
- int [PKI_TOKEN_free](#) (PKI_TOKEN *tk)
Frees a PKI_TOKEN data structure.
- void [PKI_TOKEN_free_void](#) (void *tk)
- PKI_X509_CERT * [PKI_TOKEN_get_cacert](#) (PKI_TOKEN *tk)
Returns the CA certificate of a PKI_TOKEN.
- PKI_X509_CERT * [PKI_TOKEN_get_cert](#) (PKI_TOKEN *tk)
Returns the certificate of a PKI_TOKEN.
- char * [PKI_TOKEN_get_config_dir](#) (PKI_TOKEN *tk)
Get the configuration directory used for the TOKEN operations.
- PKI_CRED * [PKI_TOKEN_get_cred](#) (PKI_TOKEN *tk)
Returns the credentials of a PKI_TOKEN.
- PKI_X509_CRL_STACK * [PKI_TOKEN_get_crls](#) (PKI_TOKEN *tk)
Returns the stack of CRLs stored in a PKI_TOKEN.
- PKI_X509_KEYPAIR * [PKI_TOKEN_get_keypair](#) (PKI_TOKEN *tk)
Returns the PKI_X509_KEYPAIR of a PKI_TOKEN.
- char * [PKI_TOKEN_get_name](#) (PKI_TOKEN *tk)

Get TOKEN name.

- `PKI_X509_CERT_STACK * PKI_TOKEN_get_otherCerts (PKI_TOKEN *tk)`
Returns the stack of other certificates (chain) of a PKI_TOKEN.
- `PKI_X509_PKCS12 * PKI_TOKEN_get_p12 (PKI_TOKEN *tk, PKI_CRED *cred)`
Returns a PKI_X509_PKCS12 data structure from the PKI_TOKEN.
- `PKI_X509_CERT_STACK * PKI_TOKEN_get_trustedCerts (PKI_TOKEN *tk)`
Returns the stack of trusted certificates (chain) of a PKI_TOKEN.
- `int PKI_TOKEN_import_cert (PKI_TOKEN *tk, PKI_X509_CERT *cert, int type, char *url_s)`
Imports a certificate into the Token.
- `int PKI_TOKEN_import_cert_stack (PKI_TOKEN *tk, PKI_X509_CERT_STACK *sk, int type, char *url_s)`
Imports a stack of certificates into the Token.
- `int PKI_TOKEN_import_keypair (PKI_TOKEN *tk, PKI_X509_KEYPAIR *key, char *url_s)`
Imports a Keypair into the Token.
- `int PKI_TOKEN_init (PKI_TOKEN *tk, char *conf_dir, char *tk_name)`
Initialize Token properties (load OIDs and PROFILES).
- `PKI_X509_CERT * PKI_TOKEN_issue_cert (PKI_TOKEN *tk, char *subject, char *serial, unsigned long validity, PKI_X509_REQ *req, char *profile_s)`
- `PKI_X509_CRL * PKI_TOKEN_issue_crl (PKI_TOKEN *tk, char *serial, unsigned long validity, PKI_X509_CRL_ENTRY_STACK *sk, char *profile_s)`
Generate a new CRL from a stack of revoked entries by using the provided token.
- `PKI_TOKEN * PKI_TOKEN_issue_proxy (PKI_TOKEN *tk, char *subject, char *serial, unsigned long validity, char *profile_s, PKI_TOKEN *px_tk)`
- `int PKI_TOKEN_load_cacert (PKI_TOKEN *tk, char *url_string)`
Get the CA certificate from a URL and assigns it to the PKI_TOKEN.
- `int PKI_TOKEN_load_cert (PKI_TOKEN *tk, char *url_string)`
Get a certificate from a URL and assigns it to the PKI_TOKEN.
- `int PKI_TOKEN_load_config (PKI_TOKEN *tk, char *tk_name)`
Loads a configuration file from the token.d directory.
- `int PKI_TOKEN_load_crls (PKI_TOKEN *tk, char *url_string)`
Get a stack of CRLs from a URL and assigns them to the PKI_TOKEN.
- `int PKI_TOKEN_load_keypair (PKI_TOKEN *tk, char *url_string)`
Get a PKI_X509_KEYPAIR from a URL and assigns it to the PKI_TOKEN.
- `int PKI_TOKEN_load_otherCerts (PKI_TOKEN *tk, char *url_string)`
Get a chain of certificates from a URL and assigns them to the PKI_TOKEN (Not Trust Anchors - i.e., trusted CA certs).

- int [PKI_TOKEN_load_profiles](#) (PKI_TOKEN *tk, char *urlStr)
- int [PKI_TOKEN_load_req](#) (PKI_TOKEN *tk, char *url_string)
Loads a certificate request from a URL and assigns it to the TOKEN.
- int [PKI_TOKEN_load_trustedCerts](#) (PKI_TOKEN *tk, char *url_string)
Get a chain of TRUSTED certificates from a URL and assigns them to the PKI_TOKEN (CA Certificates).
- int [PKI_TOKEN_login](#) (PKI_TOKEN *tk)
Login into the token (triggers keypair loading).
- PKI_TOKEN * [PKI_TOKEN_new](#) (char *config_dir, char *name)
Create a new PKI_TOKEN structure and initialize it.
- int [PKI_TOKEN_new_keypair](#) (PKI_TOKEN *tk, int bits, char *label)
Generates a PKI_X509_KEYPAIR and store it in a PKI_TOKEN.
- int [PKI_TOKEN_new_keypair_url](#) (PKI_TOKEN *tk, int bits, URL *label)
Generates a PKI_X509_KEYPAIR and store it in a PKI_TOKEN.
- PKI_TOKEN * [PKI_TOKEN_new_null](#) (void)
Create a new PKI_TOKEN structure.
- PKI_TOKEN * [PKI_TOKEN_new_p12](#) (char *url, char *config_dir, PKI_CRED *cred)
Returns a PKI_TOKEN object from a url pointing to a PKCS#12 object.
- int [PKI_TOKEN_new_req](#) (PKI_TOKEN *tk, char *subject, char *profile_s)
- PKI_OID * [PKI_TOKEN_OID_new](#) (PKI_TOKEN *tk, char *oid_s)
Returns a pointer to an Object Identifier structure.
- int [PKI_TOKEN_print_info](#) (PKI_TOKEN *tk)
- PKI_X509_PROFILE * [PKI_TOKEN_search_profile](#) (PKI_TOKEN *tk, char *profile_s)
- int [PKI_TOKEN_self_sign](#) (PKI_TOKEN *tk, char *subject, char *serial, unsigned long validity, char *profile_s)
- int [PKI_TOKEN_set_algor](#) (PKI_TOKEN *tk, PKI_ALGOR_ID algId)
Set the TOKEN's scheme algorithm via its PKI_ALGOR_ID.
- int [PKI_TOKEN_set_algor_by_name](#) (PKI_TOKEN *tk, char *alg_name)
Set the TOKEN's scheme algorithm via its name.
- int [PKI_TOKEN_set_cacert](#) (PKI_TOKEN *tk, PKI_X509_CERT *x)
Set the PKI_TOKEN CA certificate.
- int [PKI_TOKEN_set_cert](#) (PKI_TOKEN *tk, PKI_X509_CERT *x)
Set the PKI_TOKEN certificate.
- int [PKI_TOKEN_set_config_dir](#) (PKI_TOKEN *tk, char *dir)
Set the configuration directory to be used for the TOKEN operations.
- int [PKI_TOKEN_set_cred](#) (PKI_TOKEN *tk, PKI_CRED *cred)

Set the credentials to be used when retrieving/using the X509_KEYPAIR.

- int [PKI_TOKEN_set_crls](#) (PKI_TOKEN *tk, PKI_X509_CRL_STACK *stack)
Set the PKI_TOKEN stack of CRLs.
- int [PKI_TOKEN_set_keypair](#) (PKI_TOKEN *tk, PKI_X509_KEYPAIR *pkey)
Set the X509_KEYPAIR to be used in the TOKEN.
- int [PKI_TOKEN_set_otherCerts](#) (PKI_TOKEN *tk, PKI_X509_CERT_STACK *stack)
Set the PKI_TOKEN stack of additional certificates.
- int [PKI_TOKEN_set_req](#) (PKI_TOKEN *tk, PKI_X509_REQ *req)
- int [PKI_TOKEN_set_trustedCerts](#) (PKI_TOKEN *tk, PKI_X509_CERT_STACK *stack)
Set the PKI_TOKEN stack of trusted certificates.
- int [PKI_TOKEN_use_slot](#) (PKI_TOKEN *tk, long num)
Sets the slot of the current token, in PKCS#11 this is equivalent to the login.

1.75.1 Function Documentation

1.75.1.1 int [PKI_TOKEN_add_profile](#) (PKI_TOKEN * tk, PKI_X509_PROFILE * profile)

1.75.1.2 int [PKI_TOKEN_check](#) (PKI_TOKEN * tk)

Checks the integrity of a PKI_TOKEN.

1.75.1.3 PKI_CRED* [PKI_TOKEN_cred_cb_env](#) (char * env)

1.75.1.4 PKI_CRED* [PKI_TOKEN_cred_cb_stdin](#) (char * prompt)

1.75.1.5 PKI_CRED* [PKI_TOKEN_cred_get](#) (PKI_TOKEN * tk, char * st)

Returns credentials from the registered callback function.

1.75.1.6 int [PKI_TOKEN_cred_set_cb](#) (PKI_TOKEN * tk, PKI_CRED (*)(char *) cb, char * prompt)

Register the callback function for asking password to access Token.

The function to be registered should accept only one parameter (prompt) and should return a pointer to an allocated PKI_CRED structure.

1.75.1.7 int [PKI_TOKEN_del_url](#) (PKI_TOKEN * tk, URL * url, int datatype)

1.75.1.8 int [PKI_TOKEN_export_cert](#) (PKI_TOKEN * tk, char * url_string, int format)

1.75.1.9 int [PKI_TOKEN_export_keypair](#) (PKI_TOKEN * tk, char * url_string, int format)

1.75.1.10 int PKI_TOKEN_export_keypair_url (PKI_TOKEN * *tk*, URL * *url*, int *format*)

Export the TOKEN keypair to an external URI (Wrap).

1.75.1.11 int PKI_TOKEN_export_otherCerts (PKI_TOKEN * *tk*, char * *url_string*, int *format*)

Export the TOKEN otherCerts to an external URI.

1.75.1.12 int PKI_TOKEN_export_p12 (PKI_TOKEN * *tk*, int *format*, char * *url_s*, PKI_CRED * *cred*)

Exports a PKI_TOKEN to a URL as PKCS#12 format.

1.75.1.13 int PKI_TOKEN_export_req (PKI_TOKEN * *tk*, char * *url_string*, int *format*)

Export the TOKEN request from the Token to an external URI.

1.75.1.14 int PKI_TOKEN_export_trustedCerts (PKI_TOKEN * *tk*, char * *url_string*, int *format*)

Export the TOKEN trustedCerts to an external URI.

1.75.1.15 int PKI_TOKEN_free (PKI_TOKEN * *tk*)

Frees a PKI_TOKEN data structure.

This function Frees a PKI_TOKEN memory region. In case the PKI_TOKEN has already initialized pointers, the pointed data is freed.

1.75.1.16 void PKI_TOKEN_free_void (void * *tk*)**1.75.1.17 PKI_X509_CERT* PKI_TOKEN_get_cacert (PKI_TOKEN * *tk*)**

Returns the CA certificate of a PKI_TOKEN.

Use this function to get a reference (not a copy) to the CA certificate of the PKI_TOKEN.

The function returns the pointer or NULL in case of error or if no CA certificate has been assigned to the PKI_TOKEN yet.

1.75.1.18 PKI_X509_CERT* PKI_TOKEN_get_cert (PKI_TOKEN * *tk*)

Returns the certificate of a PKI_TOKEN.

Use this function to get a reference (not a copy) to the certificate of the PKI_TOKEN.

The function returns the pointer or NULL in case of error or if no certificate has been assigned to the PKI_TOKEN yet.

1.75.1.19 char* PKI_TOKEN_get_config_dir (PKI_TOKEN * *tk*)

Get the configuration directory used for the TOKEN operations.

1.75.1.20 PKI_CRED* PKI_TOKEN_get_cred (PKI_TOKEN * tk)

Returns the credentials of a PKI_TOKEN.

Use this function to get a reference (not a copy) to the credentials of the PKI_TOKEN.

The function returns the pointer or NULL in case of error of if no credentials has been assigned to the PKI_TOKEN yet.

1.75.1.21 PKI_X509_CRL_STACK* PKI_TOKEN_get_crls (PKI_TOKEN * tk)

Returns the stack of CRLs stored in a PKI_TOKEN.

Use this function to get a reference (not a copy) to the stack of CRLs of the PKI_TOKEN.

The function returns the pointer to the PKI_X509_CRL_STACK or NULL in case of error.

1.75.1.22 PKI_X509_KEYPAIR* PKI_TOKEN_get_keypair (PKI_TOKEN * tk)

Returns the PKI_X509_KEYPAIR of a PKI_TOKEN.

Use this function to get a reference (not a copy) to the PKI_X509_KEYPAIR of the PKI_TOKEN.

The function returns the pointer or NULL in case of error of if no PKI_X509_KEYPAIR has been assigned to the PKI_TOKEN yet.

1.75.1.23 char* PKI_TOKEN_get_name (PKI_TOKEN * tk)

Get TOKEN name.

1.75.1.24 PKI_X509_CERT_STACK* PKI_TOKEN_get_otherCerts (PKI_TOKEN * tk)

Returns the stack of other certificates (chain) of a PKI_TOKEN.

Use this function to get a reference (not a copy) to the stack of certificates of the PKI_TOKEN.

The function returns the pointer to the PKI_X509_CERT_STACK or NULL in case of error.

1.75.1.25 PKI_X509_PKCS12* PKI_TOKEN_get_p12 (PKI_TOKEN * tk, PKI_CRED * cred)

Returns a PKI_X509_PKCS12 data structure from the PKI_TOKEN.

1.75.1.26 PKI_X509_CERT_STACK* PKI_TOKEN_get_trustedCerts (PKI_TOKEN * tk)

Returns the stack of trusted certificates (chain) of a PKI_TOKEN.

Use this function to get a reference (not a copy) to the stack of trusted certificates (Trust Anchors) of the PKI_TOKEN.

The function returns the pointer to the PKI_X509_CERT_STACK or NULL in case of error.

1.75.1.27 int PKI_TOKEN_import_cert (PKI_TOKEN * tk, PKI_X509_CERT * cert, int type, char * url_s)

Imports a certificate into the Token.

1.75.1.28 `int PKI_TOKEN_import_cert_stack (PKI_TOKEN * tk, PKI_X509_CERT_STACK * sk, int type, char * url_s)`

Imports a stack of certificates into the Token.

1.75.1.29 `int PKI_TOKEN_import_keypair (PKI_TOKEN * tk, PKI_X509_KEYPAIR * key, char * url_s)`

Imports a Keypair into the Token.

1.75.1.30 `int PKI_TOKEN_init (PKI_TOKEN * tk, char * conf_dir, char * tk_name)`

Initialize Token properties (load OIDs and PROFILES).

1.75.1.31 `PKI_X509_CERT* PKI_TOKEN_issue_cert (PKI_TOKEN * tk, char * subject, char * serial, unsigned long validity, PKI_X509_REQ * req, char * profile_s)`

1.75.1.32 `PKI_X509_CRL* PKI_TOKEN_issue_crl (PKI_TOKEN * tk, char * serial, unsigned long validity, PKI_X509_CRL_ENTRY_STACK * sk, char * profile_s)`

Generate a new CRL from a stack of revoked entries by using the provided token.

Generates a new signed CRL from a stack of revoked entries. If a profile passed, it is used to set the right extensions in the CRL. To generate a new revoked entry the [PKI_X509_CRL_ENTRY_new\(\)](#) function has to be used.

1.75.1.33 `PKI_TOKEN* PKI_TOKEN_issue_proxy (PKI_TOKEN * tk, char * subject, char * serial, unsigned long validity, char * profile_s, PKI_TOKEN * px_tk)`

1.75.1.34 `int PKI_TOKEN_load_cacert (PKI_TOKEN * tk, char * url_string)`

Get the CA certificate from a URL and assigns it to the PKI_TOKEN.

This function loads the CA certificate from a URL and assigns it to the PKI_TOKEN structure. The certificate data structure is automatically freed when the [PKI_TOKEN_free\(\)](#) function is used.

The function returns PKI_OK in case of success, otherwise it returns PKI_ERR.

1.75.1.35 `int PKI_TOKEN_load_cert (PKI_TOKEN * tk, char * url_string)`

Get a certificate from a URL and assigns it to the PKI_TOKEN.

This function loads a certificate from a URL and assigns it to the PKI_TOKEN structure. The certificate data structure is automatically freed when the [PKI_TOKEN_free\(\)](#) function is used.

The function returns PKI_OK in case of success, otherwise it returns PKI_ERR.

1.75.1.36 `int PKI_TOKEN_load_config (PKI_TOKEN * tk, char * tk_name)`

Loads a configuration file from the token.d directory.

Loads the configuration to be used by the Token. The second parameter is checked against the name of the configuration (not the filename, but the `<pki:name></pki:name>` field in the file), and the matching is loaded. Each file which filename ends with .xml is checked in the token.d/ dir. The name comparison is case insensitive.

The function returns `PKI_OK` in case of success, or `PKI_ERR` in case of an error.

1.75.1.37 int PKI_TOKEN_load_crls (PKI_TOKEN * *tk*, char * *url_string*)

Get a stack of CRLs from a URL and assigns them to the `PKI_TOKEN`.

This function loads a set of CRLs from a URL and assigns it to the `PKI_TOKEN` structure. The memory associated with the stack of CRLs is automatically freed when the [PKI_TOKEN_free\(\)](#) function is used.

The function returns `PKI_OK` in case of success, otherwise it returns `PKI_ERR`.

1.75.1.38 int PKI_TOKEN_load_keypair (PKI_TOKEN * *tk*, char * *url_string*)

Get a `PKI_X509_KEYPAIR` from a URL and assigns it to the `PKI_TOKEN`.

This function loads a keypair from a URL and assigns it to the `PKI_TOKEN` structure. The keypair data structure is automatically freed when the [PKI_TOKEN_free\(\)](#) function is used.

The function returns `PKI_OK` in case of success, otherwise it returns `PKI_ERR`.

1.75.1.39 int PKI_TOKEN_load_otherCerts (PKI_TOKEN * *tk*, char * *url_string*)

Get a chain of certificates from a URL and assigns them to the `PKI_TOKEN` (Not Trust Anchors - i.e., trusted CA certs).

This function loads a set of certificates from a URL and assigns it to the `PKI_TOKEN` structure. The stack of certificates is automatically freed when the [PKI_TOKEN_free\(\)](#) function is used.

The function returns `PKI_OK` in case of success, otherwise it returns `PKI_ERR`.

1.75.1.40 int PKI_TOKEN_load_profiles (PKI_TOKEN * *tk*, char * *urlStr*)

1.75.1.41 int PKI_TOKEN_load_req (PKI_TOKEN * *tk*, char * *url_string*)

Loads a certificate request from a URL and assigns it to the `TOKEN`.

The function returns `PKI_OK` in case of success, otherwise it returns `PKI_ERR`.

1.75.1.42 int PKI_TOKEN_load_trustedCerts (PKI_TOKEN * *tk*, char * *url_string*)

Get a chain of TRUSTED certificates from a URL and assigns them to the `PKI_TOKEN` (CA Certificates).

This function loads a set of certificates from a URL and assigns it to the `PKI_TOKEN` structure. The stack of certificates is automatically freed when the [PKI_TOKEN_free\(\)](#) function is used.

The function returns `PKI_OK` in case of success, otherwise it returns `PKI_ERR`.

1.75.1.43 int PKI_TOKEN_login (PKI_TOKEN * *tk*)

Login into the token (triggers keypair loading).

1.75.1.44 PKI_TOKEN* PKI_TOKEN_new (char * *config_dir*, char * *name*)

Create a new `PKI_TOKEN` structure and initialize it.

Reserves the memory for a new `PKI_TOKEN` data structure. The returned memory is already zeroized. If the first passed argument is null, the default configuration directory is used (`PREFIX/etc`). If the second argument is not `NULL`, the library will try to load the specified config for the token.

It returns the pointer to the memory region or `NULL` in case of error.

1.75.1.45 `int PKI_TOKEN_new_keypair (PKI_TOKEN * tk, int bits, char * label)`

Generates a `PKI_X509_KEYPAIR` and store it in a `PKI_TOKEN`.

This function assumes a new `PKI_TOKEN` data structure is passed together with the `PKI_SCHEME` and the number of bit for the new key. It returns `PKI_OK` in case of success or `PKI_ERR` if an error occurs.

The label (a url string) is needed by some HSM(s).

1.75.1.46 `int PKI_TOKEN_new_keypair_url (PKI_TOKEN * tk, int bits, URL * label)`

Generates a `PKI_X509_KEYPAIR` and store it in a `PKI_TOKEN`.

This function assumes a new `PKI_TOKEN` data structure is passed together with the `PKI_SCHEME` and the number of bit for the new key. It returns `PKI_OK` in case of success or `PKI_ERR` if an error occurs.

The URL is needed by some HSM(s).

1.75.1.47 `PKI_TOKEN* PKI_TOKEN_new_null (void)`

Create a new `PKI_TOKEN` structure.

Reserves the memory for a new `PKI_TOKEN` data structure. The returned memory is already zeroized. No token configuration is performed. If you need to configure the token by using a configuration file, please use the [PKI_TOKEN_new\(\)](#) function.

1.75.1.48 `PKI_TOKEN* PKI_TOKEN_new_p12 (char * url, char * config_dir, PKI_CRED * cred)`

Returns a `PKI_TOKEN` object from a url pointing to a PKCS#12 object.

1.75.1.49 `int PKI_TOKEN_new_req (PKI_TOKEN * tk, char * subject, char * profile_s)`

1.75.1.50 `PKI_OID* PKI_TOKEN_OID_new (PKI_TOKEN * tk, char * oid_s)`

Returns a pointer to an Object Identifier structure.

This function returns a pointer to an Object Identifier if the OID is identified among the ones in the crypto library used or among the configured ones in the `objectIdentifiers.xml` file that is loaded when the [PKI_TOKEN_init\(\)](#) function is used.

1.75.1.51 `int PKI_TOKEN_print_info (PKI_TOKEN * tk)`

1.75.1.52 `PKI_X509_PROFILE* PKI_TOKEN_search_profile (PKI_TOKEN * tk, char * profile_s)`

1.75.1.53 `int PKI_TOKEN_self_sign (PKI_TOKEN * tk, char * subject, char * serial, unsigned long validity, char * profile_s)`

1.75.1.54 int PKI_TOKEN_set_algor (PKI_TOKEN * tk, PKI_ALGOR_ID algId)

Set the TOKEN's scheme algorithm via its PKI_ALGOR_ID.

When generating signatures (e.g., when issuing a new request or certificate) by using the PKI_TOKEN facility, a signature algorithm has to be chosen. The default one is sha1withRSA, but, depending on the capabilities of the system you may be able to use different ones as well. This algorithm is used in combination with the signature scheme (e.g., sha1withRSA or md5withDSA), therefore it must be consistent with it.

Possible algorithms are: -PKI_ALGOR_RSA_MD5 -PKI_ALGOR_RSA_MD2 -PKI_ALGOR_RSA_SHA1 -PKI_ALGOR_DSA_SHA1 -PKI_ALGOR_RSA_SHA224 -PKI_ALGOR_RSA_SHA256 -PKI_ALGOR_RSA_SHA512 -PKI_ALGOR_RSA_RIPMD160 -PKI_ALGOR_ECDSA_SHA1

1.75.1.55 int PKI_TOKEN_set_algor_by_name (PKI_TOKEN * tk, char * alg_name)

Set the TOKEN's scheme algorithm via its name.

When generating signatures (e.g., when issuing a new request or certificate) by using the PKI_TOKEN facility, a signature algorithm has to be chosen. The default one is sha1withRSA, but, depending on the capabilities of the system you may be able to use different ones as well. This algorithm is used in combination with the signature scheme (e.g., sha1withRSA or md5withDSA), therefore it must be consistent with it.

Possible algorithm names are: RSA-MD2 RSA-MD4 RSA-MD5 RSA-SHA1 DSA-SHA1 RSA-SHA224 RSA-SHA256 RSA-SHA512 RSA-RIPMD160

1.75.1.56 int PKI_TOKEN_set_cacert (PKI_TOKEN * tk, PKI_X509_CERT * x)

Set the PKI_TOKEN CA certificate.

Use this function to set the CA certificate of a PKI_TOKEN. The certificate is automatically freed when the [PKI_TOKEN_free\(\)](#) function is used. The function returns PKI_OK in case of success or PKI_ERR otherwise.

1.75.1.57 int PKI_TOKEN_set_cert (PKI_TOKEN * tk, PKI_X509_CERT * x)

Set the PKI_TOKEN certificate.

Use this function to set the certificate of a PKI_TOKEN. The certificate is automatically freed when the [PKI_TOKEN_free\(\)](#) function is used. The function returns PKI_OK in case of success or PKI_ERR otherwise.

1.75.1.58 int PKI_TOKEN_set_config_dir (PKI_TOKEN * tk, char * dir)

Set the configuration directory to be used for the TOKEN operations.

1.75.1.59 int PKI_TOKEN_set_cred (PKI_TOKEN * tk, PKI_CRED * cred)

Set the credentials to be used when retrieving/using the X509_KEYPAIR.

Use this function to set the credentials to be used by the PKI_TOKEN. The credentials are automatically freed when the [PKI_TOKEN_free\(\)](#) function is used. The function returns PKI_OK in case of success or PKI_ERR otherwise.

1.75.1.60 int PKI_TOKEN_set_crls (PKI_TOKEN * tk, PKI_X509_CRL_STACK * stack)

Set the PKI_TOKEN stack of CRLs.

Use this function to set a stack of CRLs to a `PKI_TOKEN`. The stack is automatically freed when the `PKI_TOKEN_free()` function is used. The CRLs will be used for validation purposes when a verify of a certificate is performed (if the token is passed as an argument).

The function returns `PKI_OK` in case of success or `PKI_ERR` otherwise.

1.75.1.61 `int PKI_TOKEN_set_keypair (PKI_TOKEN *tk, PKI_X509_KEYPAIR *pkey)`

Set the `X509_KEYPAIR` to be used in the `TOKEN`.

Use this function to set the credentials to be used by the `PKI_TOKEN`. The credentials are automatically freed when the `PKI_TOKEN_free()` function is used. The function returns `PKI_OK` in case of success or `PKI_ERR` otherwise.

1.75.1.62 `int PKI_TOKEN_set_otherCerts (PKI_TOKEN *tk, PKI_X509_CERT_STACK *stack)`

Set the `PKI_TOKEN` stack of additional certificates.

Use this function to set a stack of certificate to a `PKI_TOKEN`. The stack is automatically freed when the `PKI_TOKEN_free()` function is used. The additional certificates are used for validation purposes when a verify of the `PKI_TOKEN` certificate is performed.

The function returns `PKI_OK` in case of success or `PKI_ERR` otherwise.

1.75.1.63 `int PKI_TOKEN_set_req (PKI_TOKEN *tk, PKI_X509_REQ *req)`

1.75.1.64 `int PKI_TOKEN_set_trustedCerts (PKI_TOKEN *tk, PKI_X509_CERT_STACK *stack)`

Set the `PKI_TOKEN` stack of trusted certificates.

Use this function to set a stack of trusted certs to a `PKI_TOKEN`. The stack is automatically freed when the `PKI_TOKEN_free()` function is used. The additional certificates are used for validation purposes when a verify of the `PKI_TOKEN` certificate is performed.

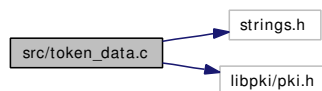
The function returns `PKI_OK` in case of success or `PKI_ERR` otherwise.

1.75.1.65 `int PKI_TOKEN_use_slot (PKI_TOKEN *tk, long num)`

Sets the slot of the current token, in PKCS#11 this is equivalent to the login.

1.76 src/token_data.c File Reference

Include dependency graph for `token_data.c`:



Functions

- `PKI_MEM * PKI_TOKEN_get_cacert_data (PKI_TOKEN *tk, PKI_DATA_FORMAT format)`

- `PKI_MEM * PKI_TOKEN_get_cert_data (PKI_TOKEN *tk, PKI_DATA_FORMAT format)`
- `PKI_MEM * PKI_TOKEN_get_identity_data (PKI_TOKEN *tk, PKI_DATA_FORMAT format)`
- `PKI_MEM * PKI_TOKEN_get_keypair_data (PKI_TOKEN *tk, PKI_DATA_FORMAT format)`

Returns a PKI_MEM that contains a keypair in the specified format (eg., PKI_FORMAT_TXT, PKI_FORMAT_PEM, etc...).

- `PKI_MEM_STACK * PKI_TOKEN_get_otherCerts_data (PKI_TOKEN *tk, PKI_DATA_FORMAT format)`
- `PKI_MEM * PKI_TOKEN_get_privkey_data (PKI_TOKEN *tk, PKI_DATA_FORMAT format)`
- `PKI_MEM * PKI_TOKEN_get_pubkey_data (PKI_TOKEN *tk, PKI_DATA_FORMAT format)`
- `PKI_MEM_STACK * PKI_TOKEN_get_trustedCerts_data (PKI_TOKEN *tk, PKI_DATA_FORMAT format)`

1.76.1 Function Documentation

1.76.1.1 `PKI_MEM* PKI_TOKEN_get_cacert_data (PKI_TOKEN * tk, PKI_DATA_FORMAT format)`

1.76.1.2 `PKI_MEM* PKI_TOKEN_get_cert_data (PKI_TOKEN * tk, PKI_DATA_FORMAT format)`

1.76.1.3 `PKI_MEM* PKI_TOKEN_get_identity_data (PKI_TOKEN * tk, PKI_DATA_FORMAT format)`

1.76.1.4 `PKI_MEM* PKI_TOKEN_get_keypair_data (PKI_TOKEN * tk, PKI_DATA_FORMAT format)`

Returns a PKI_MEM that contains a keypair in the specified format (eg., PKI_FORMAT_TXT, PKI_FORMAT_PEM, etc...).

1.76.1.5 `PKI_MEM_STACK* PKI_TOKEN_get_otherCerts_data (PKI_TOKEN * tk, PKI_DATA_FORMAT format)`

1.76.1.6 `PKI_MEM* PKI_TOKEN_get_privkey_data (PKI_TOKEN * tk, PKI_DATA_FORMAT format)`

1.76.1.7 `PKI_MEM* PKI_TOKEN_get_pubkey_data (PKI_TOKEN * tk, PKI_DATA_FORMAT format)`

1.76.1.8 `PKI_MEM_STACK* PKI_TOKEN_get_trustedCerts_data (PKI_TOKEN * tk, PKI_DATA_FORMAT format)`

1.77 src/token_id.c File Reference

Include dependency graph for token_id.c:



Functions

- PKI_ID_INFO * [PKI_TOKEN_ID_INFO_get](#) (PKI_TOKEN *tk, int num)
- PKI_ID_INFO_STACK * [PKI_TOKEN_ID_INFO_list](#) (PKI_TOKEN *tk)
- int [PKI_TOKEN_ID_num](#) (PKI_TOKEN *tk)
- int [PKI_TOKEN_ID_set](#) (PKI_TOKEN *tk, int id)

1.77.1 Function Documentation

1.77.1.1 PKI_ID_INFO* [PKI_TOKEN_ID_INFO_get](#) (PKI_TOKEN * *tk*, int *num*)

1.77.1.2 PKI_ID_INFO_STACK* [PKI_TOKEN_ID_INFO_list](#) (PKI_TOKEN * *tk*)

1.77.1.3 int [PKI_TOKEN_ID_num](#) (PKI_TOKEN * *tk*)

1.77.1.4 int [PKI_TOKEN_ID_set](#) (PKI_TOKEN * *tk*, int *id*)

Index

- _Accept
 - sock.c, [29](#)
 - _Close
 - sock.c, [29](#)
 - _Connect
 - sock.c, [29](#)
 - _Listen
 - sock.c, [29](#)
 - _Read
 - sock.c, [29](#)
 - _Select
 - sock.c, [29](#)
 - _Shutdown
 - sock.c, [29](#)
 - _Socket
 - sock.c, [29](#)
 - _Write
 - sock.c, [29](#)
- __LIBPKI_ERR__
 - pki_err.c, [88](#)
- __PKI_PRQP_LIB_C__
 - prqp_lib.c, [117](#)
- __pki_http_get_sock_s
 - http_s.c, [24](#)
- __url_get_data_http_s
 - http_s.c, [24](#)
- __url_get_data_http_s_url
 - http_s.c, [24](#)
- _dyn_create_callback
 - pthread_init.c, [81](#)
- _dyn_destroy_callback
 - pthread_init.c, [81](#)
- _dyn_lock_callback
 - pthread_init.c, [81](#)
- BAG_DATATYPE_ALL
 - pki_x509_p12.c, [68](#)
- BAG_DATATYPE_CACERT
 - pki_x509_p12.c, [68](#)
- BAG_DATATYPE_CERT
 - pki_x509_p12.c, [68](#)
- BAG_DATATYPE_KEYPAIR
 - pki_x509_p12.c, [68](#)
- BAG_DATATYPE_OTHERCERTS
 - pki_x509_p12.c, [68](#)
- bag_datatype_st
 - pki_x509_p12.c, [68](#)
- BAG_DATATYPE_UNKNOWN
 - pki_x509_p12.c, [68](#)
- BUFF_MAX_SIZE
 - http_s.c, [24](#)
- ssl.c, [31](#)
- url.c, [34](#)
- CERT_IDENTIFIER_cmp
 - prqp_lib.c, [117](#)
- CERT_REQ_MSG_get
 - cms_cert_req.c, [2](#)
- CERT_REQ_MSG_get_fd
 - cms_cert_req.c, [2](#)
- CERT_REQ_MSG_get_mem
 - cms_cert_req.c, [2](#)
- CERT_REQ_MSG_get_url
 - cms_cert_req.c, [2](#)
- CERT_REQ_MSG_put
 - cms_cert_req.c, [2](#)
- CERT_REQ_MSG_put_fp
 - cms_cert_req.c, [2](#)
- CERT_REQ_MSG_put_mem
 - cms_cert_req.c, [2](#)
- CERT_REQ_MSG_put_url
 - cms_cert_req.c, [2](#)
- cms_cert_req.c
 - CERT_REQ_MSG_get, [2](#)
 - CERT_REQ_MSG_get_fd, [2](#)
 - CERT_REQ_MSG_get_mem, [2](#)
 - CERT_REQ_MSG_get_url, [2](#)
 - CERT_REQ_MSG_put, [2](#)
 - CERT_REQ_MSG_put_fp, [2](#)
 - CERT_REQ_MSG_put_mem, [2](#)
 - CERT_REQ_MSG_put_url, [2](#)
 - d2i_CERT_REQ_MSG_bio, [2](#)
 - i2d_CERT_REQ_MSG_bio, [2](#)
 - PEM_read_bio_CERT_REQ_MSG, [2](#)
 - PEM_write_bio_CERT_REQ_MSG, [3](#)
- cms_simple.c
 - PKI_CMS_REQ, [3](#)
 - PKI_CMS_RESP, [3](#)
 - PKI_MSG_CMS_write, [3](#)
- CRYPTO_LOCK
 - pthread_init.c, [81](#)
- CRYPTO_READ
 - pthread_init.c, [81](#)
- CRYPTO_UNLOCK
 - pthread_init.c, [81](#)
- CRYPTO_WRITE
 - pthread_init.c, [81](#)
- d2i_CERT_REQ_MSG_bio
 - cms_cert_req.c, [2](#)
- d2i_OCSP_REQ_bio
 - pki_ocsp_req.c, [46](#)

- d2i_PKI_OCSP_RESP_bio
 - pki_ocsp_resp.c, [49](#)
- d2i_PRQP_REQ_bio
 - prqp_bio.c, [114](#)
- d2i_PRQP_RESP_bio
 - prqp_bio.c, [114](#)
- extensions.c
 - PKI_X509_EXTENSIONS_cert_add_profile, [4](#)
 - PKI_X509_EXTENSIONS_crl_add_profile, [4](#)
 - PKI_X509_EXTENSIONS_req_add_profile, [4](#)
- get_env_string
 - support.c, [135](#)
- Gethostbyname
 - sock.c, [29](#)
- h_errno
 - sock.c, [30](#)
- http_client.c
 - PKI_X509_PRQP_RESP_get_http, [114](#)
- http_s.c
 - __pki_http_get_sock_s, [24](#)
 - __url_get_data_http_s, [24](#)
 - __url_get_data_http_s_url, [24](#)
 - BUFF_MAX_SIZE, [24](#)
 - PKI_HTTP_free, [24](#)
 - PKI_HTTP_get_header, [24](#)
 - PKI_HTTP_get_header_txt, [24](#)
 - PKI_HTTP_get_sock, [24](#)
 - PKI_HTTP_new, [25](#)
 - PKI_HTTPS_get_sock, [25](#)
 - URL_get_data_http_s, [25](#)
 - URL_get_data_http_s_url, [25](#)
 - URL_post_data_http_s, [25](#)
 - URL_post_data_http_s_url, [25](#)
 - URL_put_data_http_s, [25](#)
- i2d_CERT_REQ_MSG_bio
 - cms_cert_req.c, [2](#)
- i2d_OCSP_REQ_bio
 - pki_ocsp_req.c, [46](#)
- i2d_PKI_OCSP_RESP_bio
 - pki_ocsp_resp.c, [49](#)
- i2d_PRQP_REQ_bio
 - prqp_bio.c, [114](#)
- i2d_PRQP_RESP_bio
 - prqp_bio.c, [114](#)
- inet_close
 - sock.c, [29](#)
- inet_connect
 - sock.c, [29](#)
- LIBPKI_OS_DETAILS
 - pki_init.c, [89](#)
- LIBXML_MIN_VERSION
 - pki_config.c, [83](#)
- LISTENQ
 - sock.c, [29](#)
- lock_cs
 - pthread_init.c, [81](#)
- logXmlMessages
 - pki_config.c, [83](#)
- mysql.c
 - parse_url_dbname, [26](#)
 - parse_url_put_query, [26](#)
 - parse_url_query, [26](#)
 - parse_url_table, [26](#)
 - URL_get_data_mysql, [26](#)
 - URL_get_data_mysql_url, [26](#)
 - URL_put_data_mysql, [26](#)
 - URL_put_data_mysql_url, [26](#)
- NID_proxyCertInfo
 - pki_init.c, [89](#)
 - pki_x509_cert.c, [61](#)
- parse_url_dbname
 - mysql.c, [26](#)
- parse_url_put_query
 - mysql.c, [26](#)
- parse_url_query
 - mysql.c, [26](#)
- parse_url_table
 - mysql.c, [26](#)
- PEM_read_bio_CERT_REQ_MSG
 - cms_cert_req.c, [2](#)
- PEM_read_bio_OCSP_REQ
 - pki_ocsp_req.c, [46](#)
- PEM_read_bio_PKCS12
 - pki_x509_p12.c, [68](#)
- PEM_read_bio_PKI_OCSP_RESP
 - pki_ocsp_resp.c, [49](#)
- PEM_read_bio_PRQP_REQ
 - prqp_bio.c, [114](#)
- PEM_read_bio_PRQP_RESP
 - prqp_bio.c, [114](#)
- PEM_write_bio_CERT_REQ_MSG
 - cms_cert_req.c, [3](#)
- PEM_write_bio_OCSP_REQ
 - pki_ocsp_req.c, [46](#)
- PEM_write_bio_PKCS12
 - pki_x509_p12.c, [68](#)
- PEM_write_bio_PKI_OCSP_RESP
 - pki_ocsp_resp.c, [49](#)
- PEM_write_bio_PRQP_REQ

- prqp_bio.c, [114](#)
- PEM_write_bio_PRQP_RESP
 - prqp_bio.c, [114](#)
- pg.c
 - pg_parse_url_dbname, [27](#)
 - pg_parse_url_put_query, [27](#)
 - pg_parse_url_query, [27](#)
 - pg_parse_url_table, [27](#)
 - URL_get_data_pg, [27](#)
 - URL_get_data_pg_url, [27](#)
 - URL_put_data_pg, [27](#)
 - URL_put_data_pg_url, [27](#)
- pg_parse_url_dbname
 - pg.c, [27](#)
- pg_parse_url_put_query
 - pg.c, [27](#)
- pg_parse_url_query
 - pg.c, [27](#)
- pg_parse_url_table
 - pg.c, [27](#)
- pkcs11.c
 - pkcs11_parse_url_getval, [27](#)
 - URL_get_data_pkcs11, [27](#)
 - URL_get_data_pkcs11_url, [28](#)
- pkcs11_parse_url_getval
 - pkcs11.c, [27](#)
- pki_algor.c
 - PKI_ALGOR_get, [37](#)
 - PKI_ALGOR_get_by_name, [37](#)
 - PKI_ALGOR_get_digest, [37](#)
 - PKI_ALGOR_get_digest_id, [37](#)
 - PKI_ALGOR_get_id, [37](#)
 - PKI_ALGOR_get_parsed, [37](#)
 - PKI_ALGOR_get_scheme, [37](#)
 - PKI_ALGOR_ID_txt, [37](#)
 - PKI_ALGOR_list, [37](#)
 - PKI_ALGOR_LIST_DSA, [38](#)
 - PKI_ALGOR_LIST_ECDSA, [38](#)
 - PKI_ALGOR_LIST_RSA, [38](#)
 - PKI_ALGOR_list_size, [37](#)
 - PKI_DIGEST_ALG_get, [37](#)
 - PKI_DIGEST_ALG_get_by_key, [37](#)
 - PKI_DIGEST_ALG_get_by_name, [38](#)
 - PKI_DIGEST_ALG_get_parsed, [38](#)
 - PKI_DIGEST_ALG_LIST, [39](#)
 - PKI_DIGEST_ALG_list, [38](#)
- PKI_ALGOR_get
 - pki_algor.c, [37](#)
- PKI_ALGOR_get_by_name
 - pki_algor.c, [37](#)
- PKI_ALGOR_get_digest
 - pki_algor.c, [37](#)
- PKI_ALGOR_get_digest_id
 - pki_algor.c, [37](#)
- PKI_ALGOR_get_id
 - pki_algor.c, [37](#)
- PKI_ALGOR_get_parsed
 - pki_algor.c, [37](#)
- PKI_ALGOR_get_scheme
 - pki_algor.c, [37](#)
- PKI_ALGOR_ID_txt
 - pki_algor.c, [37](#)
- PKI_ALGOR_list
 - pki_algor.c, [37](#)
- PKI_ALGOR_LIST_DSA
 - pki_algor.c, [38](#)
- PKI_ALGOR_LIST_ECDSA
 - pki_algor.c, [38](#)
- PKI_ALGOR_LIST_RSA
 - pki_algor.c, [38](#)
- PKI_ALGOR_list_size
 - pki_algor.c, [37](#)
- PKI_clock_gettime
 - pki_threads_vars.c, [105](#)
- PKI_CMS_REQ
 - cms_simple.c, [3](#)
- PKI_CMS_RESP
 - cms_simple.c, [3](#)
- PKI_COND_broadcast
 - pki_threads_vars.c, [105](#)
- PKI_COND_destroy
 - pki_threads_vars.c, [105](#)
- PKI_COND_free
 - pki_threads_vars.c, [105](#)
- PKI_COND_init
 - pki_threads_vars.c, [106](#)
- PKI_COND_new
 - pki_threads_vars.c, [106](#)
- PKI_COND_signal
 - pki_threads_vars.c, [106](#)
- PKI_COND_timedwait
 - pki_threads_vars.c, [106](#)
- PKI_COND_wait
 - pki_threads_vars.c, [106](#)
- pki_config.c
 - LIBXML_MIN_VERSION, [83](#)
 - logXmlMessages, [83](#)
 - PKI_CONFIG_add_node, [83](#)
 - PKI_CONFIG_ELEMENT_add_attribute, [84](#)
 - PKI_CONFIG_ELEMENT_add_child, [84](#)
 - PKI_CONFIG_ELEMENT_add_child_el, [84](#)
 - PKI_CONFIG_ELEMENT_new, [84](#)
 - PKI_CONFIG_find, [84](#)
 - PKI_CONFIG_find_all, [84](#)
 - PKI_CONFIG_free, [84](#)
 - PKI_CONFIG_free_void, [84](#)
 - PKI_CONFIG_get_attribute_value, [84](#)
 - PKI_CONFIG_get_element, [84](#)

- PKI_CONFIG_get_element_child, 84
- PKI_CONFIG_get_element_children, 84
- PKI_CONFIG_get_element_name, 85
- PKI_CONFIG_get_element_next, 85
- PKI_CONFIG_get_element_prev, 85
- PKI_CONFIG_get_element_stack, 85
- PKI_CONFIG_get_element_value, 85
- PKI_CONFIG_get_elements_num, 85
- PKI_CONFIG_get_root, 85
- PKI_CONFIG_get_search_paths, 85
- PKI_CONFIG_get_stack_value, 85
- PKI_CONFIG_get_value, 85
- PKI_CONFIG_load, 85
- PKI_CONFIG_load_all, 85
- PKI_CONFIG_load_dir, 86
- PKI_CONFIG_OID_search, 86
- PKI_DEF_CONF_DIRS_SIZE, 83
- xmlErrorPtr, 83
- PKI_CONFIG_add_node
 - pki_config.c, 83
- PKI_CONFIG_ELEMENT_add_attribute
 - pki_config.c, 84
- PKI_CONFIG_ELEMENT_add_child
 - pki_config.c, 84
- PKI_CONFIG_ELEMENT_add_child_el
 - pki_config.c, 84
- PKI_CONFIG_ELEMENT_new
 - pki_config.c, 84
- PKI_CONFIG_find
 - pki_config.c, 84
- PKI_CONFIG_find_all
 - pki_config.c, 84
- PKI_CONFIG_free
 - pki_config.c, 84
- PKI_CONFIG_free_void
 - pki_config.c, 84
- PKI_CONFIG_get_attribute_value
 - pki_config.c, 84
- PKI_CONFIG_get_element
 - pki_config.c, 84
- PKI_CONFIG_get_element_child
 - pki_config.c, 84
- PKI_CONFIG_get_element_children
 - pki_config.c, 84
- PKI_CONFIG_get_element_name
 - pki_config.c, 85
- PKI_CONFIG_get_element_next
 - pki_config.c, 85
- PKI_CONFIG_get_element_prev
 - pki_config.c, 85
- PKI_CONFIG_get_element_stack
 - pki_config.c, 85
- PKI_CONFIG_get_element_value
 - pki_config.c, 85
- PKI_CONFIG_get_elements_num
 - pki_config.c, 85
- PKI_CONFIG_get_root
 - pki_config.c, 85
- PKI_CONFIG_get_search_paths
 - pki_config.c, 85
- PKI_CONFIG_get_stack_value
 - pki_config.c, 85
- PKI_CONFIG_get_value
 - pki_config.c, 85
- PKI_CONFIG_load
 - pki_config.c, 85
- PKI_CONFIG_load_all
 - pki_config.c, 85
- PKI_CONFIG_load_dir
 - pki_config.c, 86
- PKI_CONFIG_OID_search
 - pki_config.c, 86
- pki_cred.c
 - PKI_CRED_dup, 86
 - PKI_CRED_free, 86
 - PKI_CRED_get_ssl, 87
 - PKI_CRED_new, 87
 - PKI_CRED_new_null, 87
 - PKI_CRED_set_ssl, 87
- PKI_CRED_dup
 - pki_cred.c, 86
- PKI_CRED_free
 - pki_cred.c, 86
- PKI_CRED_get_ssl
 - pki_cred.c, 87
- PKI_CRED_new
 - pki_cred.c, 87
- PKI_CRED_new_null
 - pki_cred.c, 87
- PKI_CRED_set_ssl
 - pki_cred.c, 87
- PKI_DEF_CONF_DIRS_SIZE
 - pki_config.c, 83
- pki_digest.c
 - PKI_DIGEST_free, 40
 - PKI_DIGEST_get_parsed, 40
 - PKI_DIGEST_get_size, 40
 - PKI_DIGEST_MEM_new, 40
 - PKI_DIGEST_MEM_new_by_name, 40
 - PKI_DIGEST_new, 40
 - PKI_DIGEST_new_by_name, 40
 - PKI_DIGEST_URL_new, 40
 - PKI_DIGEST_URL_new_by_name, 40
- PKI_DIGEST_ALG_get
 - pki_algor.c, 37
- PKI_DIGEST_ALG_get_by_key
 - pki_algor.c, 37
- PKI_DIGEST_ALG_get_by_name

- pki_algor.c, 38
- PKI_DIGEST_ALG_get_parsed
 - pki_algor.c, 38
- PKI_DIGEST_ALG_LIST
 - pki_algor.c, 39
- PKI_DIGEST_ALG_list
 - pki_algor.c, 38
- PKI_DIGEST_free
 - pki_digest.c, 40
- PKI_DIGEST_get_parsed
 - pki_digest.c, 40
- PKI_DIGEST_get_size
 - pki_digest.c, 40
- PKI_DIGEST_MEM_new
 - pki_digest.c, 40
- PKI_DIGEST_MEM_new_by_name
 - pki_digest.c, 40
- PKI_DIGEST_new
 - pki_digest.c, 40
- PKI_DIGEST_new_by_name
 - pki_digest.c, 40
- PKI_DIGEST_URL_new
 - pki_digest.c, 40
- PKI_DIGEST_URL_new_by_name
 - pki_digest.c, 40
- PKI_DISCOVER_get_resp
 - prqp_srv.c, 124
- PKI_DISCOVER_get_resp_url
 - prqp_srv.c, 124
- pki_err.c
 - __LIBPKI_ERR__, 88
 - pki_err_stack, 88
 - PKI_get_err, 88
 - PKI_get_strerror, 88
 - PKI_set_err, 88
- pki_err_stack
 - pki_err.c, 88
- PKI_Free
 - pki_mem.c, 92
- PKI_get_all_tokens
 - pki_init.c, 89
- PKI_get_all_tokens_dir
 - pki_init.c, 89
- PKI_get_ca_resources
 - prqp_srv.c, 124
- PKI_get_ca_service
 - prqp_srv.c, 124
- PKI_get_ca_service_sk
 - prqp_srv.c, 124
- PKI_get_env
 - support.c, 135
- PKI_get_err
 - pki_err.c, 88
- PKI_get_init_status
 - pki_init.c, 89
- PKI_get_strerror
 - pki_err.c, 88
- PKI_get_value
 - pki_x509_io.c, 15
- PKI_HTTP_free
 - http_s.c, 24
- PKI_HTTP_get_header
 - http_s.c, 24
- PKI_HTTP_get_header_txt
 - http_s.c, 24
- PKI_HTTP_get_sock
 - http_s.c, 24
- PKI_HTTP_new
 - http_s.c, 25
- PKI_HTTPS_get_sock
 - http_s.c, 25
- pki_id.c
 - PKI_ID_get, 41
 - PKI_ID_get_by_name, 41
 - PKI_ID_get_txt, 41
- PKI_ID_get
 - pki_id.c, 41
- PKI_ID_get_by_name
 - pki_id.c, 41
- PKI_ID_get_txt
 - pki_id.c, 41
- pki_init.c
 - LIBPKI_OS_DETAILS, 89
 - NID_proxyCertInfo, 89
 - PKI_get_all_tokens, 89
 - PKI_get_all_tokens_dir, 89
 - PKI_get_init_status, 89
 - PKI_init_all, 89
 - PKI_list_all_id, 89
 - PKI_list_all_tokens, 89
 - PKI_list_all_tokens_dir, 89
- PKI_init_all
 - pki_init.c, 89
- pki_integer.c
 - PKI_INTEGER_cmp, 42
 - PKI_INTEGER_dup, 42
 - PKI_INTEGER_free, 42
 - PKI_INTEGER_free_void, 42
 - PKI_INTEGER_get_parsed, 42
 - PKI_INTEGER_new, 42
 - PKI_INTEGER_new_bin, 42
 - PKI_INTEGER_new_char, 42
 - PKI_INTEGER_print, 42
 - PKI_INTEGER_print_fp, 42
- PKI_INTEGER_cmp
 - pki_integer.c, 42
- PKI_INTEGER_dup
 - pki_integer.c, 42

- PKI_INTEGER_free
 - pki_integer.c, [42](#)
- PKI_INTEGER_free_void
 - pki_integer.c, [42](#)
- PKI_INTEGER_get_parsed
 - pki_integer.c, [42](#)
- PKI_INTEGER_new
 - pki_integer.c, [42](#)
- PKI_INTEGER_new_bin
 - pki_integer.c, [42](#)
- PKI_INTEGER_new_char
 - pki_integer.c, [42](#)
- PKI_INTEGER_print
 - pki_integer.c, [42](#)
- PKI_INTEGER_print_fp
 - pki_integer.c, [42](#)
- pki_keypair.c
 - PKI_X509_KEYPAIR_free, [43](#)
 - PKI_X509_KEYPAIR_free_void, [43](#)
 - PKI_X509_KEYPAIR_get_p8, [43](#)
 - PKI_X509_KEYPAIR_get_parsed, [44](#)
 - PKI_X509_KEYPAIR_new, [44](#)
 - PKI_X509_KEYPAIR_new_null, [44](#)
 - PKI_X509_KEYPAIR_new_p8, [44](#)
 - PKI_X509_KEYPAIR_new_url, [44](#)
 - PKI_X509_KEYPAIR_pub_digest, [44](#)
 - PKI_X509_KEYPAIR_VALUE_pub_digest, [44](#)
- pki_keypair_io.c
 - PKI_X509_KEYPAIR_get, [5](#)
 - PKI_X509_KEYPAIR_get_mem, [5](#)
 - PKI_X509_KEYPAIR_get_url, [5](#)
 - PKI_X509_KEYPAIR_put, [5](#)
 - PKI_X509_KEYPAIR_put_mem, [5](#)
 - PKI_X509_KEYPAIR_put_url, [5](#)
 - PKI_X509_KEYPAIR_STACK_get, [5](#)
 - PKI_X509_KEYPAIR_STACK_get_url, [5](#)
- PKI_list_all_id
 - pki_init.c, [89](#)
- PKI_list_all_tokens
 - pki_init.c, [89](#)
- PKI_list_all_tokens_dir
 - pki_init.c, [89](#)
- PKI_log
 - pki_log.c, [90](#)
- pki_log.c
 - PKI_log, [90](#)
 - PKI_log_debug_simple, [90](#)
 - PKI_log_end, [90](#)
 - PKI_log_err_simple, [90](#)
 - PKI_log_init, [90](#)
- PKI_log_debug_simple
 - pki_log.c, [90](#)
- PKI_log_end
 - pki_log.c, [90](#)
- PKI_log_err_simple
 - pki_log.c, [90](#)
- PKI_log_init
 - pki_log.c, [90](#)
- PKI_Malloc
 - pki_mem.c, [92](#)
- pki_mem.c
 - PKI_Free, [92](#)
 - PKI_Malloc, [92](#)
 - PKI_MEM_add, [92](#)
 - PKI_MEM_B64_decode, [92](#)
 - PKI_MEM_B64_encode, [92](#)
 - PKI_MEM_dup, [92](#)
 - PKI_MEM_fprintf, [93](#)
 - PKI_MEM_free, [93](#)
 - PKI_MEM_free_void, [93](#)
 - PKI_MEM_get_data, [93](#)
 - PKI_MEM_get_size, [93](#)
 - PKI_MEM_grow, [93](#)
 - PKI_MEM_new, [93](#)
 - PKI_MEM_new_bio, [93](#)
 - PKI_MEM_new_data, [93](#)
 - PKI_MEM_new_func, [93](#)
 - PKI_MEM_new_func_bio, [93](#)
 - PKI_MEM_new_membio, [93](#)
 - PKI_MEM_new_null, [93](#)
 - PKI_MEM_printf, [94](#)
 - PKI_MEM_url_decode, [94](#)
 - PKI_MEM_url_encode, [94](#)
 - PKI_ZFree, [94](#)
 - PKI_ZFree_str, [94](#)
- PKI_MEM_add
 - pki_mem.c, [92](#)
- PKI_MEM_B64_decode
 - pki_mem.c, [92](#)
- PKI_MEM_B64_encode
 - pki_mem.c, [92](#)
- PKI_MEM_dup
 - pki_mem.c, [92](#)
- PKI_MEM_fprintf
 - pki_mem.c, [93](#)
- PKI_MEM_free
 - pki_mem.c, [93](#)
- PKI_MEM_free_void
 - pki_mem.c, [93](#)
- PKI_MEM_get_data
 - pki_mem.c, [93](#)
- PKI_MEM_get_size
 - pki_mem.c, [93](#)
- PKI_MEM_grow
 - pki_mem.c, [93](#)
- PKI_MEM_new
 - pki_mem.c, [93](#)

PKI_MEM_new_bio
 pki_mem.c, 93
PKI_MEM_new_data
 pki_mem.c, 93
PKI_MEM_new_func
 pki_mem.c, 93
PKI_MEM_new_func_bio
 pki_mem.c, 93
PKI_MEM_new_membio
 pki_mem.c, 93
PKI_MEM_new_null
 pki_mem.c, 93
PKI_MEM_printf
 pki_mem.c, 94
PKI_MEM_url_decode
 pki_mem.c, 94
PKI_MEM_url_encode
 pki_mem.c, 94
PKI_MSG_CMS_write
 cms_simple.c, 3
pki_msg_req.c
 PKI_MSG_REQ_add_data, 96
 PKI_MSG_REQ_add_recipient, 96
 PKI_MSG_REQ_clear_data, 96
 PKI_MSG_REQ_clear_recipients, 96
 PKI_MSG_REQ_encode, 96
 PKI_MSG_REQ_free, 96
 PKI_MSG_REQ_get_action, 97
 PKI_MSG_REQ_get_cacert, 97
 PKI_MSG_REQ_get_encoded, 97
 PKI_MSG_REQ_get_keypair, 97
 PKI_MSG_REQ_get_loa, 97
 PKI_MSG_REQ_get_proto, 97
 PKI_MSG_REQ_get_recipients, 97
 PKI_MSG_REQ_get_signer, 97
 PKI_MSG_REQ_get_subject, 97
 PKI_MSG_REQ_get_template, 97
 PKI_MSG_REQ_new, 97
 PKI_MSG_REQ_new_null, 97
 PKI_MSG_REQ_new_tk, 97
 PKI_MSG_REQ_replace_data, 98
 PKI_MSG_REQ_SCEP_new, 98
 PKI_MSG_REQ_SCEP_send, 98
 PKI_MSG_REQ_send, 98
 PKI_MSG_REQ_set_action, 98
 PKI_MSG_REQ_set_cacert, 98
 PKI_MSG_REQ_set_keypair, 98
 PKI_MSG_REQ_set_loa, 98
 PKI_MSG_REQ_set_proto, 98
 PKI_MSG_REQ_set_recipients, 98
 PKI_MSG_REQ_set_signer, 98
 PKI_MSG_REQ_set_subject, 98
 PKI_MSG_REQ_set_template, 99
PKI_MSG_REQ_add_data
 pki_msg_req.c, 96
PKI_MSG_REQ_add_recipient
 pki_msg_req.c, 96
PKI_MSG_REQ_clear_data
 pki_msg_req.c, 96
PKI_MSG_REQ_clear_recipients
 pki_msg_req.c, 96
PKI_MSG_REQ_encode
 pki_msg_req.c, 96
PKI_MSG_REQ_free
 pki_msg_req.c, 96
PKI_MSG_REQ_get_action
 pki_msg_req.c, 97
PKI_MSG_REQ_get_cacert
 pki_msg_req.c, 97
PKI_MSG_REQ_get_encoded
 pki_msg_req.c, 97
PKI_MSG_REQ_get_keypair
 pki_msg_req.c, 97
PKI_MSG_REQ_get_loa
 pki_msg_req.c, 97
PKI_MSG_REQ_get_proto
 pki_msg_req.c, 97
PKI_MSG_REQ_get_recipients
 pki_msg_req.c, 97
PKI_MSG_REQ_get_signer
 pki_msg_req.c, 97
PKI_MSG_REQ_get_subject
 pki_msg_req.c, 97
PKI_MSG_REQ_get_template
 pki_msg_req.c, 97
pki_msg_req_io.c
 PKI_MSG_REQ_put, 6
 PKI_MSG_REQ_put_mem, 6
PKI_MSG_REQ_new
 pki_msg_req.c, 97
PKI_MSG_REQ_new_null
 pki_msg_req.c, 97
PKI_MSG_REQ_new_tk
 pki_msg_req.c, 97
PKI_MSG_REQ_put
 pki_msg_req_io.c, 6
PKI_MSG_REQ_put_mem
 pki_msg_req_io.c, 6
PKI_MSG_REQ_replace_data
 pki_msg_req.c, 98
PKI_MSG_REQ_SCEP_new
 pki_msg_req.c, 98
PKI_MSG_REQ_SCEP_send
 pki_msg_req.c, 98
PKI_MSG_REQ_send
 pki_msg_req.c, 98
PKI_MSG_REQ_set_action
 pki_msg_req.c, 98

PKI_MSG_REQ_set_cacert
 pki_msg_req.c, 98
PKI_MSG_REQ_set_keypair
 pki_msg_req.c, 98
PKI_MSG_REQ_set_loa
 pki_msg_req.c, 98
PKI_MSG_REQ_set_proto
 pki_msg_req.c, 98
PKI_MSG_REQ_set_recipients
 pki_msg_req.c, 98
PKI_MSG_REQ_set_signer
 pki_msg_req.c, 98
PKI_MSG_REQ_set_subject
 pki_msg_req.c, 98
PKI_MSG_REQ_set_template
 pki_msg_req.c, 99
pki_msg_resp.c
 PKI_MSG_RESP_add_data, 101
 PKI_MSG_RESP_add_recipient, 101
 PKI_MSG_RESP_clear_data, 101
 PKI_MSG_RESP_clear_recipients, 101
 PKI_MSG_RESP_encode, 101
 PKI_MSG_RESP_free, 101
 PKI_MSG_RESP_get_action, 101
 PKI_MSG_RESP_get_cacert, 101
 PKI_MSG_RESP_get_encoded, 101
 PKI_MSG_RESP_get_issued_cert, 101
 PKI_MSG_RESP_get_keypair, 101
 PKI_MSG_RESP_get_proto, 101
 PKI_MSG_RESP_get_recipients, 101
 PKI_MSG_RESP_get_signer, 102
 PKI_MSG_RESP_get_status, 102
 PKI_MSG_RESP_new, 102
 PKI_MSG_RESP_new_null, 102
 PKI_MSG_RESP_new_tk, 102
 PKI_MSG_RESP_replace_data, 102
 PKI_MSG_RESP_SCEP_new, 102
 PKI_MSG_RESP_set_action, 102
 PKI_MSG_RESP_set_cacert, 102
 PKI_MSG_RESP_set_issued_cert, 102
 PKI_MSG_RESP_set_keypair, 102
 PKI_MSG_RESP_set_proto, 103
 PKI_MSG_RESP_set_recipients, 103
 PKI_MSG_RESP_set_signer, 103
 PKI_MSG_RESP_set_status, 103
PKI_MSG_RESP_add_data
 pki_msg_resp.c, 101
PKI_MSG_RESP_add_recipient
 pki_msg_resp.c, 101
PKI_MSG_RESP_clear_data
 pki_msg_resp.c, 101
PKI_MSG_RESP_clear_recipients
 pki_msg_resp.c, 101
PKI_MSG_RESP_encode
 pki_msg_resp.c, 101
PKI_MSG_RESP_free
 pki_msg_resp.c, 101
PKI_MSG_RESP_get_action
 pki_msg_resp.c, 101
PKI_MSG_RESP_get_cacert
 pki_msg_resp.c, 101
PKI_MSG_RESP_get_encoded
 pki_msg_resp.c, 101
PKI_MSG_RESP_get_issued_cert
 pki_msg_resp.c, 101
PKI_MSG_RESP_get_keypair
 pki_msg_resp.c, 101
PKI_MSG_RESP_get_proto
 pki_msg_resp.c, 101
PKI_MSG_RESP_get_recipients
 pki_msg_resp.c, 101
PKI_MSG_RESP_get_signer
 pki_msg_resp.c, 102
PKI_MSG_RESP_get_status
 pki_msg_resp.c, 102
pki_msg_resp_io.c
 PKI_MSG_RESP_put, 6
 PKI_MSG_RESP_put_mem, 6
PKI_MSG_RESP_new
 pki_msg_resp.c, 102
PKI_MSG_RESP_new_null
 pki_msg_resp.c, 102
PKI_MSG_RESP_new_tk
 pki_msg_resp.c, 102
PKI_MSG_RESP_put
 pki_msg_resp_io.c, 6
PKI_MSG_RESP_put_mem
 pki_msg_resp_io.c, 6
PKI_MSG_RESP_replace_data
 pki_msg_resp.c, 102
PKI_MSG_RESP_SCEP_new
 pki_msg_resp.c, 102
PKI_MSG_RESP_set_action
 pki_msg_resp.c, 102
PKI_MSG_RESP_set_cacert
 pki_msg_resp.c, 102
PKI_MSG_RESP_set_issued_cert
 pki_msg_resp.c, 102
PKI_MSG_RESP_set_keypair
 pki_msg_resp.c, 102
PKI_MSG_RESP_set_proto
 pki_msg_resp.c, 103
PKI_MSG_RESP_set_recipients
 pki_msg_resp.c, 103
PKI_MSG_RESP_set_signer
 pki_msg_resp.c, 103
PKI_MSG_RESP_set_status
 pki_msg_resp.c, 103

- PKI_MUTEX_acquire
 - pki_threads_vars.c, 106
- PKI_MUTEX_destroy
 - pki_threads_vars.c, 106
- PKI_MUTEX_free
 - pki_threads_vars.c, 106
- PKI_MUTEX_init
 - pki_threads_vars.c, 106
- PKI_MUTEX_new
 - pki_threads_vars.c, 106
- PKI_MUTEX_release
 - pki_threads_vars.c, 106
- PKI_NET_accept
 - sock.c, 29
- PKI_NET_close
 - sock.c, 29
- PKI_NET_get_data
 - sock.c, 30
- PKI_NET_listen
 - sock.c, 30
- PKI_NET_open
 - sock.c, 30
- PKI_NET_read
 - sock.c, 30
- PKI_NET_socket
 - sock.c, 30
- PKI_NET_write
 - sock.c, 30
- PKI_OCSP_nonce_check
 - pki_ocsp_req.c, 46
- pki_ocsp_req.c
 - d2i_OCSP_REQ_bio, 46
 - i2d_OCSP_REQ_bio, 46
 - PEM_read_bio_OCSP_REQ, 46
 - PEM_write_bio_OCSP_REQ, 46
 - PKI_OCSP_nonce_check, 46
 - PKI_X509_OCSP_REQ_add_cert, 46
 - PKI_X509_OCSP_REQ_add_longlong, 46
 - PKI_X509_OCSP_REQ_add_nonce, 46
 - PKI_X509_OCSP_REQ_add_serial, 46
 - PKI_X509_OCSP_REQ_add_txt, 47
 - PKI_X509_OCSP_REQ_DATA_sign, 47
 - PKI_X509_OCSP_REQ_elements, 47
 - PKI_X509_OCSP_REQ_free, 47
 - PKI_X509_OCSP_REQ_free_void, 47
 - PKI_X509_OCSP_REQ_get_cid, 47
 - PKI_X509_OCSP_REQ_get_data, 47
 - PKI_X509_OCSP_REQ_get_parsed, 47
 - PKI_X509_OCSP_REQ_get_serial, 47
 - PKI_X509_OCSP_REQ_new, 47
 - PKI_X509_OCSP_REQ_new_null, 47
 - PKI_X509_OCSP_REQ_print_parsed, 47
 - PKI_X509_OCSP_REQ_sign, 48
 - PKI_X509_OCSP_REQ_sign_tk, 48
- pki_ocsp_req_io.c
 - PKI_X509_OCSP_REQ_get, 7
 - PKI_X509_OCSP_REQ_get_mem, 7
 - PKI_X509_OCSP_REQ_get_url, 7
 - PKI_X509_OCSP_REQ_put, 7
 - PKI_X509_OCSP_REQ_put_mem, 7
 - PKI_X509_OCSP_REQ_put_url, 8
 - PKI_X509_OCSP_REQ_STACK_get, 8
 - PKI_X509_OCSP_REQ_STACK_get_mem, 8
 - PKI_X509_OCSP_REQ_STACK_get_url, 8
 - PKI_X509_OCSP_REQ_STACK_put, 8
 - PKI_X509_OCSP_REQ_STACK_put_mem, 8
 - PKI_X509_OCSP_REQ_STACK_put_url, 8
- pki_ocsp_resp.c
 - d2i_PKI_OCSP_RESP_bio, 49
 - i2d_PKI_OCSP_RESP_bio, 49
 - PEM_read_bio_PKI_OCSP_RESP, 49
 - PEM_write_bio_PKI_OCSP_RESP, 49
 - PKI_OCSP_RESP_free, 49
 - PKI_OCSP_RESP_new, 49
 - PKI_X509_OCSP_RESP_add, 49
 - PKI_X509_OCSP_RESP_copy_nonce, 49
 - PKI_X509_OCSP_RESP_DATA_sign, 50
 - PKI_X509_OCSP_RESP_free, 50
 - PKI_X509_OCSP_RESP_free_void, 50
 - PKI_X509_OCSP_RESP_get_data, 50
 - PKI_X509_OCSP_RESP_get_parsed, 50
 - PKI_X509_OCSP_RESP_new, 50
 - PKI_X509_OCSP_RESP_new_null, 50
 - PKI_X509_OCSP_RESP_print_parsed, 50
 - PKI_X509_OCSP_RESP_set_status, 50
 - PKI_X509_OCSP_RESP_sign, 50
 - PKI_X509_OCSP_RESP_sign_tk, 50
- PKI_OCSP_RESP_free
 - pki_ocsp_resp.c, 49
- pki_ocsp_resp_io.c
 - PKI_X509_OCSP_RESP_get, 9
 - PKI_X509_OCSP_RESP_get_mem, 9
 - PKI_X509_OCSP_RESP_get_url, 9
 - PKI_X509_OCSP_RESP_put, 9
 - PKI_X509_OCSP_RESP_put_mem, 9
 - PKI_X509_OCSP_RESP_put_url, 9
 - PKI_X509_OCSP_RESP_STACK_get, 9
 - PKI_X509_OCSP_RESP_STACK_get_mem, 9
 - PKI_X509_OCSP_RESP_STACK_get_url, 9
 - PKI_X509_OCSP_RESP_STACK_put, 9
 - PKI_X509_OCSP_RESP_STACK_put_mem, 9
 - PKI_X509_OCSP_RESP_STACK_put_url, 10
- PKI_OCSP_RESP_new
 - pki_ocsp_resp.c, 49
- pki_oid.c
 - PKI_OID_cmp, 51

- PKI_OID_dup, [51](#)
- PKI_OID_free, [51](#)
- PKI_OID_free_void, [52](#)
- PKI_OID_get, [52](#)
- PKI_OID_get_descr, [52](#)
- PKI_OID_get_id, [52](#)
- PKI_OID_get_str, [52](#)
- PKI_OID_new, [52](#)
- PKI_OID_new_id, [52](#)
- PKI_OID_new_text, [52](#)
- PKI_OID_cmp
 - pki_oid.c, [51](#)
- PKI_OID_dup
 - pki_oid.c, [51](#)
- PKI_OID_free
 - pki_oid.c, [51](#)
- PKI_OID_free_void
 - pki_oid.c, [52](#)
- PKI_OID_get
 - pki_oid.c, [52](#)
- PKI_OID_get_descr
 - pki_oid.c, [52](#)
- PKI_OID_get_id
 - pki_oid.c, [52](#)
- PKI_OID_get_str
 - pki_oid.c, [52](#)
- PKI_OID_new
 - pki_oid.c, [52](#)
- PKI_OID_new_id
 - pki_oid.c, [52](#)
- PKI_OID_new_text
 - pki_oid.c, [52](#)
- PKI_PRQP_CERTID_new
 - prqp_lib.c, [117](#)
- PKI_PRQP_CERTID_new_cert
 - prqp_lib.c, [117](#)
- PKI_PRQP_REQ_new_cert
 - prqp_lib.c, [117](#)
- PKI_RWLOCK_destroy
 - pki_threads_vars.c, [106](#)
- PKI_RWLOCK_free
 - pki_threads_vars.c, [106](#)
- PKI_RWLOCK_init
 - pki_threads_vars.c, [107](#)
- PKI_RWLOCK_new
 - pki_threads_vars.c, [107](#)
- PKI_RWLOCK_read_lock
 - pki_threads_vars.c, [107](#)
- PKI_RWLOCK_release
 - pki_threads_vars.c, [107](#)
- PKI_RWLOCK_try_read_lock
 - pki_threads_vars.c, [107](#)
- PKI_RWLOCK_try_write_lock
 - pki_threads_vars.c, [107](#)
- PKI_RWLOCK_write_lock
 - pki_threads_vars.c, [107](#)
- PKI_set_env
 - support.c, [135](#)
- PKI_set_err
 - pki_err.c, [88](#)
- PKI_SSL_close
 - ssl.c, [31](#)
- PKI_SSL_connect
 - ssl.c, [31](#)
- PKI_SSL_connect_url
 - ssl.c, [32](#)
- PKI_SSL_free
 - ssl.c, [32](#)
- PKI_SSL_get_peer_cert
 - ssl.c, [32](#)
- PKI_SSL_get_peer_chain
 - ssl.c, [32](#)
- PKI_SSL_get_servername
 - ssl.c, [32](#)
- PKI_SSL_new
 - ssl.c, [32](#)
- PKI_SSL_read
 - ssl.c, [32](#)
- PKI_SSL_set_algor
 - ssl.c, [32](#)
- PKI_SSL_set_cipher
 - ssl.c, [32](#)
- PKI_SSL_set_flags
 - ssl.c, [32](#)
- PKI_SSL_set_token
 - ssl.c, [32](#)
- PKI_SSL_set_trusted
 - ssl.c, [32](#)
- PKI_SSL_write
 - ssl.c, [32](#)
- PKI_STACK_del_num
 - stack.c, [133](#)
- PKI_STACK_elements
 - stack.c, [133](#)
- PKI_STACK_free
 - stack.c, [133](#)
- PKI_STACK_free_all
 - stack.c, [133](#)
- PKI_STACK_get_num
 - stack.c, [133](#)
- PKI_STACK_ins_num
 - stack.c, [134](#)
- PKI_STACK_new
 - stack.c, [134](#)
- PKI_STACK_new_null
 - stack.c, [134](#)
- PKI_STACK_new_type
 - stack.c, [134](#)

- PKI_STACK_pop
 - stack.c, [134](#)
- PKI_STACK_pop_free
 - stack.c, [134](#)
- PKI_STACK_push
 - stack.c, [134](#)
- PKI_STACK_X509_ATTRIBUTE_add
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_delete
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_delete_by_name
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_delete_by_num
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_free
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_free_all
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_get
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_get_by_name
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_get_by_num
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_num
 - pki_x509_attribute.c, [56](#)
- PKI_STACK_X509_ATTRIBUTE_replace
 - pki_x509_attribute.c, [56](#)
- pki_string.c
 - PKI_STRING_dup, [53](#)
 - PKI_STRING_free, [53](#)
 - PKI_STRING_get_parsed, [53](#)
 - PKI_STRING_get_type, [53](#)
 - PKI_STRING_get_utf8, [53](#)
 - PKI_STRING_new, [53](#)
 - PKI_STRING_new_null, [54](#)
 - PKI_STRING_print, [54](#)
 - PKI_STRING_print_fp, [54](#)
 - PKI_STRING_set, [54](#)
- PKI_STRING_dup
 - pki_string.c, [53](#)
- PKI_STRING_free
 - pki_string.c, [53](#)
- PKI_STRING_get_parsed
 - pki_string.c, [53](#)
- PKI_STRING_get_type
 - pki_string.c, [53](#)
- PKI_STRING_get_utf8
 - pki_string.c, [53](#)
- PKI_STRING_new
 - pki_string.c, [53](#)
- PKI_STRING_new_null
 - pki_string.c, [54](#)
- PKI_STRING_print
 - pki_string.c, [54](#)
- PKI_STRING_print_fp
 - pki_string.c, [54](#)
- PKI_STRING_set
 - pki_string.c, [54](#)
- PKI_THREAD_create
 - pki_threads.c, [103](#)
- PKI_THREAD_new
 - pki_threads.c, [103](#)
- PKI_THREAD_self
 - pki_threads.c, [104](#)
- pki_threads.c
 - PKI_THREAD_create, [103](#)
 - PKI_THREAD_new, [103](#)
 - PKI_THREAD_self, [104](#)
- pki_threads_vars.c
 - PKI_clock_gettime, [105](#)
 - PKI_COND_broadcast, [105](#)
 - PKI_COND_destroy, [105](#)
 - PKI_COND_free, [105](#)
 - PKI_COND_init, [106](#)
 - PKI_COND_new, [106](#)
 - PKI_COND_signal, [106](#)
 - PKI_COND_timedwait, [106](#)
 - PKI_COND_wait, [106](#)
 - PKI_MUTEX_acquire, [106](#)
 - PKI_MUTEX_destroy, [106](#)
 - PKI_MUTEX_free, [106](#)
 - PKI_MUTEX_init, [106](#)
 - PKI_MUTEX_new, [106](#)
 - PKI_MUTEX_release, [106](#)
 - PKI_RWLOCK_destroy, [106](#)
 - PKI_RWLOCK_free, [106](#)
 - PKI_RWLOCK_init, [107](#)
 - PKI_RWLOCK_new, [107](#)
 - PKI_RWLOCK_read_lock, [107](#)
 - PKI_RWLOCK_release, [107](#)
 - PKI_RWLOCK_try_read_lock, [107](#)
 - PKI_RWLOCK_try_write_lock, [107](#)
 - PKI_RWLOCK_write_lock, [107](#)
- pki_time.c
 - PKI_TIME_dup, [54](#)
 - PKI_TIME_free, [54](#)
 - PKI_TIME_free_void, [54](#)
 - PKI_TIME_get_parsed, [54](#)
 - PKI_TIME_new, [55](#)
 - PKI_TIME_print, [55](#)
 - PKI_TIME_print_fp, [55](#)
- PKI_TIME_dup
 - pki_time.c, [54](#)
- PKI_TIME_free
 - pki_time.c, [54](#)
- PKI_TIME_free_void
 - pki_time.c, [54](#)

- pki_time.c, [54](#)
- PKI_TIME_get_parsed
 - pki_time.c, [54](#)
- PKI_TIME_new
 - pki_time.c, [55](#)
- PKI_TIME_print
 - pki_time.c, [55](#)
- PKI_TIME_print_fp
 - pki_time.c, [55](#)
- PKI_TOKEN_add_profile
 - token.c, [139](#)
- PKI_TOKEN_check
 - token.c, [139](#)
- PKI_TOKEN_cred_cb_env
 - token.c, [139](#)
- PKI_TOKEN_cred_cb_stdin
 - token.c, [139](#)
- PKI_TOKEN_cred_get
 - token.c, [139](#)
- PKI_TOKEN_cred_set_cb
 - token.c, [139](#)
- PKI_TOKEN_del_url
 - token.c, [139](#)
- PKI_TOKEN_export_cert
 - token.c, [139](#)
- PKI_TOKEN_export_keypair
 - token.c, [139](#)
- PKI_TOKEN_export_keypair_url
 - token.c, [139](#)
- PKI_TOKEN_export_otherCerts
 - token.c, [140](#)
- PKI_TOKEN_export_p12
 - token.c, [140](#)
- PKI_TOKEN_export_req
 - token.c, [140](#)
- PKI_TOKEN_export_trustedCerts
 - token.c, [140](#)
- PKI_TOKEN_free
 - token.c, [140](#)
- PKI_TOKEN_free_void
 - token.c, [140](#)
- PKI_TOKEN_get_cacert
 - token.c, [140](#)
- PKI_TOKEN_get_cacert_data
 - token_data.c, [147](#)
- PKI_TOKEN_get_cert
 - token.c, [140](#)
- PKI_TOKEN_get_cert_data
 - token_data.c, [147](#)
- PKI_TOKEN_get_config_dir
 - token.c, [140](#)
- PKI_TOKEN_get_cred
 - token.c, [140](#)
- PKI_TOKEN_get_crls
 - token.c, [141](#)
- PKI_TOKEN_get_identity_data
 - token_data.c, [147](#)
- PKI_TOKEN_get_keypair
 - token.c, [141](#)
- PKI_TOKEN_get_keypair_data
 - token_data.c, [147](#)
- PKI_TOKEN_get_name
 - token.c, [141](#)
- PKI_TOKEN_get_otherCerts
 - token.c, [141](#)
- PKI_TOKEN_get_otherCerts_data
 - token_data.c, [147](#)
- PKI_TOKEN_get_p12
 - token.c, [141](#)
- PKI_TOKEN_get_privkey_data
 - token_data.c, [147](#)
- PKI_TOKEN_get_pubkey_data
 - token_data.c, [147](#)
- PKI_TOKEN_get_trustedCerts
 - token.c, [141](#)
- PKI_TOKEN_get_trustedCerts_data
 - token_data.c, [147](#)
- PKI_TOKEN_ID_INFO_get
 - token_id.c, [148](#)
- PKI_TOKEN_ID_INFO_list
 - token_id.c, [148](#)
- PKI_TOKEN_ID_num
 - token_id.c, [148](#)
- PKI_TOKEN_ID_set
 - token_id.c, [148](#)
- PKI_TOKEN_import_cert
 - token.c, [141](#)
- PKI_TOKEN_import_cert_stack
 - token.c, [141](#)
- PKI_TOKEN_import_keypair
 - token.c, [142](#)
- PKI_TOKEN_init
 - token.c, [142](#)
- PKI_TOKEN_issue_cert
 - token.c, [142](#)
- PKI_TOKEN_issue_crl
 - token.c, [142](#)
- PKI_TOKEN_issue_proxy
 - token.c, [142](#)
- PKI_TOKEN_load_cacert
 - token.c, [142](#)
- PKI_TOKEN_load_cert
 - token.c, [142](#)
- PKI_TOKEN_load_config
 - token.c, [142](#)
- PKI_TOKEN_load_crls
 - token.c, [143](#)
- PKI_TOKEN_load_keypair

- token.c, [143](#)
- PKI_TOKEN_load_otherCerts
 - token.c, [143](#)
- PKI_TOKEN_load_profiles
 - token.c, [143](#)
- PKI_TOKEN_load_req
 - token.c, [143](#)
- PKI_TOKEN_load_trustedCerts
 - token.c, [143](#)
- PKI_TOKEN_login
 - token.c, [143](#)
- PKI_TOKEN_new
 - token.c, [143](#)
- PKI_TOKEN_new_keypair
 - token.c, [144](#)
- PKI_TOKEN_new_keypair_url
 - token.c, [144](#)
- PKI_TOKEN_new_null
 - token.c, [144](#)
- PKI_TOKEN_new_p12
 - token.c, [144](#)
- PKI_TOKEN_new_req
 - token.c, [144](#)
- PKI_TOKEN_OID_new
 - token.c, [144](#)
- PKI_TOKEN_print_info
 - token.c, [144](#)
- PKI_TOKEN_search_profile
 - token.c, [144](#)
- PKI_TOKEN_self_sign
 - token.c, [144](#)
- PKI_TOKEN_set_algor
 - token.c, [144](#)
- PKI_TOKEN_set_algor_by_name
 - token.c, [145](#)
- PKI_TOKEN_set_cacert
 - token.c, [145](#)
- PKI_TOKEN_set_cert
 - token.c, [145](#)
- PKI_TOKEN_set_config_dir
 - token.c, [145](#)
- PKI_TOKEN_set_cred
 - token.c, [145](#)
- PKI_TOKEN_set_crls
 - token.c, [145](#)
- PKI_TOKEN_set_keypair
 - token.c, [146](#)
- PKI_TOKEN_set_otherCerts
 - token.c, [146](#)
- PKI_TOKEN_set_req
 - token.c, [146](#)
- PKI_TOKEN_set_trustedCerts
 - token.c, [146](#)
- PKI_TOKEN_use_slot
 - token.c, [146](#)
- pki_x509.c
 - PKI_X509_CALLBACKS_get, [109](#)
 - PKI_X509_delete, [109](#)
 - PKI_X509_dup, [109](#)
 - PKI_X509_dup_value, [109](#)
 - PKI_X509_free, [109](#)
 - PKI_X509_free_void, [109](#)
 - PKI_X509_get_data, [109](#)
 - PKI_X509_get_hsm, [109](#)
 - PKI_X509_get_parsed, [109](#)
 - PKI_X509_get_reference, [109](#)
 - PKI_X509_get_type, [109](#)
 - PKI_X509_get_value, [109](#)
 - PKI_X509_is_signed, [109](#)
 - PKI_X509_new, [109](#)
 - PKI_X509_new_dup_value, [110](#)
 - PKI_X509_new_value, [110](#)
 - PKI_X509_print_parsed, [110](#)
 - PKI_X509_set_hsm, [110](#)
 - PKI_X509_set_reference, [110](#)
 - PKI_X509_set_value, [110](#)
- pki_x509_attribute.c
 - PKI_STACK_X509_ATTRIBUTE_add, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_delete, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_delete_by_name, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_delete_by_num, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_free, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_free_all, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_get, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_get_by_name, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_get_by_num, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_num, [56](#)
 - PKI_STACK_X509_ATTRIBUTE_replace, [56](#)
 - PKI_X509_ATTRIBUTE_free, [56](#)
 - PKI_X509_ATTRIBUTE_free_null, [56](#)
 - PKI_X509_ATTRIBUTE_get_descr, [57](#)
 - PKI_X509_ATTRIBUTE_get_parsed, [57](#)
 - PKI_X509_ATTRIBUTE_get_value, [57](#)
 - PKI_X509_ATTRIBUTE_new, [57](#)
 - PKI_X509_ATTRIBUTE_new_name, [57](#)
 - PKI_X509_ATTRIBUTE_new_null, [57](#)
- PKI_X509_ATTRIBUTE_free
 - pki_x509_attribute.c, [56](#)
- PKI_X509_ATTRIBUTE_free_null
 - pki_x509_attribute.c, [56](#)
- PKI_X509_ATTRIBUTE_get_descr
 - pki_x509_attribute.c, [57](#)

- PKI_X509_ATTRIBUTE_get_parsed
 - pki_x509_attribute.c, [57](#)
- PKI_X509_ATTRIBUTE_get_value
 - pki_x509_attribute.c, [57](#)
- PKI_X509_ATTRIBUTE_new
 - pki_x509_attribute.c, [57](#)
- PKI_X509_ATTRIBUTE_new_name
 - pki_x509_attribute.c, [57](#)
- PKI_X509_ATTRIBUTE_new_null
 - pki_x509_attribute.c, [57](#)
- PKI_X509_CALLBACKS_get
 - pki_x509.c, [109](#)
- pki_x509_cert.c
 - NID_proxyCertInfo, [61](#)
 - PKI_X509_CERT_add_extension, [59](#)
 - PKI_X509_CERT_add_extension_stack, [59](#)
 - PKI_X509_CERT_check_domain, [59](#)
 - PKI_X509_CERT_dup, [59](#)
 - PKI_X509_CERT_fingerprint, [59](#)
 - PKI_X509_CERT_fingerprint_by_name, [59](#)
 - PKI_X509_CERT_free, [59](#)
 - PKI_X509_CERT_free_void, [59](#)
 - PKI_X509_CERT_get_cdp, [59](#)
 - PKI_X509_CERT_get_data, [59](#)
 - PKI_X509_CERT_get_email, [60](#)
 - PKI_X509_CERT_get_extension_by_id, [60](#)
 - PKI_X509_CERT_get_extension_by_name, [60](#)
 - PKI_X509_CERT_get_extension_by_oid, [60](#)
 - PKI_X509_CERT_get_extensions, [60](#)
 - PKI_X509_CERT_get_keysize, [60](#)
 - PKI_X509_CERT_get_parsed, [60](#)
 - PKI_X509_CERT_get_type, [60](#)
 - PKI_X509_CERT_is_ca, [60](#)
 - PKI_X509_CERT_is_proxy, [60](#)
 - PKI_X509_CERT_is_selfsigned, [60](#)
 - PKI_X509_CERT_key_hash, [60](#)
 - PKI_X509_CERT_key_hash_by_name, [60](#)
 - PKI_X509_CERT_new, [61](#)
 - PKI_X509_CERT_new_null, [61](#)
 - PKI_X509_CERT_print_parsed, [61](#)
- PKI_X509_CERT_add_extension
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_add_extension_stack
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_check_domain
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_dup
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_fingerprint
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_fingerprint_by_name
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_free
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_free_void
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_get
 - pki_x509_cert_io.c, [11](#)
- PKI_X509_CERT_get_cdp
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_get_data
 - pki_x509_cert.c, [59](#)
- PKI_X509_CERT_get_email
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_get_extension_by_id
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_get_extension_by_name
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_get_extension_by_oid
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_get_extensions
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_get_keysize
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_get_mem
 - pki_x509_cert_io.c, [11](#)
- PKI_X509_CERT_get_parsed
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_get_type
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_get_url
 - pki_x509_cert_io.c, [11](#)
- pki_x509_cert_io.c
 - PKI_X509_CERT_get, [11](#)
 - PKI_X509_CERT_get_mem, [11](#)
 - PKI_X509_CERT_get_url, [11](#)
 - PKI_X509_CERT_put, [11](#)
 - PKI_X509_CERT_put_mem, [11](#)
 - PKI_X509_CERT_put_url, [11](#)
 - PKI_X509_CERT_STACK_get, [11](#)
 - PKI_X509_CERT_STACK_get_mem, [11](#)
 - PKI_X509_CERT_STACK_get_url, [11](#)
 - PKI_X509_CERT_STACK_put, [12](#)
 - PKI_X509_CERT_STACK_put_mem, [12](#)
 - PKI_X509_CERT_STACK_put_url, [12](#)
- PKI_X509_CERT_is_ca
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_is_proxy
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_is_selfsigned
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_key_hash
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_key_hash_by_name
 - pki_x509_cert.c, [60](#)
- PKI_X509_CERT_new
 - pki_x509_cert.c, [61](#)

PKI_X509_CERT_new_null
 pki_x509_cert.c, 61
PKI_X509_CERT_print_parsed
 pki_x509_cert.c, 61
PKI_X509_CERT_put
 pki_x509_cert_io.c, 11
PKI_X509_CERT_put_mem
 pki_x509_cert_io.c, 11
PKI_X509_CERT_put_url
 pki_x509_cert_io.c, 11
PKI_X509_CERT_STACK_get
 pki_x509_cert_io.c, 11
PKI_X509_CERT_STACK_get_mem
 pki_x509_cert_io.c, 11
PKI_X509_CERT_STACK_get_url
 pki_x509_cert_io.c, 11
PKI_X509_CERT_STACK_put
 pki_x509_cert_io.c, 12
PKI_X509_CERT_STACK_put_mem
 pki_x509_cert_io.c, 12
PKI_X509_CERT_STACK_put_url
 pki_x509_cert_io.c, 12
pki_x509_crl.c
 PKI_X509_CRL_add_extension, 62
 PKI_X509_CRL_add_extension_stack, 62
 PKI_X509_CRL_ENTRY_free, 62
 PKI_X509_CRL_ENTRY_free_void, 62
 PKI_X509_CRL_ENTRY_new, 62
 PKI_X509_CRL_ENTRY_new_serial, 63
 PKI_X509_CRL_free, 63
 PKI_X509_CRL_free_void, 63
 PKI_X509_CRL_get_data, 63
 PKI_X509_CRL_get_parsed, 63
 PKI_X509_CRL_lookup, 63
 PKI_X509_CRL_lookup_cert, 63
 PKI_X509_CRL_lookup_long, 63
 PKI_X509_CRL_lookup_serial, 63
 PKI_X509_CRL_new, 64
 PKI_X509_CRL_print_parsed, 64
PKI_X509_CRL_add_extension
 pki_x509_crl.c, 62
PKI_X509_CRL_add_extension_stack
 pki_x509_crl.c, 62
PKI_X509_CRL_ENTRY_free
 pki_x509_crl.c, 62
PKI_X509_CRL_ENTRY_free_void
 pki_x509_crl.c, 62
PKI_X509_CRL_ENTRY_new
 pki_x509_crl.c, 62
PKI_X509_CRL_ENTRY_new_serial
 pki_x509_crl.c, 63
PKI_X509_CRL_free
 pki_x509_crl.c, 63
PKI_X509_CRL_free_void
 pki_x509_crl.c, 63
PKI_X509_CRL_get
 pki_x509_crl_io.c, 13
PKI_X509_CRL_get_data
 pki_x509_crl.c, 63
PKI_X509_CRL_get_mem
 pki_x509_crl_io.c, 13
PKI_X509_CRL_get_parsed
 pki_x509_crl.c, 63
PKI_X509_CRL_get_url
 pki_x509_crl_io.c, 13
pki_x509_crl_io.c
 PKI_X509_CRL_get, 13
 PKI_X509_CRL_get_mem, 13
 PKI_X509_CRL_get_url, 13
 PKI_X509_CRL_put, 13
 PKI_X509_CRL_put_mem, 13
 PKI_X509_CRL_put_url, 13
 PKI_X509_CRL_STACK_get, 13
 PKI_X509_CRL_STACK_get_mem, 13
 PKI_X509_CRL_STACK_get_url, 14
 PKI_X509_CRL_STACK_put, 14
 PKI_X509_CRL_STACK_put_mem, 14
 PKI_X509_CRL_STACK_put_url, 14
PKI_X509_CRL_lookup
 pki_x509_crl.c, 63
PKI_X509_CRL_lookup_cert
 pki_x509_crl.c, 63
PKI_X509_CRL_lookup_long
 pki_x509_crl.c, 63
PKI_X509_CRL_lookup_serial
 pki_x509_crl.c, 63
PKI_X509_CRL_new
 pki_x509_crl.c, 64
PKI_X509_CRL_print_parsed
 pki_x509_crl.c, 64
PKI_X509_CRL_put
 pki_x509_crl_io.c, 13
PKI_X509_CRL_put_mem
 pki_x509_crl_io.c, 13
PKI_X509_CRL_put_url
 pki_x509_crl_io.c, 13
PKI_X509_CRL_STACK_get
 pki_x509_crl_io.c, 13
PKI_X509_CRL_STACK_get_mem
 pki_x509_crl_io.c, 13
PKI_X509_CRL_STACK_get_url
 pki_x509_crl_io.c, 14
PKI_X509_CRL_STACK_put
 pki_x509_crl_io.c, 14
PKI_X509_CRL_STACK_put_mem
 pki_x509_crl_io.c, 14
PKI_X509_CRL_STACK_put_url
 pki_x509_crl_io.c, 14

- PKI_X509_delete
 - pki_x509.c, 109
- PKI_X509_dup
 - pki_x509.c, 109
- PKI_X509_dup_value
 - pki_x509.c, 109
- pki_x509_extension.c
 - PKI_X509_EXTENSION_free, 64
 - PKI_X509_EXTENSION_free_void, 64
 - PKI_X509_EXTENSION_get_list, 64
 - PKI_X509_EXTENSION_new, 64
 - PKI_X509_EXTENSION_value_new_profile, 64
- PKI_X509_EXTENSION_free
 - pki_x509_extension.c, 64
- PKI_X509_EXTENSION_free_void
 - pki_x509_extension.c, 64
- PKI_X509_EXTENSION_get_list
 - pki_x509_extension.c, 64
- PKI_X509_EXTENSION_new
 - pki_x509_extension.c, 64
- PKI_X509_EXTENSION_value_new_profile
 - pki_x509_extension.c, 64
- PKI_X509_EXTENSIONS_cert_add_profile
 - extensions.c, 4
- PKI_X509_EXTENSIONS_crl_add_profile
 - extensions.c, 4
- PKI_X509_EXTENSIONS_req_add_profile
 - extensions.c, 4
- PKI_X509_free
 - pki_x509.c, 109
- PKI_X509_free_void
 - pki_x509.c, 109
- PKI_X509_get
 - pki_x509_io.c, 15
- PKI_X509_get_data
 - pki_x509.c, 109
- PKI_X509_get_hsm
 - pki_x509.c, 109
- PKI_X509_get_mem
 - pki_x509_mem.c, 111
- PKI_X509_get_mem_value
 - pki_x509_mem.c, 111
- PKI_X509_get_mimetype
 - pki_x509_mime.c, 112
- PKI_X509_get_parsed
 - pki_x509.c, 109
- PKI_X509_get_reference
 - pki_x509.c, 109
- PKI_X509_get_type
 - pki_x509.c, 109
- PKI_X509_get_url
 - pki_x509_io.c, 15
- PKI_X509_get_value
 - pki_x509.c, 109
- pki_x509_io.c
 - PKI_get_value, 15
 - PKI_X509_get, 15
 - PKI_X509_get_url, 15
 - PKI_X509_put, 15
 - PKI_X509_put_url, 15
 - PKI_X509_put_value, 16
 - PKI_X509_STACK_get, 16
 - PKI_X509_STACK_get_url, 16
 - PKI_X509_STACK_put, 16
 - PKI_X509_STACK_put_url, 16
- PKI_X509_is_signed
 - pki_x509.c, 109
- PKI_X509_KEYPAIR_free
 - pki_keypair.c, 43
- PKI_X509_KEYPAIR_free_void
 - pki_keypair.c, 43
- PKI_X509_KEYPAIR_get
 - pki_keypair_io.c, 5
- PKI_X509_KEYPAIR_get_mem
 - pki_keypair_io.c, 5
- PKI_X509_KEYPAIR_get_p8
 - pki_keypair.c, 43
- PKI_X509_KEYPAIR_get_parsed
 - pki_keypair.c, 44
- PKI_X509_KEYPAIR_get_url
 - pki_keypair_io.c, 5
- PKI_X509_KEYPAIR_new
 - pki_keypair.c, 44
- PKI_X509_KEYPAIR_new_null
 - pki_keypair.c, 44
- PKI_X509_KEYPAIR_new_p8
 - pki_keypair.c, 44
- PKI_X509_KEYPAIR_new_url
 - pki_keypair.c, 44
- PKI_X509_KEYPAIR_pub_digest
 - pki_keypair.c, 44
- PKI_X509_KEYPAIR_put
 - pki_keypair_io.c, 5
- PKI_X509_KEYPAIR_put_mem
 - pki_keypair_io.c, 5
- PKI_X509_KEYPAIR_put_url
 - pki_keypair_io.c, 5
- PKI_X509_KEYPAIR_STACK_get
 - pki_keypair_io.c, 5
- PKI_X509_KEYPAIR_STACK_get_url
 - pki_keypair_io.c, 5
- PKI_X509_KEYPAIR_VALUE_pub_digest
 - pki_keypair.c, 44
- pki_x509_mem.c
 - PKI_X509_get_mem, 111
 - PKI_X509_get_mem_value, 111
 - PKI_X509_put_mem, 111

- PKI_X509_put_mem_value, [111](#)
- PKI_X509_STACK_get_mem, [111](#)
- PKI_X509_STACK_put_mem, [111](#)
- pki_x509_mime.c
 - PKI_X509_get_mimetype, [112](#)
- pki_x509_name.c
 - PKI_X509_NAME_add, [66](#)
 - PKI_X509_NAME_cmp, [66](#)
 - PKI_X509_NAME_dup, [66](#)
 - PKI_X509_NAME_free, [66](#)
 - PKI_X509_NAME_get_digest, [66](#)
 - PKI_X509_NAME_get_list, [66](#)
 - PKI_X509_NAME_get_parsed, [66](#)
 - PKI_X509_NAME_list_free, [66](#)
 - PKI_X509_NAME_new, [66](#)
 - PKI_X509_NAME_new_null, [66](#)
 - PKI_X509_NAME_RDN_type_descr, [66](#)
 - PKI_X509_NAME_RDN_type_id, [66](#)
 - PKI_X509_NAME_RDN_type_text, [66](#)
 - PKI_X509_NAME_RDN_value, [66](#)
- PKI_X509_NAME_add
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_cmp
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_dup
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_free
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_get_digest
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_get_list
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_get_parsed
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_list_free
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_new
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_new_null
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_RDN_type_descr
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_RDN_type_id
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_RDN_type_text
 - pki_x509_name.c, [66](#)
- PKI_X509_NAME_RDN_value
 - pki_x509_name.c, [66](#)
- PKI_X509_new
 - pki_x509.c, [109](#)
- PKI_X509_new_dup_value
 - pki_x509.c, [110](#)
- PKI_X509_new_value
 - pki_x509.c, [110](#)
- PKI_X509_OCSP_REQ_add_cert
 - pki_ocsp_req.c, [46](#)
- PKI_X509_OCSP_REQ_add_longlong
 - pki_ocsp_req.c, [46](#)
- PKI_X509_OCSP_REQ_add_nonce
 - pki_ocsp_req.c, [46](#)
- PKI_X509_OCSP_REQ_add_serial
 - pki_ocsp_req.c, [46](#)
- PKI_X509_OCSP_REQ_add_txt
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_DATA_sign
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_elements
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_free
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_free_void
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_get
 - pki_ocsp_req_io.c, [7](#)
- PKI_X509_OCSP_REQ_get_cid
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_get_data
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_get_mem
 - pki_ocsp_req_io.c, [7](#)
- PKI_X509_OCSP_REQ_get_parsed
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_get_serial
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_get_url
 - pki_ocsp_req_io.c, [7](#)
- PKI_X509_OCSP_REQ_new
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_new_null
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_print_parsed
 - pki_ocsp_req.c, [47](#)
- PKI_X509_OCSP_REQ_put
 - pki_ocsp_req_io.c, [7](#)
- PKI_X509_OCSP_REQ_put_mem
 - pki_ocsp_req_io.c, [7](#)
- PKI_X509_OCSP_REQ_put_url
 - pki_ocsp_req_io.c, [8](#)
- PKI_X509_OCSP_REQ_sign
 - pki_ocsp_req.c, [48](#)
- PKI_X509_OCSP_REQ_sign_tk
 - pki_ocsp_req.c, [48](#)
- PKI_X509_OCSP_REQ_STACK_get
 - pki_ocsp_req_io.c, [8](#)
- PKI_X509_OCSP_REQ_STACK_get_mem
 - pki_ocsp_req_io.c, [8](#)
- PKI_X509_OCSP_REQ_STACK_get_url
 - pki_ocsp_req_io.c, [8](#)

- PKI_X509_OCSP_REQ_STACK_put
 - pki_ocsp_req_io.c, [8](#)
- PKI_X509_OCSP_REQ_STACK_put_mem
 - pki_ocsp_req_io.c, [8](#)
- PKI_X509_OCSP_REQ_STACK_put_url
 - pki_ocsp_req_io.c, [8](#)
- PKI_X509_OCSP_RESP_add
 - pki_ocsp_resp.c, [49](#)
- PKI_X509_OCSP_RESP_copy_nonce
 - pki_ocsp_resp.c, [49](#)
- PKI_X509_OCSP_RESP_DATA_sign
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_free
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_free_void
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_get
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_get_data
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_get_mem
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_get_parsed
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_get_url
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_new
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_new_null
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_print_parsed
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_put
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_put_mem
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_put_url
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_set_status
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_sign
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_sign_tk
 - pki_ocsp_resp.c, [50](#)
- PKI_X509_OCSP_RESP_STACK_get
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_STACK_get_mem
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_STACK_get_url
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_STACK_put
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_STACK_put_mem
 - pki_ocsp_resp_io.c, [9](#)
- PKI_X509_OCSP_RESP_STACK_put_url
 - pki_ocsp_resp_io.c, [10](#)
- pki_x509_p12.c
 - BAG_DATATYPE_ALL, [68](#)
 - BAG_DATATYPE_CACERT, [68](#)
 - BAG_DATATYPE_CERT, [68](#)
 - BAG_DATATYPE_KEYPAIR, [68](#)
 - BAG_DATATYPE_OTHERCERTS, [68](#)
 - BAG_DATATYPE_UNKNOWN, [68](#)
- pki_x509_p12.c
 - bag_datatype_st, [68](#)
 - PEM_read_bio_PKCS12, [68](#)
 - PEM_write_bio_PKCS12, [68](#)
 - PKI_X509_PKCS12_DATA_add_certs, [68](#)
 - PKI_X509_PKCS12_DATA_add_keypair, [68](#)
 - PKI_X509_PKCS12_DATA_add_other_certs, [68](#)
 - PKI_X509_PKCS12_DATA_free, [69](#)
 - PKI_X509_PKCS12_DATA_new, [69](#)
 - PKI_X509_PKCS12_free, [69](#)
 - PKI_X509_PKCS12_free_void, [69](#)
 - PKI_X509_PKCS12_get_cacert, [69](#)
 - PKI_X509_PKCS12_get_cert, [69](#)
 - PKI_X509_PKCS12_get_keypair, [69](#)
 - PKI_X509_PKCS12_get_otherCerts, [69](#)
 - PKI_X509_PKCS12_new, [69](#)
 - PKI_X509_PKCS12_new_null, [69](#)
 - PKI_X509_PKCS12_TOKEN_export, [69](#)
 - PKI_X509_PKCS12_verify_cred, [69](#)
- pki_x509_p12_io.c
 - PKI_X509_PKCS12_get, [17](#)
 - PKI_X509_PKCS12_get_mem, [17](#)
 - PKI_X509_PKCS12_get_url, [17](#)
 - PKI_X509_PKCS12_put, [17](#)
 - PKI_X509_PKCS12_put_mem, [17](#)
 - PKI_X509_PKCS12_put_url, [17](#)
 - PKI_X509_PKCS12_STACK_get, [17](#)
 - PKI_X509_PKCS12_STACK_get_mem, [17](#)
 - PKI_X509_PKCS12_STACK_get_url, [18](#)
 - PKI_X509_PKCS12_STACK_put, [18](#)
 - PKI_X509_PKCS12_STACK_put_mem, [18](#)
 - PKI_X509_PKCS12_STACK_put_url, [18](#)
- PKI_X509_PKCS12_DATA_add_certs
 - pki_x509_p12.c, [68](#)
- PKI_X509_PKCS12_DATA_add_keypair
 - pki_x509_p12.c, [68](#)
- PKI_X509_PKCS12_DATA_add_other_certs
 - pki_x509_p12.c, [68](#)
- PKI_X509_PKCS12_DATA_free
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_DATA_new
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_free
 - pki_x509_p12.c, [69](#)

- PKI_X509_PKCS12_free_void
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_get
 - pki_x509_p12_io.c, [17](#)
- PKI_X509_PKCS12_get_cacert
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_get_cert
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_get_keypair
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_get_mem
 - pki_x509_p12_io.c, [17](#)
- PKI_X509_PKCS12_get_otherCerts
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_get_url
 - pki_x509_p12_io.c, [17](#)
- PKI_X509_PKCS12_new
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_new_null
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_put
 - pki_x509_p12_io.c, [17](#)
- PKI_X509_PKCS12_put_mem
 - pki_x509_p12_io.c, [17](#)
- PKI_X509_PKCS12_put_url
 - pki_x509_p12_io.c, [17](#)
- PKI_X509_PKCS12_STACK_get
 - pki_x509_p12_io.c, [17](#)
- PKI_X509_PKCS12_STACK_get_mem
 - pki_x509_p12_io.c, [17](#)
- PKI_X509_PKCS12_STACK_get_url
 - pki_x509_p12_io.c, [18](#)
- PKI_X509_PKCS12_STACK_put
 - pki_x509_p12_io.c, [18](#)
- PKI_X509_PKCS12_STACK_put_mem
 - pki_x509_p12_io.c, [18](#)
- PKI_X509_PKCS12_STACK_put_url
 - pki_x509_p12_io.c, [18](#)
- PKI_X509_PKCS12_TOKEN_export
 - pki_x509_p12.c, [69](#)
- PKI_X509_PKCS12_verify_cred
 - pki_x509_p12.c, [69](#)
- pki_x509_pkcs7.c
 - PKI_X509_PKCS7_add_attribute, [72](#)
 - PKI_X509_PKCS7_add_cert, [72](#)
 - PKI_X509_PKCS7_add_cert_stack, [72](#)
 - PKI_X509_PKCS7_add_crl, [72](#)
 - PKI_X509_PKCS7_add_crl_stack, [72](#)
 - PKI_X509_PKCS7_add_recipient, [72](#)
 - PKI_X509_PKCS7_add_signed_attribute, [72](#)
 - PKI_X509_PKCS7_add_signer, [72](#)
 - PKI_X509_PKCS7_add_signer_tk, [72](#)
 - PKI_X509_PKCS7_clear_certs, [73](#)
 - PKI_X509_PKCS7_decode, [73](#)
 - PKI_X509_PKCS7_delete_attribute, [73](#)
 - PKI_X509_PKCS7_delete_signed_attribute, [73](#)
 - PKI_X509_PKCS7_encode, [73](#)
 - PKI_X509_PKCS7_free, [73](#)
 - PKI_X509_PKCS7_free_void, [73](#)
 - PKI_X509_PKCS7_get_attribute, [73](#)
 - PKI_X509_PKCS7_get_attribute_by_name, [73](#)
 - PKI_X509_PKCS7_get_cert, [73](#)
 - PKI_X509_PKCS7_get_certs_num, [73](#)
 - PKI_X509_PKCS7_get_crl, [73](#)
 - PKI_X509_PKCS7_get_crls_num, [73](#)
 - PKI_X509_PKCS7_get_data, [74](#)
 - PKI_X509_PKCS7_get_data_tk, [74](#)
 - PKI_X509_PKCS7_get_encode_alg, [74](#)
 - PKI_X509_PKCS7_get_raw_data, [74](#)
 - PKI_X509_PKCS7_get_recipient_cert, [74](#)
 - PKI_X509_PKCS7_get_recipient_info, [74](#)
 - PKI_X509_PKCS7_get_recipients_num, [74](#)
 - PKI_X509_PKCS7_get_signed_attribute, [74](#)
 - PKI_X509_PKCS7_get_signed_attribute_by_name, [74](#)
 - PKI_X509_PKCS7_get_signer_info, [74](#)
 - PKI_X509_PKCS7_get_signers_num, [74](#)
 - PKI_X509_PKCS7_get_type, [74](#)
 - PKI_X509_PKCS7_has_recipients, [74](#)
 - PKI_X509_PKCS7_has_signers, [75](#)
 - PKI_X509_PKCS7_new, [75](#)
 - PKI_X509_PKCS7_set_cipher, [75](#)
 - PKI_X509_PKCS7_set_recipients, [75](#)
 - PKI_X509_PKCS7_VALUE_print_bio, [75](#)
- PKI_X509_PKCS7_add_attribute
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_add_cert
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_add_cert_stack
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_add_crl
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_add_crl_stack
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_add_recipient
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_add_signed_attribute
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_add_signer
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_add_signer_tk
 - pki_x509_pkcs7.c, [72](#)
- PKI_X509_PKCS7_clear_certs
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_decode
 - pki_x509_pkcs7.c, [73](#)

- PKI_X509_PKCS7_delete_attribute
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_delete_signed_attribute
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_encode
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_free
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_free_void
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_get
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_get_attribute
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_get_attribute_by_name
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_get_cert
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_get_certs_num
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_get_crl
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_get_crls_num
 - pki_x509_pkcs7.c, [73](#)
- PKI_X509_PKCS7_get_data
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_data_tk
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_encode_alg
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_mem
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_get_raw_data
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_recipient_cert
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_recipient_info
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_recipients_num
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_signed_attribute
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_signed_attribute_by_name
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_signer_info
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_signers_num
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_type
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_get_url
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_has_recipients
 - pki_x509_pkcs7.c, [74](#)
- PKI_X509_PKCS7_has_signers
 - pki_x509_pkcs7.c, [75](#)
- pki_x509_pkcs7_io.c
 - PKI_X509_PKCS7_get, [19](#)
 - PKI_X509_PKCS7_get_mem, [19](#)
 - PKI_X509_PKCS7_get_url, [19](#)
 - PKI_X509_PKCS7_put, [19](#)
 - PKI_X509_PKCS7_put_mem, [19](#)
 - PKI_X509_PKCS7_put_url, [19](#)
 - PKI_X509_PKCS7_STACK_get, [19](#)
 - PKI_X509_PKCS7_STACK_get_mem, [19](#)
 - PKI_X509_PKCS7_STACK_get_url, [19](#)
 - PKI_X509_PKCS7_STACK_put, [19](#)
 - PKI_X509_PKCS7_STACK_put_mem, [19](#)
 - PKI_X509_PKCS7_STACK_put_url, [19](#)
- PKI_X509_PKCS7_new
 - pki_x509_pkcs7.c, [75](#)
- PKI_X509_PKCS7_put
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_put_mem
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_put_url
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_set_cipher
 - pki_x509_pkcs7.c, [75](#)
- PKI_X509_PKCS7_set_recipients
 - pki_x509_pkcs7.c, [75](#)
- PKI_X509_PKCS7_STACK_get
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_STACK_get_mem
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_STACK_get_url
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_STACK_put
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_STACK_put_mem
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_STACK_put_url
 - pki_x509_pkcs7_io.c, [19](#)
- PKI_X509_PKCS7_VALUE_print_bio
 - pki_x509_pkcs7.c, [75](#)
- PKI_X509_print_parsed
 - pki_x509.c, [110](#)
- PKI_X509_PROFILE_add_extension
 - profile.c, [112](#)
- PKI_X509_PROFILE_free
 - profile.c, [112](#)
- PKI_X509_PROFILE_free_void
 - profile.c, [112](#)
- PKI_X509_PROFILE_get_default
 - profile.c, [112](#)
- PKI_X509_PROFILE_get_ext_by_num
 - profile.c, [112](#)
- PKI_X509_PROFILE_get_extensions

- profile.c, [112](#)
- PKI_X509_PROFILE_get_exts_num
 - profile.c, [113](#)
- PKI_X509_PROFILE_get_name
 - profile.c, [113](#)
- PKI_X509_PROFILE_get_value
 - profile.c, [113](#)
- PKI_X509_PROFILE_load
 - profile.c, [113](#)
- PKI_X509_PROFILE_new
 - profile.c, [113](#)
- PKI_X509_PROFILE_put_file
 - profile.c, [113](#)
- PKI_X509_PRQP_NONCE_new
 - prqp_lib.c, [117](#)
- PKI_X509_PRQP_REQ_add_service
 - prqp_lib.c, [117](#)
- PKI_X509_PRQP_REQ_add_service_stack
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_free
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_free_void
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_get
 - prqp_req_io.c, [121](#)
- PKI_X509_PRQP_REQ_get_data
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_get_mem
 - prqp_req_io.c, [121](#)
- PKI_X509_PRQP_REQ_get_url
 - prqp_req_io.c, [121](#)
- PKI_X509_PRQP_REQ_new_certs_res
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_new_null
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_new_url
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_print
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_print_fp
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_put
 - prqp_req_io.c, [121](#)
- PKI_X509_PRQP_REQ_put_mem
 - prqp_req_io.c, [121](#)
- PKI_X509_PRQP_REQ_put_url
 - prqp_req_io.c, [121](#)
- PKI_X509_PRQP_REQ_STACK_get
 - prqp_req_io.c, [121](#)
- PKI_X509_PRQP_REQ_STACK_get_mem
 - prqp_req_io.c, [121](#)
- PKI_X509_PRQP_REQ_STACK_get_url
 - prqp_req_io.c, [122](#)
- PKI_X509_PRQP_REQ_STACK_put
 - prqp_req_io.c, [122](#)
- PKI_X509_PRQP_REQ_STACK_put_mem
 - prqp_req_io.c, [122](#)
- PKI_X509_PRQP_REQ_STACK_put_url
 - prqp_req_io.c, [122](#)
- PKI_X509_PRQP_REQ_VALUE_print_bio
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_REQ_verify
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_RESP_add_referrals
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_RESP_add_service
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_RESP_add_service_stack
 - prqp_lib.c, [118](#)
- PKI_X509_PRQP_RESP_free
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_free_void
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_get
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_get_data
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_get_http
 - http_client.c, [114](#)
- PKI_X509_PRQP_RESP_get_mem
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_get_status
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_get_url
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_new_null
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_new_req
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_nonce_dup
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_pkistatus_set
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_print
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_print_fp
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_put
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_put_mem
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_put_url
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_STACK_get
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_STACK_get_mem
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_STACK_get_url

- prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_STACK_put
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_STACK_put_mem
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_STACK_put_url
 - prqp_resp_io.c, [123](#)
- PKI_X509_PRQP_RESP_url_sk
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_VALUE_print_bio
 - prqp_lib.c, [119](#)
- PKI_X509_PRQP_RESP_verify
 - prqp_lib.c, [120](#)
- PKI_X509_PRQP_RESP_version_set
 - prqp_lib.c, [120](#)
- PKI_X509_PRQP_sign
 - prqp_lib.c, [120](#)
- PKI_X509_PRQP_sign_tk
 - prqp_lib.c, [120](#)
- PKI_X509_PRQP_STATUS_STRING
 - prqp_lib.c, [120](#)
- PKI_X509_PRQP_verify
 - prqp_lib.c, [120](#)
- PKI_X509_put
 - pki_x509_io.c, [15](#)
- PKI_X509_put_mem
 - pki_x509_mem.c, [111](#)
- PKI_X509_put_mem_value
 - pki_x509_mem.c, [111](#)
- PKI_X509_put_url
 - pki_x509_io.c, [15](#)
- PKI_X509_put_value
 - pki_x509_io.c, [16](#)
- pki_x509_req.c
 - PKI_X509_REQ_add_attribute, [76](#)
 - PKI_X509_REQ_add_extension, [76](#)
 - PKI_X509_REQ_add_extension_stack, [77](#)
 - PKI_X509_REQ_clear_attributes, [77](#)
 - PKI_X509_REQ_delete_attribute, [77](#)
 - PKI_X509_REQ_delete_attribute_by_name, [77](#)
 - PKI_X509_REQ_delete_attribute_by_num, [77](#)
 - PKI_X509_REQ_free, [77](#)
 - PKI_X509_REQ_free_void, [77](#)
 - PKI_X509_REQ_get_attribute, [77](#)
 - PKI_X509_REQ_get_attribute_by_name, [77](#)
 - PKI_X509_REQ_get_attribute_by_num, [77](#)
 - PKI_X509_REQ_get_attributes_num, [77](#)
 - PKI_X509_REQ_get_data, [77](#)
 - PKI_X509_REQ_get_extension_by_name, [78](#)
 - PKI_X509_REQ_get_extension_by_num, [78](#)
 - PKI_X509_REQ_get_extension_by_oid, [78](#)
 - PKI_X509_REQ_get_keysize, [78](#)
- PKI_X509_REQ_get_parsed, [78](#)
- PKI_X509_REQ_new, [78](#)
- PKI_X509_REQ_new_null, [78](#)
- PKI_X509_REQ_print_parsed, [78](#)
- PKI_X509_REQ_add_attribute
 - pki_x509_req.c, [76](#)
- PKI_X509_REQ_add_extension
 - pki_x509_req.c, [76](#)
- PKI_X509_REQ_add_extension_stack
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_clear_attributes
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_delete_attribute
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_delete_attribute_by_name
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_delete_attribute_by_num
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_free
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_free_void
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_get
 - pki_x509_req_io.c, [20](#)
- PKI_X509_REQ_get_attribute
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_get_attribute_by_name
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_get_attribute_by_num
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_get_attributes_num
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_get_data
 - pki_x509_req.c, [77](#)
- PKI_X509_REQ_get_extension_by_name
 - pki_x509_req.c, [78](#)
- PKI_X509_REQ_get_extension_by_num
 - pki_x509_req.c, [78](#)
- PKI_X509_REQ_get_extension_by_oid
 - pki_x509_req.c, [78](#)
- PKI_X509_REQ_get_keysize
 - pki_x509_req.c, [78](#)
- PKI_X509_REQ_get_mem
 - pki_x509_req_io.c, [20](#)
- PKI_X509_REQ_get_parsed
 - pki_x509_req.c, [78](#)
- PKI_X509_REQ_get_url
 - pki_x509_req_io.c, [20](#)
- pki_x509_req_io.c
 - PKI_X509_REQ_get, [20](#)
 - PKI_X509_REQ_get_mem, [20](#)
 - PKI_X509_REQ_get_url, [20](#)
 - PKI_X509_REQ_put, [20](#)
 - PKI_X509_REQ_put_mem, [20](#)

- PKI_X509_REQ_put_url, [20](#)
- PKI_X509_REQ_STACK_get, [20](#)
- PKI_X509_REQ_STACK_get_mem, [20](#)
- PKI_X509_REQ_STACK_get_url, [20](#)
- PKI_X509_REQ_STACK_put, [21](#)
- PKI_X509_REQ_STACK_put_mem, [21](#)
- PKI_X509_REQ_STACK_put_url, [21](#)
- PKI_X509_REQ_new
 - pki_x509_req.c, [78](#)
- PKI_X509_REQ_new_null
 - pki_x509_req.c, [78](#)
- PKI_X509_REQ_print_parsed
 - pki_x509_req.c, [78](#)
- PKI_X509_REQ_put
 - pki_x509_req_io.c, [20](#)
- PKI_X509_REQ_put_mem
 - pki_x509_req_io.c, [20](#)
- PKI_X509_REQ_put_url
 - pki_x509_req_io.c, [20](#)
- PKI_X509_REQ_STACK_get
 - pki_x509_req_io.c, [20](#)
- PKI_X509_REQ_STACK_get_mem
 - pki_x509_req_io.c, [20](#)
- PKI_X509_REQ_STACK_get_url
 - pki_x509_req_io.c, [20](#)
- PKI_X509_REQ_STACK_put
 - pki_x509_req_io.c, [21](#)
- PKI_X509_REQ_STACK_put_mem
 - pki_x509_req_io.c, [21](#)
- PKI_X509_REQ_STACK_put_url
 - pki_x509_req_io.c, [21](#)
- pki_x509_scep_attr.c
 - PKI_X509_SCEP_ATTRIBUTE_get_nid, [127](#)
 - PKI_X509_SCEP_ATTRIBUTE_get_txt, [127](#)
 - PKI_X509_SCEP_init, [127](#)
 - PKI_X509_SCEP_MSG_get_attr_value, [127](#)
 - PKI_X509_SCEP_MSG_get_attr_value_int, [127](#)
 - PKI_X509_SCEP_MSG_get_failinfo, [127](#)
 - PKI_X509_SCEP_MSG_get_oid, [127](#)
 - PKI_X509_SCEP_MSG_get_proxy, [127](#)
 - PKI_X509_SCEP_MSG_get_recipient_nonce, [127](#)
 - PKI_X509_SCEP_MSG_get_sender_nonce, [127](#)
 - PKI_X509_SCEP_MSG_get_status, [127](#)
 - PKI_X509_SCEP_MSG_get_trans_id, [128](#)
 - PKI_X509_SCEP_MSG_get_type, [128](#)
 - PKI_X509_SCEP_MSG_new_trans_id, [128](#)
 - PKI_X509_SCEP_MSG_set_attribute, [128](#)
 - PKI_X509_SCEP_MSG_set_attribute_by_name, [128](#)
 - PKI_X509_SCEP_MSG_set_attribute_int, [128](#)
 - PKI_X509_SCEP_MSG_set_failinfo, [128](#)
 - PKI_X509_SCEP_MSG_set_proxy, [128](#)
 - PKI_X509_SCEP_MSG_set_recipient_nonce, [128](#)
 - PKI_X509_SCEP_MSG_set_sender_nonce, [128](#)
 - PKI_X509_SCEP_MSG_set_status, [128](#)
 - PKI_X509_SCEP_MSG_set_trans_id, [128](#)
 - PKI_X509_SCEP_MSG_set_type, [129](#)
 - SCEP_ATTRIBUTE_list, [129](#)
 - SCEP_CONF_LIST_SIZE, [127](#)
- PKI_X509_SCEP_ATTRIBUTE_get_nid
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_ATTRIBUTE_get_txt
 - pki_x509_scep_attr.c, [127](#)
- pki_x509_scep_data.c
 - PKI_X509_SCEP_DATA_add_recipient, [130](#)
 - PKI_X509_SCEP_DATA_free, [130](#)
 - PKI_X509_SCEP_DATA_new, [130](#)
 - PKI_X509_SCEP_DATA_set_ias, [130](#)
 - PKI_X509_SCEP_DATA_set_raw_data, [130](#)
 - PKI_X509_SCEP_DATA_set_recipients, [130](#)
 - PKI_X509_SCEP_DATA_set_x509_obj, [130](#)
- PKI_X509_SCEP_DATA_add_recipient
 - pki_x509_scep_data.c, [130](#)
- PKI_X509_SCEP_DATA_free
 - pki_x509_scep_data.c, [130](#)
- PKI_X509_SCEP_DATA_new
 - pki_x509_scep_data.c, [130](#)
- PKI_X509_SCEP_DATA_set_ias
 - pki_x509_scep_data.c, [130](#)
- PKI_X509_SCEP_DATA_set_raw_data
 - pki_x509_scep_data.c, [130](#)
- PKI_X509_SCEP_DATA_set_recipients
 - pki_x509_scep_data.c, [130](#)
- PKI_X509_SCEP_DATA_set_x509_obj
 - pki_x509_scep_data.c, [130](#)
- PKI_X509_SCEP_init
 - pki_x509_scep_attr.c, [127](#)
- pki_x509_scep_msg.c
 - PKI_X509_SCEP_MSG_add_signer, [131](#)
 - PKI_X509_SCEP_MSG_add_signer_tk, [131](#)
 - PKI_X509_SCEP_MSG_decode, [131](#)
 - PKI_X509_SCEP_MSG_encode, [131](#)
 - PKI_X509_SCEP_MSG_free, [132](#)
 - PKI_X509_SCEP_MSG_new, [132](#)
 - PKI_X509_SCEP_MSG_new_certreq, [132](#)
- PKI_X509_SCEP_MSG_add_signer
 - pki_x509_scep_msg.c, [131](#)
- PKI_X509_SCEP_MSG_add_signer_tk
 - pki_x509_scep_msg.c, [131](#)
- PKI_X509_SCEP_MSG_decode
 - pki_x509_scep_msg.c, [131](#)
- PKI_X509_SCEP_MSG_encode

- pki_x509_scep_msg.c, [131](#)
- PKI_X509_SCEP_MSG_free
 - pki_x509_scep_msg.c, [132](#)
- PKI_X509_SCEP_MSG_get_attr_value
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_MSG_get_attr_value_int
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_MSG_get_failinfo
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_MSG_get_oid
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_MSG_get_proxy
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_MSG_get_recipient_nonce
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_MSG_get_sender_nonce
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_MSG_get_status
 - pki_x509_scep_attr.c, [127](#)
- PKI_X509_SCEP_MSG_get_trans_id
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_get_type
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_new
 - pki_x509_scep_msg.c, [132](#)
- PKI_X509_SCEP_MSG_new_certreq
 - pki_x509_scep_msg.c, [132](#)
- PKI_X509_SCEP_MSG_new_trans_id
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_attribute
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_attribute_by_name
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_attribute_int
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_failinfo
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_proxy
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_recipient_nonce
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_sender_nonce
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_status
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_trans_id
 - pki_x509_scep_attr.c, [128](#)
- PKI_X509_SCEP_MSG_set_type
 - pki_x509_scep_attr.c, [129](#)
- PKI_X509_set_hsm
 - pki_x509.c, [110](#)
- PKI_X509_set_reference
 - pki_x509.c, [110](#)
- PKI_X509_set_value
 - pki_x509.c, [110](#)
- pki_x509_signature.c
 - PKI_X509_SIGNATURE_get_parsed, [78](#)
- PKI_X509_SIGNATURE_get_parsed
 - pki_x509_signature.c, [78](#)
- PKI_X509_STACK_get
 - pki_x509_io.c, [16](#)
- PKI_X509_STACK_get_mem
 - pki_x509_mem.c, [111](#)
- PKI_X509_STACK_get_url
 - pki_x509_io.c, [16](#)
- PKI_X509_STACK_put
 - pki_x509_io.c, [16](#)
- PKI_X509_STACK_put_mem
 - pki_x509_mem.c, [111](#)
- PKI_X509_STACK_put_url
 - pki_x509_io.c, [16](#)
- pki_x509_xpair.c
 - PKI_X509_XPAIR_free, [79](#)
 - PKI_X509_XPAIR_free_void, [79](#)
 - PKI_X509_XPAIR_get_forward, [79](#)
 - PKI_X509_XPAIR_get_reverse, [79](#)
 - PKI_X509_XPAIR_new_certs, [79](#)
 - PKI_X509_XPAIR_new_null, [79](#)
 - PKI_X509_XPAIR_set_forward, [80](#)
 - PKI_X509_XPAIR_set_reverse, [80](#)
 - PKI_XPAIR_new_null, [80](#)
- PKI_X509_XPAIR_free
 - pki_x509_xpair.c, [79](#)
- PKI_X509_XPAIR_free_void
 - pki_x509_xpair.c, [79](#)
- PKI_X509_XPAIR_get
 - pki_x509_xpair_io.c, [22](#)
- PKI_X509_XPAIR_get_forward
 - pki_x509_xpair.c, [79](#)
- PKI_X509_XPAIR_get_reverse
 - pki_x509_xpair.c, [79](#)
- PKI_X509_XPAIR_get_url
 - pki_x509_xpair_io.c, [22](#)
- pki_x509_xpair_io.c
 - PKI_X509_XPAIR_get, [22](#)
 - PKI_X509_XPAIR_get_url, [22](#)
 - PKI_X509_XPAIR_put, [22](#)
 - PKI_X509_XPAIR_put_mem, [22](#)
 - PKI_X509_XPAIR_STACK_get, [22](#)
 - PKI_X509_XPAIR_STACK_get_mem, [22](#)
 - PKI_X509_XPAIR_STACK_get_url, [22](#)
 - PKI_X509_XPAIR_STACK_put, [22](#)
 - PKI_X509_XPAIR_STACK_put_mem, [23](#)
 - PKI_X509_XPAIR_STACK_put_url, [23](#)
 - PKI_XPAIR_print, [23](#)
- PKI_X509_XPAIR_new_certs
 - pki_x509_xpair.c, [79](#)
- PKI_X509_XPAIR_new_null

- pki_x509_xpair.c, 79
- PKI_X509_XPAIR_put
 - pki_x509_xpair_io.c, 22
- PKI_X509_XPAIR_put_mem
 - pki_x509_xpair_io.c, 22
- PKI_X509_XPAIR_set_forward
 - pki_x509_xpair.c, 80
- PKI_X509_XPAIR_set_reverse
 - pki_x509_xpair.c, 80
- PKI_X509_XPAIR_STACK_get
 - pki_x509_xpair_io.c, 22
- PKI_X509_XPAIR_STACK_get_mem
 - pki_x509_xpair_io.c, 22
- PKI_X509_XPAIR_STACK_get_url
 - pki_x509_xpair_io.c, 22
- PKI_X509_XPAIR_STACK_put
 - pki_x509_xpair_io.c, 22
- PKI_X509_XPAIR_STACK_put_mem
 - pki_x509_xpair_io.c, 23
- PKI_X509_XPAIR_STACK_put_url
 - pki_x509_xpair_io.c, 23
- PKI_XPAIR_new_null
 - pki_x509_xpair.c, 80
- PKI_XPAIR_print
 - pki_x509_xpair_io.c, 23
- PKI_ZFree
 - pki_mem.c, 94
- PKI_ZFree_str
 - pki_mem.c, 94
- profile.c
 - PKI_X509_PROFILE_add_extension, 112
 - PKI_X509_PROFILE_free, 112
 - PKI_X509_PROFILE_free_void, 112
 - PKI_X509_PROFILE_get_default, 112
 - PKI_X509_PROFILE_get_ext_by_num, 112
 - PKI_X509_PROFILE_get_extensions, 112
 - PKI_X509_PROFILE_get_exts_num, 113
 - PKI_X509_PROFILE_get_name, 113
 - PKI_X509_PROFILE_get_value, 113
 - PKI_X509_PROFILE_load, 113
 - PKI_X509_PROFILE_new, 113
 - PKI_X509_PROFILE_put_file, 113
- proto_list
 - url.c, 35
- prqp_bio.c
 - d2i_PRQP_REQ_bio, 114
 - d2i_PRQP_RESP_bio, 114
 - i2d_PRQP_REQ_bio, 114
 - i2d_PRQP_RESP_bio, 114
 - PEM_read_bio_PRQP_REQ, 114
 - PEM_read_bio_PRQP_RESP, 114
 - PEM_write_bio_PRQP_REQ, 114
 - PEM_write_bio_PRQP_RESP, 114
- PRQP_init_all_services
- prqp_lib.c, 120
- prqp_lib.c
 - __PKI_PRQP_LIB_C__, 117
 - CERT_IDENTIFIER_cmp, 117
 - PKI_PRQP_CERTID_new, 117
 - PKI_PRQP_CERTID_new_cert, 117
 - PKI_PRQP_REQ_new_cert, 117
 - PKI_X509_PRQP_NONCE_new, 117
 - PKI_X509_PRQP_REQ_add_service, 117
 - PKI_X509_PRQP_REQ_add_service_stack, 118
 - PKI_X509_PRQP_REQ_free, 118
 - PKI_X509_PRQP_REQ_free_void, 118
 - PKI_X509_PRQP_REQ_get_data, 118
 - PKI_X509_PRQP_REQ_new_certs_res, 118
 - PKI_X509_PRQP_REQ_new_null, 118
 - PKI_X509_PRQP_REQ_new_url, 118
 - PKI_X509_PRQP_REQ_print, 118
 - PKI_X509_PRQP_REQ_print_fp, 118
 - PKI_X509_PRQP_REQ_VALUE_print_bio, 118
 - PKI_X509_PRQP_REQ_verify, 118
 - PKI_X509_PRQP_RESP_add_referrals, 118
 - PKI_X509_PRQP_RESP_add_service, 118
 - PKI_X509_PRQP_RESP_add_service_stack, 118
 - PKI_X509_PRQP_RESP_free, 119
 - PKI_X509_PRQP_RESP_free_void, 119
 - PKI_X509_PRQP_RESP_get_data, 119
 - PKI_X509_PRQP_RESP_get_status, 119
 - PKI_X509_PRQP_RESP_new_null, 119
 - PKI_X509_PRQP_RESP_new_req, 119
 - PKI_X509_PRQP_RESP_nonce_dup, 119
 - PKI_X509_PRQP_RESP_pkistatus_set, 119
 - PKI_X509_PRQP_RESP_print, 119
 - PKI_X509_PRQP_RESP_print_fp, 119
 - PKI_X509_PRQP_RESP_url_sk, 119
 - PKI_X509_PRQP_RESP_VALUE_print_bio, 119
 - PKI_X509_PRQP_RESP_verify, 120
 - PKI_X509_PRQP_RESP_version_set, 120
 - PKI_X509_PRQP_sign, 120
 - PKI_X509_PRQP_sign_tk, 120
 - PKI_X509_PRQP_STATUS_STRING, 120
 - PKI_X509_PRQP_verify, 120
 - PRQP_init_all_services, 120
 - PRQP_RESOURCE_RESPONSE_TOKEN_get_oid, 120
 - PRQP_RESOURCE_RESPONSE_TOKEN_get_services, 120
- prqp_req_io.c
 - PKI_X509_PRQP_REQ_get, 121
 - PKI_X509_PRQP_REQ_get_mem, 121
 - PKI_X509_PRQP_REQ_get_url, 121

- PKI_X509_PRQP_REQ_put, [121](#)
- PKI_X509_PRQP_REQ_put_mem, [121](#)
- PKI_X509_PRQP_REQ_put_url, [121](#)
- PKI_X509_PRQP_REQ_STACK_get, [121](#)
- PKI_X509_PRQP_REQ_STACK_get_mem, [121](#)
- PKI_X509_PRQP_REQ_STACK_get_url, [122](#)
- PKI_X509_PRQP_REQ_STACK_put, [122](#)
- PKI_X509_PRQP_REQ_STACK_put_mem, [122](#)
- PKI_X509_PRQP_REQ_STACK_put_url, [122](#)
- PRQP_RESOURCE_RESPONSE_TOKEN_get_oid
 - prqp_lib.c, [120](#)
- PRQP_RESOURCE_RESPONSE_TOKEN_get_services
 - prqp_lib.c, [120](#)
- prqp_resp_io.c
 - PKI_X509_PRQP_RESP_get, [123](#)
 - PKI_X509_PRQP_RESP_get_mem, [123](#)
 - PKI_X509_PRQP_RESP_get_url, [123](#)
 - PKI_X509_PRQP_RESP_put, [123](#)
 - PKI_X509_PRQP_RESP_put_mem, [123](#)
 - PKI_X509_PRQP_RESP_put_url, [123](#)
 - PKI_X509_PRQP_RESP_STACK_get, [123](#)
 - PKI_X509_PRQP_RESP_STACK_get_mem, [123](#)
 - PKI_X509_PRQP_RESP_STACK_get_url, [123](#)
 - PKI_X509_PRQP_RESP_STACK_put, [123](#)
 - PKI_X509_PRQP_RESP_STACK_put_mem, [123](#)
 - PKI_X509_PRQP_RESP_STACK_put_url, [123](#)
- prqp_srv.c
 - PKI_DISCOVER_get_resp, [124](#)
 - PKI_DISCOVER_get_resp_url, [124](#)
 - PKI_get_ca_resources, [124](#)
 - PKI_get_ca_service, [124](#)
 - PKI_get_ca_service_sk, [124](#)
- pthread_init.c
 - _dyn_create_callback, [81](#)
 - _dyn_destroy_callback, [81](#)
 - _dyn_lock_callback, [81](#)
 - CRYPTO_LOCK, [81](#)
 - CRYPTO_READ, [81](#)
 - CRYPTO_UNLOCK, [81](#)
 - CRYPTO_WRITE, [81](#)
 - lock_cs, [81](#)
 - thread_cleanup, [81](#)
 - win32_locking_callback, [81](#)
- SCEP_ATTRIBUTE_list
 - pki_x509_scep_attr.c, [129](#)
- SCEP_CONF_LIST_SIZE
 - pki_x509_scep_attr.c, [127](#)
- sock.c
 - _Accept, [29](#)
 - _Close, [29](#)
 - _Connect, [29](#)
 - _Listen, [29](#)
 - _Read, [29](#)
 - _Select, [29](#)
 - _Shutdown, [29](#)
 - _Socket, [29](#)
 - _Write, [29](#)
 - Gethostbyname, [29](#)
 - h_errno, [30](#)
 - inet_close, [29](#)
 - inet_connect, [29](#)
 - LISTENQ, [29](#)
 - PKI_NET_accept, [29](#)
 - PKI_NET_close, [29](#)
 - PKI_NET_get_data, [30](#)
 - PKI_NET_listen, [30](#)
 - PKI_NET_open, [30](#)
 - PKI_NET_read, [30](#)
 - PKI_NET_socket, [30](#)
 - PKI_NET_write, [30](#)
- src/cms/asn1.c, [1](#)
- src/cms/cms_cert_req.c, [1](#)
- src/cms/cms_simple.c, [3](#)
- src/extensions.c, [3](#)
- src/io/pki_keypair_io.c, [4](#)
- src/io/pki_msg_req_io.c, [5](#)
- src/io/pki_msg_resp_io.c, [6](#)
- src/io/pki_ocsp_req_io.c, [7](#)
- src/io/pki_ocsp_resp_io.c, [8](#)
- src/io/pki_x509_cert_io.c, [10](#)
- src/io/pki_x509_crl_io.c, [12](#)
- src/io/pki_x509_io.c, [14](#)
- src/io/pki_x509_p12_io.c, [16](#)
- src/io/pki_x509_pkcs7_io.c, [18](#)
- src/io/pki_x509_req_io.c, [19](#)
- src/io/pki_x509_xpair_io.c, [21](#)
- src/net/http_s.c, [23](#)
- src/net/ldap.c, [25](#)
- src/net/mysql.c, [26](#)
- src/net/pg.c, [26](#)
- src/net/pkcs11.c, [27](#)
- src/net/sock.c, [28](#)
- src/net/ssl.c, [30](#)
- src/net/url.c, [33](#)
- src/openssl/pki_algor.c, [36](#)
- src/openssl/pki_digest.c, [39](#)
- src/openssl/pki_id.c, [40](#)

- src/openssl/pki_integer.c, 41
- src/openssl/pki_keypair.c, 43
- src/openssl/pki_ocsp_req.c, 44
- src/openssl/pki_ocsp_resp.c, 48
- src/openssl/pki_oid.c, 51
- src/openssl/pki_string.c, 52
- src/openssl/pki_time.c, 54
- src/openssl/pki_x509_attribute.c, 55
- src/openssl/pki_x509_cert.c, 57
- src/openssl/pki_x509_crl.c, 61
- src/openssl/pki_x509_extension.c, 64
- src/openssl/pki_x509_name.c, 65
- src/openssl/pki_x509_p12.c, 67
- src/openssl/pki_x509_pkcs7.c, 70
- src/openssl/pki_x509_req.c, 75
- src/openssl/pki_x509_signature.c, 78
- src/openssl/pki_x509_xpair.c, 79
- src/openssl/pki_x509_xpair_asn1.c, 80
- src/openssl/pthread_init.c, 80
- src/pki_config.c, 81
- src/pki_cred.c, 86
- src/pki_err.c, 87
- src/pki_init.c, 88
- src/pki_log.c, 90
- src/pki_mem.c, 91
- src/pki_msg_req.c, 94
- src/pki_msg_resp.c, 99
- src/pki_threads.c, 103
- src/pki_threads_vars.c, 104
- src/pki_x509.c, 107
- src/pki_x509_mem.c, 110
- src/pki_x509_mime.c, 111
- src/profile.c, 112
- src/prqp/asn1_req.c, 113
- src/prqp/asn1_res.c, 113
- src/prqp/http_client.c, 113
- src/prqp/prqp_bio.c, 114
- src/prqp/prqp_lib.c, 115
- src/prqp/prqp_req_io.c, 120
- src/prqp/prqp_resp_io.c, 122
- src/prqp/prqp_srv.c, 123
- src/scep/pki_x509_scep_asn1.c, 125
- src/scep/pki_x509_scep_attr.c, 125
- src/scep/pki_x509_scep_data.c, 129
- src/scep/pki_x509_scep_msg.c, 131
- src/stack.c, 132
- src/support.c, 134
- src/token.c, 135
- src/token_data.c, 146
- src/token_id.c, 147
- ssl.c
 - BUFF_MAX_SIZE, 31
 - PKI_SSL_close, 31
 - PKI_SSL_connect, 31
 - PKI_SSL_connect_url, 32
 - PKI_SSL_free, 32
 - PKI_SSL_get_peer_cert, 32
 - PKI_SSL_get_peer_chain, 32
 - PKI_SSL_get_servername, 32
 - PKI_SSL_new, 32
 - PKI_SSL_read, 32
 - PKI_SSL_set_algor, 32
 - PKI_SSL_set_cipher, 32
 - PKI_SSL_set_flags, 32
 - PKI_SSL_set_token, 32
 - PKI_SSL_set_trusted, 32
 - PKI_SSL_write, 32
- stack.c
 - PKI_STACK_del_num, 133
 - PKI_STACK_elements, 133
 - PKI_STACK_free, 133
 - PKI_STACK_free_all, 133
 - PKI_STACK_get_num, 133
 - PKI_STACK_ins_num, 134
 - PKI_STACK_new, 134
 - PKI_STACK_new_null, 134
 - PKI_STACK_new_type, 134
 - PKI_STACK_pop, 134
 - PKI_STACK_pop_free, 134
 - PKI_STACK_push, 134
- strcmp_nocase
 - support.c, 135
- strncmp_nocase
 - support.c, 135
- strstr_nocase
 - support.c, 135
- support.c
 - get_env_string, 135
 - PKI_get_env, 135
 - PKI_set_env, 135
 - strcmp_nocase, 135
 - strncmp_nocase, 135
 - strstr_nocase, 135
- thread_cleanup
 - pthread_init.c, 81
- token.c
 - PKI_TOKEN_add_profile, 139
 - PKI_TOKEN_check, 139
 - PKI_TOKEN_cred_cb_env, 139
 - PKI_TOKEN_cred_cb_stdin, 139
 - PKI_TOKEN_cred_get, 139
 - PKI_TOKEN_cred_set_cb, 139
 - PKI_TOKEN_del_url, 139
 - PKI_TOKEN_export_cert, 139
 - PKI_TOKEN_export_keypair, 139
 - PKI_TOKEN_export_keypair_url, 139
 - PKI_TOKEN_export_otherCerts, 140

- PKI_TOKEN_export_p12, [140](#)
- PKI_TOKEN_export_req, [140](#)
- PKI_TOKEN_export_trustedCerts, [140](#)
- PKI_TOKEN_free, [140](#)
- PKI_TOKEN_free_void, [140](#)
- PKI_TOKEN_get_cacert, [140](#)
- PKI_TOKEN_get_cert, [140](#)
- PKI_TOKEN_get_config_dir, [140](#)
- PKI_TOKEN_get_cred, [140](#)
- PKI_TOKEN_get_crls, [141](#)
- PKI_TOKEN_get_keypair, [141](#)
- PKI_TOKEN_get_name, [141](#)
- PKI_TOKEN_get_otherCerts, [141](#)
- PKI_TOKEN_get_p12, [141](#)
- PKI_TOKEN_get_trustedCerts, [141](#)
- PKI_TOKEN_import_cert, [141](#)
- PKI_TOKEN_import_cert_stack, [141](#)
- PKI_TOKEN_import_keypair, [142](#)
- PKI_TOKEN_init, [142](#)
- PKI_TOKEN_issue_cert, [142](#)
- PKI_TOKEN_issue_crl, [142](#)
- PKI_TOKEN_issue_proxy, [142](#)
- PKI_TOKEN_load_cacert, [142](#)
- PKI_TOKEN_load_cert, [142](#)
- PKI_TOKEN_load_config, [142](#)
- PKI_TOKEN_load_crls, [143](#)
- PKI_TOKEN_load_keypair, [143](#)
- PKI_TOKEN_load_otherCerts, [143](#)
- PKI_TOKEN_load_profiles, [143](#)
- PKI_TOKEN_load_req, [143](#)
- PKI_TOKEN_load_trustedCerts, [143](#)
- PKI_TOKEN_login, [143](#)
- PKI_TOKEN_new, [143](#)
- PKI_TOKEN_new_keypair, [144](#)
- PKI_TOKEN_new_keypair_url, [144](#)
- PKI_TOKEN_new_null, [144](#)
- PKI_TOKEN_new_p12, [144](#)
- PKI_TOKEN_new_req, [144](#)
- PKI_TOKEN_OID_new, [144](#)
- PKI_TOKEN_print_info, [144](#)
- PKI_TOKEN_search_profile, [144](#)
- PKI_TOKEN_self_sign, [144](#)
- PKI_TOKEN_set_algor, [144](#)
- PKI_TOKEN_set_algor_by_name, [145](#)
- PKI_TOKEN_set_cacert, [145](#)
- PKI_TOKEN_set_cert, [145](#)
- PKI_TOKEN_set_config_dir, [145](#)
- PKI_TOKEN_set_cred, [145](#)
- PKI_TOKEN_set_crls, [145](#)
- PKI_TOKEN_set_keypair, [146](#)
- PKI_TOKEN_set_otherCerts, [146](#)
- PKI_TOKEN_set_req, [146](#)
- PKI_TOKEN_set_trustedCerts, [146](#)
- PKI_TOKEN_use_slot, [146](#)
- token_data.c
 - PKI_TOKEN_get_cacert_data, [147](#)
 - PKI_TOKEN_get_cert_data, [147](#)
 - PKI_TOKEN_get_identity_data, [147](#)
 - PKI_TOKEN_get_keypair_data, [147](#)
 - PKI_TOKEN_get_otherCerts_data, [147](#)
 - PKI_TOKEN_get_privkey_data, [147](#)
 - PKI_TOKEN_get_pubkey_data, [147](#)
 - PKI_TOKEN_get_trustedCerts_data, [147](#)
- token_id.c
 - PKI_TOKEN_ID_INFO_get, [148](#)
 - PKI_TOKEN_ID_INFO_list, [148](#)
 - PKI_TOKEN_ID_num, [148](#)
 - PKI_TOKEN_ID_set, [148](#)
- URI_PROTOCOL
 - url.c, [34](#)
- url.c
 - BUFF_MAX_SIZE, [34](#)
 - proto_list, [35](#)
 - URI_PROTOCOL, [34](#)
 - URL_free, [34](#)
 - URL_get_data, [34](#)
 - URL_get_data_fd, [34](#)
 - URL_get_data_file, [34](#)
 - URL_get_data_url, [34](#)
 - URL_get_local_addr, [35](#)
 - URL_new, [35](#)
 - URL_proto_to_string, [35](#)
 - URL_put_data, [35](#)
 - URL_put_data_fd, [35](#)
 - URL_put_data_file, [35](#)
 - URL_put_data_url, [35](#)
- URL_free
 - url.c, [34](#)
- URL_get_data
 - url.c, [34](#)
- URL_get_data_fd
 - url.c, [34](#)
- URL_get_data_file
 - url.c, [34](#)
- URL_get_data_http_s
 - http_s.c, [25](#)
- URL_get_data_http_s_url
 - http_s.c, [25](#)
- URL_get_data_mysql
 - mysql.c, [26](#)
- URL_get_data_mysql_url
 - mysql.c, [26](#)
- URL_get_data_pg
 - pg.c, [27](#)
- URL_get_data_pg_url
 - pg.c, [27](#)
- URL_get_data_pkcs11

- pkcs11.c, [27](#)
- URL_get_data_pkcs11_url
 - pkcs11.c, [28](#)
- URL_get_data_url
 - url.c, [34](#)
- URL_get_local_addr
 - url.c, [35](#)
- URL_new
 - url.c, [35](#)
- URL_post_data_http_s
 - http_s.c, [25](#)
- URL_post_data_http_s_url
 - http_s.c, [25](#)
- URL_proto_to_string
 - url.c, [35](#)
- URL_put_data
 - url.c, [35](#)
- URL_put_data_fd
 - url.c, [35](#)
- URL_put_data_file
 - url.c, [35](#)
- URL_put_data_http_s
 - http_s.c, [25](#)
- URL_put_data_mysql
 - mysql.c, [26](#)
- URL_put_data_mysql_url
 - mysql.c, [26](#)
- URL_put_data_pg
 - pg.c, [27](#)
- URL_put_data_pg_url
 - pg.c, [27](#)
- URL_put_data_url
 - url.c, [35](#)
- win32_locking_callback
 - pthread_init.c, [81](#)
- xmlErrorPtr
 - pki_config.c, [83](#)