

# libpki Reference Manual

## 0.5.1

Generated by Doxygen 1.5.1

Thu Feb 10 19:39:50 2011

## Contents

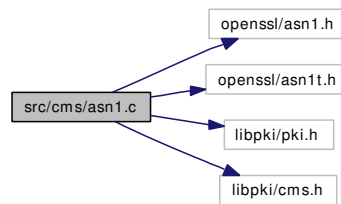
### 1 libpki File Documentation

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](#)

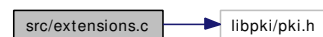
1.3.1.2 typedef void [PKI\\_CMS\\_RESP](#)

### 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, PKI\_TOKEN \*tk)
- int [PKI\\_X509\\_EXTENSIONS\\_crl\\_add\\_profile](#) (PKI\_X509\_PROFILE \*conf, PKI\_CONFIG \*oids, PKI\_X509\_CRL \*crl, PKI\_TOKEN \*tk)  
*Adds extensions to a CRL according to the profile passed as argument.*
- int [PKI\\_X509\\_EXTENSIONS\\_req\\_add\\_profile](#) (PKI\_X509\_PROFILE \*conf, PKI\_CONFIG \*oids, PKI\_X509\_REQ \*req, PKI\_TOKEN \*tk)

### 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, PKI_TOKEN * tk)`

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

Adds extensions to a CRL according to the profile passed as argument.

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

## 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)

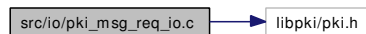
Returns a STACK of PKI\_X509\_KEYPAIR objects from a url address.

#### 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.

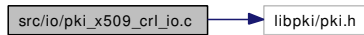
### 1.10.1.10 `int` `PKI_X509_CERT_STACK_put` (`PKI_X509_CERT_STACK *` `sk`, `PKI_DATA_FORMAT` `format`, `char *` `url_s`, `char *` `mime`, `PKI_CRED *` `cred`, `HSM *` `hsm`)

### 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.

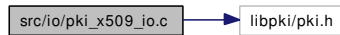
**1.11.1.10** `int PKI_X509_CRL_STACK_put (PKI_X509_CRL_STACK *sk, PKI_DATA_FORMAT format, char *url_s, PKI_CRED *cred, HSM *hsm)`

**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)`

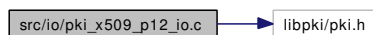
Puts a stack of PKI\_X509 objects to the url passed as a string.

**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)`

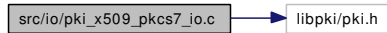
**1.13.1.10** `int PKI_X509_PKCS12_STACK_put (PKI_X509_PKCS12_STACK * sk, PKI_DATA_FORMAT format, char * url_s, char * mime, PKI_CRED * cred, HSM * hsm)`

**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

- `void` [PKI\\_HTTP\\_free](#) (PKI\_HTTP \**rv*)  
*Frees the memory associated with a PKI\_HTTP data structure.*
- `int` [PKI\\_HTTP\\_GET\\_data](#) (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` [PKI\\_HTTP\\_GET\\_data\\_socket](#) (PKI\_SOCKET \**sock*, int *timeout*, size\_t *max\_size*, PKI\_MEM\_STACK \*\**ret*)  
*Returns HTTP data from a PKI\_SOCKET by using the GET command.*
- `int` [PKI\\_HTTP\\_GET\\_data\\_url](#) (URL \**url*, int *timeout*, size\_t *max\_size*, PKI\_MEM\_STACK \*\**ret*, PKI\_SSL \**ssl*)  
*Returns the data from an HTTP URL by using the GET command.*
- `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\\_message](#) (PKI\_SOCKET \**sock*, int *timeout*, size\_t *max\_size*)
- `int` [PKI\\_HTTP\\_get\\_socket](#) (PKI\_SOCKET \**sock*, char \**data*, size\_t *data\_size*, char \**content\_type*, int *method*, int *timeout*, size\_t *max\_size*, PKI\_MEM\_STACK \*\**sk*)



*Reads a data from an HTTP server.*

- int [PKI\\_HTTP\\_get\\_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)

*Sends a HTTP message to a URL and retrieve the response.*

- PKI\_HTTP \* [PKI\\_HTTP\\_new](#) (void)

*Allocates the memory for a new PKI\_HTTP data structure.*

- int [PKI\\_HTTP\\_POST\\_data](#) (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 [PKI\\_HTTP\\_POST\\_data\\_socket](#) (PKI\_SOCKET \*sock, char \*data, size\_t size, char \*content\_type, int timeout, size\_t max\_size, PKI\_MEM\_STACK \*\*ret\_sk)

*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 [PKI\\_HTTP\\_POST\\_data\\_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)

*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.*

### 1.17.1 Define Documentation

#### 1.17.1.1 #define BUFF\_MAX\_SIZE 2048

### 1.17.2 Function Documentation

#### 1.17.2.1 void PKI\_HTTP\_free (PKI\_HTTP \*rv)

Frees the memory associated with a PKI\_HTTP data structure.

#### 1.17.2.2 int PKI\_HTTP\_GET\_data (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.

#### 1.17.2.3 int PKI\_HTTP\_GET\_data\_socket (PKI\_SOCKET \* sock, int timeout, size\_t max\_size, PKI\_MEM\_STACK \*\* ret)

Returns HTTP data from a PKI\_SOCKET by using the GET command.

#### 1.17.2.4 int PKI\_HTTP\_GET\_data\_url (URL \* url, int timeout, size\_t max\_size, PKI\_MEM\_STACK \*\* ret, PKI\_SSL \* ssl)

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

**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\_message (PKI\_SOCKET \* *sock*, int *timeout*, size\_t *max\_size*)****1.17.2.8 int PKI\_HTTP\_get\_socket (PKI\_SOCKET \* *sock*, char \* *data*, size\_t *data\_size*, char \* *content\_type*, int *method*, int *timeout*, size\_t *max\_size*, PKI\_MEM\_STACK \*\* *sk*)**

Reads a data from an HTTP server.

**1.17.2.9 int PKI\_HTTP\_get\_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*)**

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.

**1.17.2.10 PKI\_HTTP\* PKI\_HTTP\_new (void)**

Allocates the memory for a new PKI\_HTTP data structure.

**1.17.2.11 int PKI\_HTTP\_POST\_data (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.

**1.17.2.12 int PKI\_HTTP\_POST\_data\_socket (PKI\_SOCKET \* *sock*, char \* *data*, size\_t *size*, char \* *content\_type*, int *timeout*, size\_t *max\_size*, PKI\_MEM\_STACK \*\* *ret\_sk*)**

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 PKI\_HTTP\_POST\_data\_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*)**

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.

**1.18 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/pki\_socket.c File Reference

Include dependency graph for `pki_socket.c`:



### Functions

- `int PKI\_SOCKET\_close (PKI_SOCKET *sock)`  
*Opens a Connection to a URL via an already initialized PKI\_SOCKET Closes a connected socket.*
- `int PKI\_SOCKET\_connect (PKI_SOCKET *sock, URL *url, int timeout)`  
*Opens a Connection to a URL via an already initialized PKI\_SOCKET.*
- `int PKI\_SOCKET\_connect\_ssl (PKI_SOCKET *sock, URL *url, int timeout)`  
*Opens a Secure Connection to a URL via an already initialized PKI\_SOCKET.*
- `void PKI\_SOCKET\_free (PKI_SOCKET *sock)`  
*Creates a new PKI\_SOCKET from an existing file descriptor Frees memory associated with a PKI\_SOCKET data structure.*
- `int PKI\_SOCKET\_get\_fd (PKI_SOCKET *sock)`  
*Returns the underlying file descriptor (if present).*
- `PKI_SSL * PKI\_SOCKET\_get\_ssl (PKI_SOCKET *sock)`  
*Returns the PKI\_SSL layer (if present).*
- `URL * PKI\_SOCKET\_get\_url (PKI_SOCKET *sock)`  
*Returns the URL used in PKI\_SOCKET\_open or PKI\_SOCKET\_open\_url.*
- `PKI_SOCKET * PKI\_SOCKET\_new ()`  
*Allocates and Inizializes a new PKI\_SOCKET.*
- `PKI_SOCKET * PKI\_SOCKET\_new\_ssl (PKI_SSL *ssl)`  
*Creates a new PKI\_SOCKET from an existing PKI\_SSL.*
- `int PKI\_SOCKET\_open (PKI_SOCKET *sock, char *url_s, int timeout)`  
*Opens a connection to the passed url.*
- `int PKI\_SOCKET\_open\_url (PKI_SOCKET *sock, URL *url, int timeout)`  
*Opens a connection to the server identified by URL.*
- `ssize_t PKI\_SOCKET\_read (PKI_SOCKET *sock, char *buf, size_t n, int timeout)`

*Reads n bytes from a connected socket.*

- int [PKI\\_SOCKET\\_set\\_fd](#) (PKI\_SOCKET \*sock, int fd)  
*Sets an already connected fd layer in an existing PKI\_SOCKET.*
- int [PKI\\_SOCKET\\_set\\_ssl](#) (PKI\_SOCKET \*sock, PKI\_SSL \*ssl)  
*Sets an already connected PKI\_SSL layer in an existing PKI\_SOCKET.*
- int [PKI\\_SOCKET\\_start\\_ssl](#) (PKI\_SOCKET \*sock)  
*Starts an SSL/TLS session on a connected FD socket.*
- ssize\_t [PKI\\_SOCKET\\_write](#) (PKI\_SOCKET \*sock, char \*buf, size\_t n)  
*Writes n bytes to a connected socket.*

### 1.22.1 Function Documentation

#### 1.22.1.1 int PKI\_SOCKET\_close (PKI\_SOCKET \* sock)

Opens a Connection to a URL via an already initialized PKI\_SOCKET Closes a connected socket.

#### 1.22.1.2 int PKI\_SOCKET\_connect (PKI\_SOCKET \* sock, URL \* url, int timeout)

Opens a Connection to a URL via an already initialized PKI\_SOCKET.

#### 1.22.1.3 int PKI\_SOCKET\_connect\_ssl (PKI\_SOCKET \* sock, URL \* url, int timeout)

Opens a Secure Connection to a URL via an already initialized PKI\_SOCKET.

#### 1.22.1.4 void PKI\_SOCKET\_free (PKI\_SOCKET \* sock)

Creates a new PKI\_SOCKET from an existing file descriptor Frees memory associated with a PKI\_SOCKET data structure.

#### 1.22.1.5 int PKI\_SOCKET\_get\_fd (PKI\_SOCKET \* sock)

Returns the underlying file descriptor (if present).

#### 1.22.1.6 PKI\_SSL\* PKI\_SOCKET\_get\_ssl (PKI\_SOCKET \* sock)

Returns the PKI\_SSL layer (if present).

#### 1.22.1.7 URL\* PKI\_SOCKET\_get\_url (PKI\_SOCKET \* sock)

Returns the URL used in PKI\_SOCKET\_open or PKI\_SOCKET\_open\_url.

#### 1.22.1.8 PKI\_SOCKET\* PKI\_SOCKET\_new ()

Allocates and Initializes a new PKI\_SOCKET.

**1.22.1.9** `PKI_SOCKET* PKI_SOCKET_new_ssl (PKI_SSL * ssl)`

Creates a new PKI\_SOCKET from an existing PKI\_SSL.

**1.22.1.10** `int PKI_SOCKET_open (PKI_SOCKET * sock, char * url_s, int timeout)`

Opens a connection to the passed url.

**1.22.1.11** `int PKI_SOCKET_open_url (PKI_SOCKET * sock, URL * url, int timeout)`

Opens a connection to the server identified by URL.

**1.22.1.12** `ssize_t PKI_SOCKET_read (PKI_SOCKET * sock, char * buf, size_t n, int timeout)`

Reads n bytes from a connected socket.

**1.22.1.13** `int PKI_SOCKET_set_fd (PKI_SOCKET * sock, int fd)`

Sets an already connected fd layer in an existing PKI\_SOCKET.

**1.22.1.14** `int PKI_SOCKET_set_ssl (PKI_SOCKET * sock, PKI_SSL * ssl)`

Sets an already connected PKI\_SSL layer in an existing PKI\_SOCKET.

**1.22.1.15** `int PKI_SOCKET_start_ssl (PKI_SOCKET * sock)`

Starts an SSL/TLS session on a connected FD socket.

**1.22.1.16** `ssize_t PKI_SOCKET_write (PKI_SOCKET * sock, char * buf, size_t n)`

Writes n bytes to a connected socket.

**1.23** src/net/sock.c File Reference

Include dependency graph for sock.c:

**Defines**

- #define [LISTENQ](#) 30

**Functions**

- [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\\_errno](#)

### 1.23.1 Define Documentation

#### 1.23.1.1 #define LISTENQ 30

### 1.23.2 Function Documentation

#### 1.23.2.1 int \_Close (int fd)

#### 1.23.2.2 int \_Connect (int sockfd, const SA \*srvaddr, socklen\_t addrlen)

#### 1.23.2.3 int \_Listen (char \*hostname, int port)

#### 1.23.2.4 ssize\_t \_Read (int fd, void \*bufptr, size\_t nbytes)



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

**1.23.2.6** `void _Shutdown (int fd, int howto)`

**1.23.2.7** `int _Socket (int family, int type, int protocol)`

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

**1.23.2.9** `struct hostent* Gethostbyname (const char * hostname)`

**1.23.2.10** `int inet_close (int fd)`

**1.23.2.11** `int inet_connect (URL * url)`

**1.23.2.12** `int PKI_NET_accept (int sock, int timeout)`

Returns the connected socket as a result of an Accept.

**1.23.2.13** `int PKI_NET_close (int sock)`

Closes the connection to an open connection to an host.

**1.23.2.14** `PKI_MEM* PKI_NET_get_data (int fd, int timeout, size_t max_size)`

Returns data read from a socket.

**1.23.2.15** `int PKI_NET_listen (char * host, int port)`

Returns a reference to a listen socket.

**1.23.2.16** `int PKI_NET_open (URL * url, int timeout)`

Connects to an host and returns the connected socket.

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

Reads n-bytes of data from a socket.

**1.23.2.18** `int PKI_NET_socket (int family, int type, int protocol)`

Returns a reference to a new socket (int).

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

Writes n bytes of data to a socket.

### 1.23.3 Variable Documentation

#### 1.23.3.1 int [h\\_errno](#)

## 1.24 src/net/ssl.c File Reference

Include dependency graph for ssl.c:



### Defines

- `#define BUFF\_MAX\_SIZE 2048`

### Functions

- int [\\_\\_pki\\_ssl\\_start\\_ssl](#) (PKI\_SSL \*ssl)
- int [PKI\\_SSL\\_check\\_verify](#) (PKI\_SSL \*ssl, PKI\_SSL\_VERIFY flag)  
*Checks if a verify flag has been set.*
- 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.*
- PKI\_SSL \* [PKI\\_SSL\\_dup](#) (PKI\_SSL \*ssl)
- void [PKI\\_SSL\\_free](#) (PKI\_SSL \*ssl)  
*Frees memory associated with a PKI\_SSL.*
- int [PKI\\_SSL\\_get\\_fd](#) (PKI\_SSL \*ssl)  
*Returns the underlying socket descriptor.*
- 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\\_fd](#) (PKI\_SSL \*ssl, int fd)  
*Sets the underlying socket descriptor.*
- int [PKI\\_SSL\\_set\\_flags](#) (PKI\_SSL \*ssl, PKI\_SSL\_FLAGS 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.*
- int [PKI\\_SSL\\_set\\_verify](#) (PKI\_SSL \*ssl, PKI\_SSL\_VERIFY vflags)  
*Sets the verify flags to be used when validating cert chain.*
- int [PKI\\_SSL\\_start\\_ssl](#) (PKI\_SSL \*ssl, int fd)  
*Initiates an SSL connection over an already connected socket.*
- ssize\_t [PKI\\_SSL\\_write](#) (PKI\_SSL \*ssl, char \*buf, ssize\_t size)  
*Writes data to a connected PKI\_SSL.*

### 1.24.1 Define Documentation

#### 1.24.1.1 #define BUFF\_MAX\_SIZE 2048

### 1.24.2 Function Documentation

#### 1.24.2.1 int \_\_pki\_ssl\_start\_ssl (PKI\_SSL \*ssl)

#### 1.24.2.2 int PKI\_SSL\_check\_verify (PKI\_SSL \*ssl, PKI\_SSL\_VERIFY flag)

Checks if a verify flag has been set.

#### 1.24.2.3 int PKI\_SSL\_close (PKI\_SSL \*ssl)

Closes an SSL connection.

#### 1.24.2.4 int PKI\_SSL\_connect (PKI\_SSL \*ssl, char \*url\_s, int timeout)

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

**1.24.2.5 int PKI\_SSL\_connect\_url (PKI\_SSL \* *ssl*, URL \* *url*, int *timeout*)**

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

**1.24.2.6 PKI\_SSL\* PKI\_SSL\_dup (PKI\_SSL \* *ssl*)****1.24.2.7 void PKI\_SSL\_free (PKI\_SSL \* *ssl*)**

Frees memory associated with a PKI\_SSL.

**1.24.2.8 int PKI\_SSL\_get\_fd (PKI\_SSL \* *ssl*)**

Returns the underlying socket descriptor.

**1.24.2.9 struct pki\_x509\_st\* PKI\_SSL\_get\_peer\_cert (PKI\_SSL \* *ssl*)**

Returns the Peer certificate used in a connected PKI\_SSL.

**1.24.2.10 PKI\_X509\_CERT\_STACK\* PKI\_SSL\_get\_peer\_chain (PKI\_SSL \* *ssl*)**

Returns the peer certificate chain as a new PKI\_X509\_CERT\_STACK.

**1.24.2.11 const char\* PKI\_SSL\_get\_servername (PKI\_SSL \* *ssl*)**

Returns the extension value provided in Client Hello or NULL.

**1.24.2.12 PKI\_SSL\* PKI\_SSL\_new (PKI\_SSL\_ALGOR \* *algor*)**

Sets the options for a new PKI\_SSL object.

**1.24.2.13 ssize\_t PKI\_SSL\_read (PKI\_SSL \* *ssl*, char \* *buf*, ssize\_t *size*)**

Reads data from a connected PKI\_SSL.

**1.24.2.14 int PKI\_SSL\_set\_algor (PKI\_SSL \* *ssl*, PKI\_SSL\_ALGOR \* *algor*)**

Sets the protocol for a new PKI\_SSL object.

**1.24.2.15 int PKI\_SSL\_set\_cipher (PKI\_SSL \* *ssl*, char \* *cipher*)**

Sets the Chiphers to be used.

**1.24.2.16 int PKI\_SSL\_set\_fd (PKI\_SSL \* *ssl*, int *fd*)**

Sets the underlying socket descriptor.

**1.24.2.17 int PKI\_SSL\_set\_flags (PKI\_SSL \* *ssl*, PKI\_SSL\_FLAGS *flags*)**

Sets the SSL connection flags.

**1.24.2.18 int PKI\_SSL\_set\_token (PKI\_SSL \*ssl, struct pki\_token\_st \*tk)**

Sets the PKI\_TOKEN to be used for the SSL connection.

**1.24.2.19 int PKI\_SSL\_set\_trusted (PKI\_SSL \*ssl, PKI\_X509\_CERT\_STACK \*sk)**

Sets the list of trusted certificates for SSL connections.

**1.24.2.20 int PKI\_SSL\_set\_verify (PKI\_SSL \*ssl, PKI\_SSL\_VERIFY vflags)**

Sets the verify flags to be used when validating cert chain.

**1.24.2.21 int PKI\_SSL\_start\_ssl (PKI\_SSL \*ssl, int fd)**

Initiates an SSL connection over an already connected socket.

**1.24.2.22 ssize\_t PKI\_SSL\_write (PKI\_SSL \*ssl, char \*buf, ssize\_t size)**

Writes data to a connected PKI\_SSL.

**1.25 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\\_socket](#) (PKI\_SOCKET \*sock, int timeout, ssize\_t size)

*Returns a `PKI_MEM_STACK` filled with data from a `PKI_SOCKET`.*

- `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_socket` (`PKI_SOCKET *sock`, `PKI_MEM *data`, `char *contType`, `PKI_MEM_STACK **ret_sk`, `int timeout`, `ssize_t max_size`)
- `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` []

## 1.25.1 Define Documentation

### 1.25.1.1 #define BUFF\_MAX\_SIZE 2048

## 1.25.2 Typedef Documentation

### 1.25.2.1 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.

## 1.25.3 Function Documentation

### 1.25.3.1 void URL\_free (URL \* url)

Releases the Memory associated to a `URL` data structure.

### 1.25.3.2 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.

**1.25.3.3 PKI\_MEM\_STACK\* URL\_get\_data\_fd (URL \* url, ssize\_t size)****1.25.3.4 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.

**1.25.3.5 PKI\_MEM\_STACK\* URL\_get\_data\_socket (PKI\_SOCKET \* sock, int timeout, ssize\_t size)**

Returns a PKI\_MEM\_STACK filled with data from a PKI\_SOCKET.

Returns a PKI\_MEM\_STACK object filled with the data retrieved from a connected PKI\_SOCKET. 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.

**1.25.3.6 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.

**1.25.3.7 char\* URL\_get\_local\_addr (void)****1.25.3.8 URL\* URL\_new (char \* url\_s)****1.25.3.9 const char\* URL\_proto\_to\_string (URI\_PROTO proto)****1.25.3.10 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.

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

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

1.25.3.13 `int URL_put_data_socket (PKI_SOCKET * sock, PKI_MEM * data, char * contType, PKI_MEM_STACK ** ret_sk, int timeout, ssize_t max_size)`

1.25.3.14 `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)`

## 1.25.4 Variable Documentation

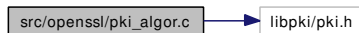
### 1.25.4.1 `URI_PROTOCOL proto_list[]`

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 }
}
```

## 1.26 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` (const 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](#) (`const 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](#) [ ]

## 1.26.1 Function Documentation

### 1.26.1.1 `PKI_ALGOR*` [PKI\\_ALGOR\\_get](#) (`PKI_ALGOR_ID algor`)

Build a `PKI_ALGOR` structure from its ID.

### 1.26.1.2 `PKI_ALGOR*` [PKI\\_ALGOR\\_get\\_by\\_name](#) (`const 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".

### 1.26.1.3 `PKI_DIGEST_ALG*` [PKI\\_ALGOR\\_get\\_digest](#) (`PKI_ALGOR *algor`)

Get the Digest Algorithm from the passed `PKI_ALGOR`.

**1.26.1.4 PKI\_ALGOR\_ID PKI\_ALGOR\_get\_digest\_id (PKI\_ALGOR \* *algor*)**

Returns the PKI\_ALGOR\_ID of the digest used in the PKI\_ALGOR.

**1.26.1.5 PKI\_ALGOR\_ID PKI\_ALGOR\_get\_id (PKI\_ALGOR \* *algor*)****1.26.1.6 const char\* PKI\_ALGOR\_get\_parsed (PKI\_ALGOR \* *algor*)**

Returns a text representation of the algorithm identifier.

**1.26.1.7 PKI\_SCHEME\_ID PKI\_ALGOR\_get\_scheme (PKI\_ALGOR \* *algor*)**

Returns the PKI\_SCHEME\_ID (signature scheme ID) of the algorithm.

**1.26.1.8 char\* PKI\_ALGOR\_ID\_txt (PKI\_ALGOR\_ID *algor*)**

Returns a text string with the algorithm identifier.

**1.26.1.9 PKI\_ALGOR\_ID\* PKI\_ALGOR\_list (PKI\_SCHEME\_ID *scheme*)****1.26.1.10 size\_t PKI\_ALGOR\_list\_size (PKI\_ALGOR\_ID \* *list*)****1.26.1.11 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.

**1.26.1.12 PKI\_DIGEST\_ALG\* PKI\_DIGEST\_ALG\_get\_by\_key (PKI\_X509\_KEYPAIR \* *pkey*)**

Returns the digest algorithm based on the key.

**1.26.1.13 PKI\_DIGEST\_ALG\* PKI\_DIGEST\_ALG\_get\_by\_name (const 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.

**1.26.1.14 const char\* PKI\_DIGEST\_ALG\_get\_parsed (PKI\_DIGEST\_ALG \* *alg*)**

Returns the string representation of a digest algorithm.

**1.26.1.15 PKI\_ALGOR\_ID\* PKI\_DIGEST\_ALG\_list (void)**

## 1.26.2 Variable Documentation

### 1.26.2.1 PKI\_ALGOR\_ID [PKI\\_ALGOR\\_LIST\\_DSA](#) []

**Initial value:**

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

### 1.26.2.2 PKI\_ALGOR\_ID [PKI\\_ALGOR\\_LIST\\_ECDSA](#) []

**Initial value:**

```
{
    PKI_ALGOR_UNKNOWN
}
```

### 1.26.2.3 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
}
```

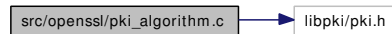
### 1.26.2.4 PKI\_ALGOR\_ID [PKI\\_DIGEST\\_ALG\\_LIST](#) []

**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
}
```

## 1.27 src/openssl/pki\_algorithm.c File Reference

Include dependency graph for pki\_algorithm.c:



### Functions

- void [PKI\\_ALGORITHM\\_free](#) (PKI\_ALGORITHM \*a)  
*Frees memory associated with a PKI\_ALGORITHM data structure.*
- PKI\_ALGORITHM \* [PKI\\_ALGORITHM\\_new](#) ()  
*Returns an empty PKI\_ALGORITHM data structure.*
- PKI\_ALGORITHM \* [PKI\\_ALGORITHM\\_new\\_digest](#) (PKI\_DIGEST\_ALG \*alg)  
*Returns a PKI\_ALGORITHM initialized with digest algorithm.*
- PKI\_ALGORITHM \* [PKI\\_ALGORITHM\\_new\\_type](#) (int type)  
*Returns a PKI\_ALGORITHM initialized with provided algor id.*

### 1.27.1 Function Documentation

#### 1.27.1.1 void PKI\_ALGORITHM\_free (PKI\_ALGORITHM \* a)

Frees memory associated with a PKI\_ALGORITHM data structure.

#### 1.27.1.2 PKI\_ALGORITHM\* PKI\_ALGORITHM\_new ()

Returns an empty PKI\_ALGORITHM data structure.

#### 1.27.1.3 PKI\_ALGORITHM\* PKI\_ALGORITHM\_new\_digest (PKI\_DIGEST\_ALG \* alg)

Returns a PKI\_ALGORITHM initialized with digest algorithm.

#### 1.27.1.4 PKI\_ALGORITHM\* PKI\_ALGORITHM\_new\_type (int type)

Returns a PKI\_ALGORITHM initialized with provided algor id.

## 1.28 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)

### 1.28.1 Function Documentation

#### 1.28.1.1 void PKI\_DIGEST\_free (PKI\_DIGEST \* data)

Free the memory associated with a PKI\_DIGEST data structure.

#### 1.28.1.2 char\* PKI\_DIGEST\_get\_parsed (PKI\_DIGEST \* digest)

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

#### 1.28.1.3 size\_t PKI\_DIGEST\_get\_size (PKI\_DIGEST\_ALG \* alg)

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

#### 1.28.1.4 PKI\_DIGEST\* PKI\_DIGEST\_MEM\_new (PKI\_DIGEST\_ALG \* alg, PKI\_MEM \* data)

Calculates a digest over data contained in a PKI\_MEM.

#### 1.28.1.5 PKI\_DIGEST\* PKI\_DIGEST\_MEM\_new\_by\_name (char \* alg\_name, PKI\_MEM \* data)

#### 1.28.1.6 PKI\_DIGEST\* PKI\_DIGEST\_new (PKI\_DIGEST\_ALG \* alg, unsigned char \* data, size\_t size)

Calculate digest over data provided in a buffer.

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

Calculates a digest over data buffer.

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

Calculate the digest of data retrieved via a URL.

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

## 1.29 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`)

### 1.29.1 Function Documentation

#### 1.29.1.1 `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.

#### 1.29.1.2 `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`

#### 1.29.1.3 `const char*` `PKI_ID_get_txt` (`PKI_ID id`)

## 1.30 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).*

### 1.30.1 Function Documentation

#### 1.30.1.1 `int` [PKI\\_INTEGER\\_cmp](#) (`PKI_INTEGER *a`, `PKI_INTEGER *b`)

Compare two PKI\_INTEGERs.

#### 1.30.1.2 `PKI_INTEGER*` [PKI\\_INTEGER\\_dup](#) (`PKI_INTEGER *a`)

Duplicates a PKI\_INTEGER data structure.

#### 1.30.1.3 `int` [PKI\\_INTEGER\\_free](#) (`PKI_INTEGER *i`)

Frees memory associated with a PKI\_INTEGER.

#### 1.30.1.4 `void` [PKI\\_INTEGER\\_free\\_void](#) (`void *i`)

**1.30.1.5 char\* PKI\_INTEGER\_get\_parsed (PKI\_INTEGER \* i)**

Returns a string representation of the PKI\_INTEGER.

**1.30.1.6 PKI\_INTEGER\* PKI\_INTEGER\_new (long long val)**

Returns a PKI\_INTEGER object starting from a long long value.

**1.30.1.7 PKI\_INTEGER\* PKI\_INTEGER\_new\_bin (unsigned char \* data, size\_t size)**

Generate a new PKI\_INTEGER from raw bit data.

**1.30.1.8 PKI\_INTEGER\* PKI\_INTEGER\_new\_char (char \* val)**

Returns a PKI\_INTEGER object from a string.

**1.30.1.9 int PKI\_INTEGER\_print (PKI\_INTEGER \* s)**

Prints the contents of a PKI\_INTEGER to Standard Output.

**1.30.1.10 int PKI\_INTEGER\_print\_fp (FILE \* fp, PKI\_INTEGER \* s)**

Prints the contents of a PKI\_INTEGER to a FILE pointer (eg., stdout).

**1.31 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\_ALGOR \* [PKI\\_X509\\_KEYPAIR\\_get\\_algor](#) (PKI\_X509\_KEYPAIR \*k)  
*Returns the default signing algorithm from a keypair.*
- 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\_SCHEME\_ID [PKI\\_X509\\_KEYPAIR\\_get\\_scheme](#) (PKI\_X509\_KEYPAIR \*k)  
*Returns the signing scheme from a keypair.*
- int [PKI\\_X509\\_KEYPAIR\\_get\\_size](#) (PKI\_X509\_KEYPAIR \*k)  
*Returns the size (in bits) of a pubkey.*



- `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_kp` (PKI\_KEYPARAMS \*kp, 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_X509_KEYPAIR * PKI_X509_KEYPAIR_new_url_kp` (PKI\_KEYPARAMS \*kp, 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_ALGOR * PKI_X509_KEYPAIR_VALUE_get_algor` (PKI\_X509\_KEYPAIR\_VALUE \*pVal)  
*Returns the default signing algorithm from a keypair value.*
- `PKI_SCHEME_ID PKI_X509_KEYPAIR_VALUE_get_scheme` (PKI\_X509\_KEYPAIR\_VALUE \*pVal)  
*Returns the signing scheme from a keypair value.*
- `int PKI_X509_KEYPAIR_VALUE_get_size` (PKI\_X509\_KEYPAIR\_VALUE \*pKey)  
*Returns the size (in bits) of a pubkey value.*
- `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.*

### 1.31.1 Function Documentation

**1.31.1.1 void PKI\_X509\_KEYPAIR\_free (PKI\_X509\_KEYPAIR \* key)**

**1.31.1.2 void PKI\_X509\_KEYPAIR\_free\_void (void \* key)**

**1.31.1.3 PKI\_ALGOR\* PKI\_X509\_KEYPAIR\_get\_algor (PKI\_X509\_KEYPAIR \* k)**

Returns the default signing algorithm from a keypair.

**1.31.1.4 PKI\_MEM\* PKI\_X509\_KEYPAIR\_get\_p8 (PKI\_X509\_KEYPAIR \* k)**

Returns the passed PKI\_X509\_KEYPAIR in PKCS#8 format.

**1.31.1.5 char\* PKI\_X509\_KEYPAIR\_get\_parsed (PKI\_X509\_KEYPAIR \* pkey)**

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

**1.31.1.6 PKI\_SCHEME\_ID PKI\_X509\_KEYPAIR\_get\_scheme (PKI\_X509\_KEYPAIR \* k)**

Returns the signing scheme from a keypair.

**1.31.1.7 int PKI\_X509\_KEYPAIR\_get\_size (PKI\_X509\_KEYPAIR \* k)**

Returns the size (in bits) of a pubkey.

**1.31.1.8 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.

**1.31.1.9 PKI\_X509\_KEYPAIR\* PKI\_X509\_KEYPAIR\_new\_kp (PKI\_KEYPARAMS \* kp, char \* label, PKI\_CRED \* cred, HSM \* hsm)**

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

**1.31.1.10 PKI\_X509\_KEYPAIR\* PKI\_X509\_KEYPAIR\_new\_null ()****1.31.1.11 PKI\_X509\_KEYPAIR\* PKI\_X509\_KEYPAIR\_new\_p8 (PKI\_MEM \* buf)**

Reads a PKI\_X509\_KEYPAIR from a PKCS#8 format.

**1.31.1.12 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.

**1.31.1.13 PKI\_X509\_KEYPAIR\* PKI\_X509\_KEYPAIR\_new\_url\_kp (PKI\_KEYPARAMS \* kp, URL \* url, PKI\_CRED \* cred, HSM \* hsm)**

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

**1.31.1.14 PKI\_DIGEST\* PKI\_X509\_KEYPAIR\_pub\_digest (PKI\_X509\_KEYPAIR \* k, PKI\_DIGEST\_ALG \* md)**

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

**1.31.1.15 PKI\_ALGOR\* PKI\_X509\_KEYPAIR\_VALUE\_get\_algor (PKI\_X509\_KEYPAIR\_VALUE \* pVal)**

Returns the default signing algorithm from a keypair value.

**1.31.1.16** `PKI_SCHEME_ID PKI_X509_KEYPAIR_VALUE_get_scheme (PKI_X509_KEYPAIR_VALUE *pVal)`

Returns the signing scheme from a keypair value.

**1.31.1.17** `int PKI_X509_KEYPAIR_VALUE_get_size (PKI_X509_KEYPAIR_VALUE *pKey)`

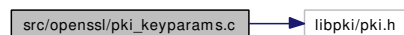
Returns the size (in bits) of a pubkey value.

**1.31.1.18** `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.

## 1.32 src/openssl/pki\_keyparams.c File Reference

Include dependency graph for pki\_keyparams.c:



### Functions

- void [PKI\\_KEYPARAMS\\_free](#) (PKI\_KEYPARAMS \*kp)  
*Frees the memory associated with a PKI\_KEYPARAMS structure.*
- PKI\_SCHEME\_ID [PKI\\_KEYPARAMS\\_get\\_type](#) (PKI\_KEYPARAMS \*kp)  
*Returns the type (PKI\_SCHEME\_ID) of the PKI\_KEYPARAMS.*
- PKI\_KEYPARAMS \* [PKI\\_KEYPARAMS\\_new](#) (int scheme, PKI\_X509\_PROFILE \*prof)  
*Allocates memory for a new PKI\_KEYPARAMS (for key of type 'scheme').*

### 1.32.1 Function Documentation

**1.32.1.1** `void PKI_KEYPARAMS_free (PKI_KEYPARAMS *kp)`

Frees the memory associated with a PKI\_KEYPARAMS structure.

**1.32.1.2** `PKI_SCHEME_ID PKI_KEYPARAMS_get_type (PKI_KEYPARAMS *kp)`

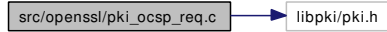
Returns the type (PKI\_SCHEME\_ID) of the PKI\_KEYPARAMS.

**1.32.1.3** `PKI_KEYPARAMS* PKI_KEYPARAMS_new (int scheme, PKI_X509_PROFILE *prof)`

Allocates memory for a new PKI\_KEYPARAMS (for key of type 'scheme').

## 1.33 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.*

### 1.33.1 Function Documentation

**1.33.1.1** PKI\_X509\_OCSP\_REQ\_VALUE\* [d2i\\_OCSP\\_REQ\\_bio](#) (PKI\_IO \* bp, PKI\_X509\_OCSP\_REQ\_VALUE \* p)

**1.33.1.2** int [i2d\\_OCSP\\_REQ\\_bio](#) (PKI\_IO \* bp, PKI\_X509\_OCSP\_REQ\_VALUE \* o)

**1.33.1.3** PKI\_X509\_OCSP\_REQ\_VALUE\* [PEM\\_read\\_bio\\_OCSP\\_REQ](#) (PKI\_IO \* bp, void \* a, void \* b, void \* c)

**1.33.1.4** int [PEM\\_write\\_bio\\_OCSP\\_REQ](#) (PKI\_IO \* bp, PKI\_X509\_OCSP\_REQ\_VALUE \* o)

**1.33.1.5** int [PKI\\_OCSP\\_nonce\\_check](#) (PKI\_X509\_OCSP\_REQ \* req, PKI\_X509\_OCSP\_RESP \* resp)

Checks the NONCE between a Request and a Response.

**1.33.1.6** `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.

**1.33.1.7** `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.

**1.33.1.8** `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.

**1.33.1.9** `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.

**1.33.1.10** `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.

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

**1.33.1.12** `int PKI_X509_OCSP_REQ_elements (PKI_X509_OCSP_REQ * req)`

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

**1.33.1.13** `void PKI_X509_OCSP_REQ_free (PKI_X509_OCSP_REQ * x)`

Frees the memory associated with a PKI\_X509\_OCSP\_REQ object.

**1.33.1.14** `void PKI_X509_OCSP_REQ_free_void (void * x)`

**1.33.1.15** `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.

**1.33.1.16** `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.

**1.33.1.17** `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.

**1.33.1.18** `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.

**1.33.1.19** `PKI_X509_OCSP_REQ* PKI_X509_OCSP_REQ_new (void)`

Generates an empty OCSP request.

**1.33.1.20** `PKI_X509_OCSP_REQ* PKI_X509_OCSP_REQ_new_null (void)`

**1.33.1.21** `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.

**1.33.1.22** `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)`

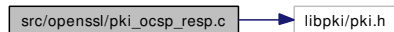
Signs a PKI\_X509\_OCSP\_REQ, for a simpler API use PKI\_X509\_OCSP\_REQ\_sign\_tk.

**1.33.1.23** `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.

## 1.34 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 \*otherCerts, 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.*

### 1.34.1 Function Documentation

#### 1.34.1.1 PKI\_OCSP\_RESP\* d2i\_PKI\_OCSP\_RESP\_bio (PKI\_IO \*bp, PKI\_OCSP\_RESP \*\*p)

#### 1.34.1.2 int i2d\_PKI\_OCSP\_RESP\_bio (PKI\_IO \*bp, PKI\_OCSP\_RESP \*o)

#### 1.34.1.3 PKI\_OCSP\_RESP\* PEM\_read\_bio\_PKI\_OCSP\_RESP (PKI\_IO \*bp, void \*a, void \*b, void \*c)



**1.34.1.4** `int PEM_write_bio_PKI_OCSP_RESP (PKI_IO * bp, PKI_OCSP_RESP * o)`

**1.34.1.5** `void PKI_OCSP_RESP_free (PKI_OCSP_RESP * x)`

**1.34.1.6** `PKI_OCSP_RESP* PKI_OCSP_RESP_new (void)`

Generates an empty OCSP request.

**1.34.1.7** `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.

**1.34.1.8** `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.

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

**1.34.1.10** `void PKI_X509_OCSP_RESP_free (PKI_X509_OCSP_RESP * x)`

Frees the memory associated with a PKI\_X509\_OCSP\_RESP object.

**1.34.1.11** `void PKI_X509_OCSP_RESP_free_void (void * x)`

**1.34.1.12** `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.

**1.34.1.13** `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.

**1.34.1.14** `PKI_X509_OCSP_RESP* PKI_X509_OCSP_RESP_new (void)`

**1.34.1.15** `PKI_X509_OCSP_RESP* PKI_X509_OCSP_RESP_new_null (void)`

**1.34.1.16** `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.

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

Sets the status of the request.

**1.34.1.18** `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.

**1.34.1.19** `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.

## 1.35 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_CONFIG * PKI_OID_load (char *uri)`
- `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.*

### 1.35.1 Function Documentation

#### 1.35.1.1 `int PKI_OID_cmp (PKI_OID * a, PKI_OID * b)`

Compares two PKI\_OID and returns 0 if they match.

#### 1.35.1.2 `PKI_OID* PKI_OID_dup (PKI_OID * a)`

Returns a duplicate of the passed PKI\_OID structure.

#### 1.35.1.3 `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.

#### 1.35.1.4 `void PKI_OID_free_void (void * buf)`

#### 1.35.1.5 `PKI_OID* PKI_OID_get (char * name)`

See `PKI_OID_new_text`.

#### 1.35.1.6 `const char* PKI_OID_get_descr (PKI_OID * a)`

Return the description associated with a PKI\_OID object.

#### 1.35.1.7 `PKI_ID PKI_OID_get_id (PKI_OID * a)`

Returns the PKI\_ID of the object if recognized.

#### 1.35.1.8 `char* PKI_OID_get_str (PKI_OID * a)`

Returns a new allocated string representation of an OID.

#### 1.35.1.9 `PKI_CONFIG* PKI_OID_load (char * uri)`

#### 1.35.1.10 `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

**1.35.1.11 PKI\_OID\* PKI\_OID\_new\_id (PKI\_ID id)**

Returns the OID associated with a PKI\_ID.

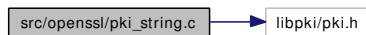
**1.35.1.12 PKI\_OID\* PKI\_OID\_new\_text (char \* name)**

Retrieve a pointer to an OID.

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

**1.36 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.*
- PKI\_DIGEST \* [PKI\\_STRING\\_get\\_digest](#) (PKI\_STRING \*s, PKI\_DIGEST\_ALG \*digest)  
*Returns the digest calculated on the string value.*
- 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.*

### 1.36.1 Function Documentation

#### 1.36.1.1 **PKI\_STRING\*** **PKI\_STRING\_dup** (**PKI\_STRING** \* *a*)

Duplicates a PKI\_STRING data structure.

#### 1.36.1.2 **void** **PKI\_STRING\_free** (**PKI\_STRING** \* *s*)

Releases the memory associated with a PKI\_STRING.

#### 1.36.1.3 **PKI\_DIGEST\*** **PKI\_STRING\_get\_digest** (**PKI\_STRING** \* *s*, **PKI\_DIGEST\_ALG** \* *digest*)

Returns the digest calculated on the string value.

#### 1.36.1.4 **char\*** **PKI\_STRING\_get\_parsed** (**PKI\_STRING** \* *s*)

Returns the parsed value (char \*) of the PKI\_STRING (in UTF-8).

#### 1.36.1.5 **int** **PKI\_STRING\_get\_type** (**PKI\_STRING** \* *s*)

Returns the type of the PKI\_STRING (PKI\_STRING\_IA5, etc.).

#### 1.36.1.6 **char\*** **PKI\_STRING\_get\_utf8** (**PKI\_STRING** \* *s*)

Returns the parsed value (char \*) of the PKI\_STRING (in UTF-8).

#### 1.36.1.7 **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.

#### 1.36.1.8 **PKI\_STRING\*** **PKI\_STRING\_new\_null** (**int** *type*)

Create an empty PKI\_STRING of type passed as the only argument.

#### 1.36.1.9 **int** **PKI\_STRING\_print** (**PKI\_STRING** \* *s*)

Prints the contents of a PKI\_STRING to Standard Output.

#### 1.36.1.10 **int** **PKI\_STRING\_print\_fp** (**FILE** \* *fp*, **PKI\_STRING** \* *s*)

Prints the contents of a PKI\_STRING to a FILE pointer (eg., stdout).

#### 1.36.1.11 **int** **PKI\_STRING\_set** (**PKI\_STRING** \* *s*, **char** \* *content*, **ssize\_t** *size*)

Sets the content of a PKI\_STRING.

## 1.37 src/openssl/pki\_time.c File Reference

Include dependency graph for pki\_time.c:



## Functions

- `int PKI_TIME_adj (PKI_TIME *time, long long offset)`  
*Adjusts the time by adding/subtracting the offset seconds from current value.*
- `PKI_TIME * PKI_TIME_dup (PKI_TIME *time)`  
*Returns a duplicate of the PKI\_TIME object.*
- `int PKI_TIME_free (PKI_TIME *time)`  
*Frees memory associated with a PKI\_TIME.*
- `void PKI_TIME_free_void (void *time)`
- `char * PKI_TIME_get_parsed (PKI_TIME *t)`  
*Returns a Human readable version of a PKI\_TIME.*
- `PKI_TIME * PKI_TIME_new (long long offset)`  
*Returns a new PKI\_TIME with offset (secs) from current time.*
- `int PKI_TIME_print (PKI_TIME *time)`  
*Prints a PKI\_TIME to standard output.*
- `int PKI_TIME_print_fp (FILE *fp, PKI_TIME *time)`  
*Prints out a PKI\_TIME to a FILE stream.*

### 1.37.1 Function Documentation

#### 1.37.1.1 `int PKI_TIME_adj (PKI_TIME *time, long long offset)`

Adjusts the time by adding/subtracting the offset seconds from current value.

#### 1.37.1.2 `PKI_TIME* PKI_TIME_dup (PKI_TIME *time)`

Returns a duplicate of the PKI\_TIME object.

#### 1.37.1.3 `int PKI_TIME_free (PKI_TIME *time)`

Frees memory associated with a PKI\_TIME.

#### 1.37.1.4 `void PKI_TIME_free_void (void *time)`

#### 1.37.1.5 `char* PKI_TIME_get_parsed (PKI_TIME *t)`

Returns a Human readable version of a PKI\_TIME.

**1.37.1.6** `PKI_TIME*` `PKI_TIME_new` (long long *offset*)

Returns a new `PKI_TIME` with offset (secs) from current time.

**1.37.1.7** `int` `PKI_TIME_print` (`PKI_TIME` \* *time*)

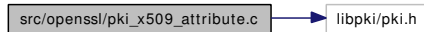
Prints a `PKI_TIME` to standard output.

**1.37.1.8** `int` `PKI_TIME_print_fp` (`FILE` \* *fp*, `PKI_TIME` \* *time*)

Prints out a `PKI_TIME` to a `FILE` stream.

**1.38** 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.*

### 1.38.1 Function Documentation

**1.38.1.1** `int PKI_STACK_X509_ATTRIBUTE_add` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*, PKI\_X509\_ATTRIBUTE \* *a*)

**1.38.1.2** `int PKI_STACK_X509_ATTRIBUTE_delete` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*, PKI\_ID *attr*)

**1.38.1.3** `int PKI_STACK_X509_ATTRIBUTE_delete_by_name` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*, char \* *name*)

**1.38.1.4** `int PKI_STACK_X509_ATTRIBUTE_delete_by_num` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*, int *num*)

**1.38.1.5** `void PKI_STACK_X509_ATTRIBUTE_free` (PKI\_X509\_ATTRIBUTE\_STACK \* *sk*)

Frees the memory associated with a stack of PKI\_X509\_ATTRIBUTE.

**1.38.1.6** `void PKI_STACK_X509_ATTRIBUTE_free_all` (PKI\_X509\_ATTRIBUTE\_STACK \* *sk*)

**1.38.1.7** `PKI_X509_ATTRIBUTE* PKI_STACK_X509_ATTRIBUTE_get` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*, PKI\_ID *attribute\_id*)

**1.38.1.8** `PKI_X509_ATTRIBUTE* PKI_STACK_X509_ATTRIBUTE_get_by_name` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*, char \* *name*)

**1.38.1.9** `PKI_X509_ATTRIBUTE* PKI_STACK_X509_ATTRIBUTE_get_by_num` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*, int *num*)

**1.38.1.10** `int PKI_STACK_X509_ATTRIBUTE_num` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*)

**1.38.1.11** `int PKI_STACK_X509_ATTRIBUTE_replace` (PKI\_X509\_ATTRIBUTE\_STACK \* *a\_sk*, PKI\_X509\_ATTRIBUTE \* *a*)

**1.38.1.12** `void PKI_X509_ATTRIBUTE_free` (PKI\_X509\_ATTRIBUTE \* *a*)

Frees the memory associated with a PKI\_X509\_ATTRIBUTE.



**1.38.1.13** void `PKI_X509_ATTRIBUTE_free_null` (void \* *a*)

**1.38.1.14** const char\* `PKI_X509_ATTRIBUTE_get_descr` (PKI\_X509\_ATTRIBUTE \* *a*)

**1.38.1.15** char\* `PKI_X509_ATTRIBUTE_get_parsed` (PKI\_X509\_ATTRIBUTE \* *a*)

**1.38.1.16** PKI\_STRING\* `PKI_X509_ATTRIBUTE_get_value` (PKI\_X509\_ATTRIBUTE \* *a*)

**1.38.1.17** PKI\_X509\_ATTRIBUTE\* `PKI_X509_ATTRIBUTE_new` (PKI\_ID *attribute\_id*, int *data\_type*, unsigned char \* *value*, size\_t *size*)

**1.38.1.18** PKI\_X509\_ATTRIBUTE\* `PKI_X509_ATTRIBUTE_new_name` (char \* *name*, int *data\_type*, char \* *value*, size\_t *size*)

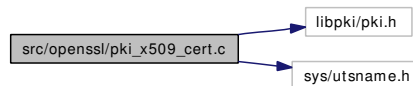
Returns a PKI\_X509\_ATTRIBUTE from a string description.

**1.38.1.19** PKI\_X509\_ATTRIBUTE\* `PKI_X509_ATTRIBUTE_new_null` (void)

Returns an empty PKI\_X509\_ATTRIBUTE.

## 1.39 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).*
- `int PKI_X509_CERT_set_data` (PKI\_X509\_CERT \*x, int type, void \*data)  
*Sets Data in a PKI\_X509\_CERT.*
- `int PKI_X509_CERT_sign` (PKI\_X509\_CERT \*cert, PKI\_X509\_KEYPAIR \*kp, PKI\_DIGEST\_ALG \*digest)  
*Signs a PKI\_X509\_CERT.*
- `int PKI_X509_CERT_sign_tk` (PKI\_X509\_CERT \*cert, PKI\_TOKEN \*tk, PKI\_DIGEST\_ALG \*digest)  
*Signs a PKI\_X509\_CERT by using a configured PKI\_TOKEN.*

## Variables

- `int NID_proxyCertInfo`

### 1.39.1 Function Documentation

#### 1.39.1.1 `int PKI_X509_CERT_add_extension` (PKI\_X509\_CERT \* x, PKI\_X509\_EXTENSION \* ext)

Adds a specific extension to a certificate.

#### 1.39.1.2 `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.

#### 1.39.1.3 `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.

#### 1.39.1.4 `PKI_X509_CERT* PKI_X509_CERT_dup` (PKI\_X509\_CERT \* x)

Returns a copy of the PKI\_X509\_CERT structure.

#### 1.39.1.5 `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.

#### 1.39.1.6 `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.

**1.39.1.7 void PKI\_X509\_CERT\_free (PKI\_X509\_CERT \* x)**

Frees the memory associated with a certificate.

**1.39.1.8 void PKI\_X509\_CERT\_free\_void (void \* x)****1.39.1.9 PKI\_STACK\* PKI\_X509\_CERT\_get\_cdp (PKI\_X509\_CERT \* x)**

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

**1.39.1.10 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.

**1.39.1.11 char\*\* PKI\_X509\_CERT\_get\_email (PKI\_X509\_CERT \* x)**

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

**1.39.1.12 PKI\_X509\_EXTENSION\* PKI\_X509\_CERT\_get\_extension\_by\_id (PKI\_X509\_CERT \* x, PKI\_ID num)****1.39.1.13 PKI\_X509\_EXTENSION\* PKI\_X509\_CERT\_get\_extension\_by\_name (PKI\_X509\_CERT \* x, char \* name)****1.39.1.14 PKI\_X509\_EXTENSION\* PKI\_X509\_CERT\_get\_extension\_by\_oid (PKI\_X509\_CERT \* x, PKI\_OID \* id)****1.39.1.15 PKI\_X509\_EXTENSION\_STACK\* PKI\_X509\_CERT\_get\_extensions (PKI\_X509\_CERT \* x)****1.39.1.16 int PKI\_X509\_CERT\_get\_keysize (PKI\_X509\_CERT \* x)**

Returns the size of the certificate public key.

**1.39.1.17 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).

**1.39.1.18 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.

**1.39.1.19 int PKI\_X509\_CERT\_is\_ca (PKI\_X509\_CERT \* x)**

Returns PKI\_OK if a certificate is allowed to sign certs.

**1.39.1.20 int PKI\_X509\_CERT\_is\_proxy (PKI\_X509\_CERT \* x)**

Returns PKI\_OK if a certificate is a Proxy Certificate.

**1.39.1.21 int PKI\_X509\_CERT\_is\_selfsigned (PKI\_X509\_CERT \* x)**

Returns PKI\_OK if a certificate is self-signed, PKI\_ERR otherwise.

**1.39.1.22 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.

**1.39.1.23 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 \*).

**1.39.1.24 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.

**1.39.1.25 PKI\_X509\_CERT\* PKI\_X509\_CERT\_new\_null (void)**

Returns an empty PKI\_X509\_CERT data structure.

**1.39.1.26 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).

**1.39.1.27 int PKI\_X509\_CERT\_set\_data (PKI\_X509\_CERT \* x, int type, void \* data)**

Sets Data in a PKI\_X509\_CERT.

**1.39.1.28 int PKI\_X509\_CERT\_sign (PKI\_X509\_CERT \* cert, PKI\_X509\_KEYPAIR \* kp, PKI\_DIGEST\_ALG \* digest)**

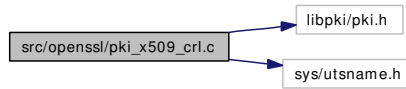
Signs a PKI\_X509\_CERT.

**1.39.1.29 int PKI\_X509\_CERT\_sign\_tk (PKI\_X509\_CERT \* cert, PKI\_TOKEN \* tk, PKI\_DIGEST\_ALG \* digest)**

Signs a PKI\_X509\_CERT by using a configured PKI\_TOKEN.

**1.39.2 Variable Documentation****1.39.2.1 int [NID\\_proxyCertInfo](#)****1.40 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)

### 1.40.1 Function Documentation

#### 1.40.1.1 int PKI\_X509\_CRL\_add\_extension (PKI\_X509\_CRL \*x, PKI\_X509\_EXTENSION \*ext)

Adds an Extension to a CRL object.

#### 1.40.1.2 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.

#### 1.40.1.3 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

#### 1.40.1.4 void PKI\_X509\_CRL\_ENTRY\_free\_void (void \*entry)

#### 1.40.1.5 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.

#### 1.40.1.6 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)

#### 1.40.1.7 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).

#### 1.40.1.8 void PKI\_X509\_CRL\_free\_void (void \*x)

#### 1.40.1.9 void\* PKI\_X509\_CRL\_get\_data (PKI\_X509\_CRL \*x, PKI\_X509\_DATA type)

Get Data from a CRL object.

**1.40.1.10** `char* PKI_X509_CRL_get_parsed (PKI_X509_CRL * x, PKI_X509_DATA type)`

**1.40.1.11** `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.

**1.40.1.12** `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.

**1.40.1.13** `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.

**1.40.1.14** `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.

**1.40.1.15** `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)`

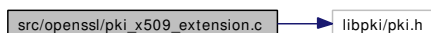
Generate a new CRL from a stack of revoked entries.

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.

**1.40.1.16** `int PKI_X509_CRL_print_parsed (PKI_X509_CRL * x, PKI_X509_DATA type, int fd)`

## 1.41 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, PKI\_TOKEN \*tk)

### 1.41.1 Function Documentation

**1.41.1.1 void PKI\_X509\_EXTENSION\_free (PKI\_X509\_EXTENSION \* ext)**

**1.41.1.2 void PKI\_X509\_EXTENSION\_free\_void (void \* ext)**

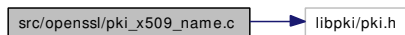
**1.41.1.3 PKI\_X509\_EXTENSION\_STACK\* PKI\_X509\_EXTENSION\_get\_list (void \* x, PKI\_X509\_DATA type)**

**1.41.1.4 PKI\_X509\_EXTENSION\* PKI\_X509\_EXTENSION\_new (void)**

**1.41.1.5 PKI\_X509\_EXTENSION\* PKI\_X509\_EXTENSION\_value\_new\_profile (PKI\_X509\_PROFILE \* profile, PKI\_CONFIG \* oids, PKI\_CONFIG\_ELEMENT \* extNode, PKI\_TOKEN \* tk)**

## 1.42 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.*

### 1.42.1 Function Documentation

#### 1.42.1.1 `PKI_X509_NAME* PKI_X509_NAME_add` (`PKI_X509_NAME * name`, `const char * entry`)

Adds a new entry to the passed PKI\_X509\_NAME.

#### 1.42.1.2 `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.

#### 1.42.1.3 `PKI_X509_NAME* PKI_X509_NAME_dup` (`PKI_X509_NAME * name`)

Returns a pointer to a duplicate of the passed name.

#### 1.42.1.4 `int PKI_X509_NAME_free` (`PKI_X509_NAME * name`)

#### 1.42.1.5 `PKI_DIGEST* PKI_X509_NAME_get_digest` (`PKI_X509_NAME * name`, `PKI_DIGEST_ALG * alg`)

Returns the digest of a PKI\_X509\_NAME.

**1.42.1.6** `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.

**1.42.1.7** `char* PKI_X509_NAME_get_parsed (PKI_X509_NAME * name)`

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

**1.42.1.8** `void PKI_X509_NAME_list_free (PKI_X509_NAME_RDN ** list)`

Frees the memory associated with a PKI\_X509\_NAME\_RDN data st.

**1.42.1.9** `PKI_X509_NAME* PKI_X509_NAME_new (char * name)`

**1.42.1.10** `PKI_X509_NAME* PKI_X509_NAME_new_null (void)`

**1.42.1.11** `const char* PKI_X509_NAME_RDN_type_descr (PKI_X509_NAME_RDN * rdn)`

Returns the description of the type of an RDN.

**1.42.1.12** `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.

**1.42.1.13** `const char* PKI_X509_NAME_RDN_type_text (PKI_X509_NAME_RDN * rdn)`

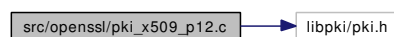
Returns the text representation of the type of an RDN.

**1.42.1.14** `char* PKI_X509_NAME_RDN_value (PKI_X509_NAME_RDN * rdn)`

Returns the value (char \*) of an RDN.

## 1.43 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.*

### 1.43.1 Enumeration Type Documentation

#### 1.43.1.1 enum `bag_datatype_st`

Enumerator:

*BAG\_DATATYPE\_ALL*  
*BAG\_DATATYPE\_KEYPAIR*  
*BAG\_DATATYPE\_CERT*  
*BAG\_DATATYPE\_CACERT*  
*BAG\_DATATYPE\_OTHERCERTS*  
*BAG\_DATATYPE\_UNKNOWN*

### 1.43.2 Function Documentation

#### 1.43.2.1 `PKI_X509_PKCS12_VALUE* PEM_read_bio_PKCS12 (PKI_IO * bp)`

#### 1.43.2.2 `int PEM_write_bio_PKCS12 (PKI_IO * bp, PKI_X509_PKCS12_VALUE * o)`

#### 1.43.2.3 `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.

#### 1.43.2.4 `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.

#### 1.43.2.5 `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.

#### 1.43.2.6 `void PKI_X509_PKCS12_DATA_free (PKI_X509_PKCS12_DATA * p12_data)`

#### 1.43.2.7 `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.

#### 1.43.2.8 `void PKI_X509_PKCS12_free (PKI_X509_PKCS12 * p12)`

Releases the memory associated with a PKI\_X509\_PKCS12 object.

#### 1.43.2.9 `void PKI_X509_PKCS12_free_void (void * p12)`

**1.43.2.10 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.

**1.43.2.11 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.

**1.43.2.12 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.

**1.43.2.13 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.

**1.43.2.14 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.

**1.43.2.15 PKI\_X509\_PKCS12\* PKI\_X509\_PKCS12\_new\_null (void)**

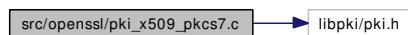
Allocates memory for a new PKI\_X509\_PKCS12 object.

**1.43.2.16 int PKI\_X509\_PKCS12\_TOKEN\_export (PKI\_TOKEN \* *tk*, URL \* *url*, int *format*, HSM \* *hsm*)****1.43.2.17 int PKI\_X509\_PKCS12\_verify\_cred (PKI\_X509\_PKCS12 \* *p12*, PKI\_CRED \* *cred*)**

Verifies the MAC against the passed credentials.

**1.44 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`)



### 1.44.1 Function Documentation

**1.44.1.1** `int PKI_X509_PKCS7_add_attribute (PKI_X509_PKCS7 * p7, PKI_X509_ATTRIBUTE * a)`

**1.44.1.2** `int PKI_X509_PKCS7_add_cert (PKI_X509_PKCS7 * p7, PKI_X509_CERT * x)`

Adds a certificate to the signature's certificate chain.

**1.44.1.3** `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.

**1.44.1.4** `int PKI_X509_PKCS7_add_crl (PKI_X509_PKCS7 * p7, PKI_X509_CRL * crl)`

**1.44.1.5** `int PKI_X509_PKCS7_add_crl_stack (PKI_X509_PKCS7 * p7, PKI_X509_CRL_STACK * crl_sk)`

**1.44.1.6** `int PKI_X509_PKCS7_add_recipient (PKI_X509_PKCS7 * p7, PKI_X509_CERT * x)`

Adds a new recipient for the PKI\_X509\_PKCS7.

**1.44.1.7** `int PKI_X509_PKCS7_add_signed_attribute (PKI_X509_PKCS7 * p7, PKI_X509_ATTRIBUTE * a)`

**1.44.1.8** `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).

**1.44.1.9** `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.

**1.44.1.10** `int PKI_X509_PKCS7_clear_certs (PKI_X509_PKCS7 * p7)`

Clears the chain of certificate for the signer.

**1.44.1.11** `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.

**1.44.1.12** `int PKI_X509_PKCS7_delete_attribute (PKI_X509_PKCS7 * p7, PKI_ID id)`

Deletes an attribute (id) from a PKI\_X509\_PKCS7.

**1.44.1.13 int PKI\_X509\_PKCS7\_delete\_signed\_attribute (PKI\_X509\_PKCS7 \* *p7*, PKI\_ID *id*)**

Deletes a signed attribute (*id*) from a PKI\_X509\_PKCS7.

**1.44.1.14 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.

**1.44.1.15 void PKI\_X509\_PKCS7\_free (PKI\_X509\_PKCS7 \* *p7*)****1.44.1.16 void PKI\_X509\_PKCS7\_free\_void (void \* *p7*)****1.44.1.17 PKI\_X509\_ATTRIBUTE\* PKI\_X509\_PKCS7\_get\_attribute (PKI\_X509\_PKCS7 \* *p7*, PKI\_ID *id*)****1.44.1.18 PKI\_X509\_ATTRIBUTE\* PKI\_X509\_PKCS7\_get\_attribute\_by\_name (PKI\_X509\_PKCS7 \* *p7*, char \* *name*)****1.44.1.19 PKI\_X509\_CERT\* PKI\_X509\_PKCS7\_get\_cert (PKI\_X509\_PKCS7 \* *p7*, int *idx*)**

Returns the *n*-th cert from a signed/signature PKCS7.

**1.44.1.20 int PKI\_X509\_PKCS7\_get\_certs\_num (PKI\_X509\_PKCS7 \* *p7*)**

Returns the number of certificates present in the signature chain.

**1.44.1.21 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.

**1.44.1.22 int PKI\_X509\_PKCS7\_get\_crls\_num (PKI\_X509\_PKCS7 \* *p7*)**

Returns the number of CRLs present in the signature.

**1.44.1.23 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.

**1.44.1.24 PKI\_MEM\* PKI\_X509\_PKCS7\_get\_dataTk (PKI\_X509\_PKCS7 \* *p7*, PKI\_TOKEN \* *tk*)**

Decrypts (if needed) and returns the data from a PKI\_X509\_PKCS7 by using keypair and, if present, cert of the PKI\_TOKEN argument.

**1.44.1.25 PKI\_ALGOR\* PKI\_X509\_PKCS7\_get\_encode\_alg (PKI\_X509\_PKCS7 \* *p7*)**

Returns the encryption algorithm.

**1.44.1.26** `PKI_MEM* PKI_X509_PKCS7_get_raw_data (PKI_X509_PKCS7 * p7)`

Returns the raw data contained in a PKI\_X509\_PKCS7 (any type).

**1.44.1.27** `PKI_X509_CERT* PKI_X509_PKCS7_get_recipient_cert (PKI_X509_PKCS7 * p7, int idx)`

Returns a copy of the n-th recipient certificate.

**1.44.1.28** `PKCS7_RECIP_INFO* PKI_X509_PKCS7_get_recipient_info (PKI_X509_PKCS7 * p7, int idx)`**1.44.1.29** `int PKI_X509_PKCS7_get_recipients_num (PKI_X509_PKCS7 * p7)`

Returns the number of recipients.

**1.44.1.30** `PKI_X509_ATTRIBUTE* PKI_X509_PKCS7_get_signed_attribute (PKI_X509_PKCS7 * p7, PKI_ID id)`**1.44.1.31** `PKI_X509_ATTRIBUTE* PKI_X509_PKCS7_get_signed_attribute_by_name (PKI_X509_PKCS7 * p7, char * name)`**1.44.1.32** `PKCS7_SIGNER_INFO* PKI_X509_PKCS7_get_signer_info (PKI_X509_PKCS7 * p7, int idx)`**1.44.1.33** `int PKI_X509_PKCS7_get_signers_num (PKI_X509_PKCS7 * p7)`

Returns the number of signers.

**1.44.1.34** `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).

**1.44.1.35** `int PKI_X509_PKCS7_has_recipients (PKI_X509_PKCS7 * p7)`

Returns PKI\_OK if the p7 has recipients already set, PKI\_ERR otherwise.

**1.44.1.36** `int PKI_X509_PKCS7_has_signers (PKI_X509_PKCS7 * p7)`

Returns PKI\_OK if the p7 has signers already set, PKI\_ERR otherwise.

**1.44.1.37** `PKI_X509_PKCS7* PKI_X509_PKCS7_new (PKI_X509_PKCS7_TYPE type)`**1.44.1.38** `int PKI_X509_PKCS7_set_cipher (PKI_X509_PKCS7 * p7, PKI_CIPHER * cipher)`

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

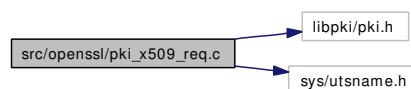
**1.44.1.39** `int PKI_X509_PKCS7_set_recipients (PKI_X509_PKCS7 * p7, PKI_X509_CERT_STACK * x_sk)`

Sets the recipients for a PKI\_X509\_PKCS7.

**1.44.1.40** `int PKI_X509_PKCS7_VALUE_print_bio (PKI_IO * bio, PKI_X509_PKCS7_VALUE * p7val)`

## 1.45 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.*

### 1.45.1 Function Documentation

#### 1.45.1.1 `int PKI_X509_REQ_add_attribute` (`PKI_X509_REQ * req`, `PKI_X509_ATTRIBUTE * attr`)

Adds an Attribute to a `PKI_X509_REQ`

#### 1.45.1.2 `int PKI_X509_REQ_add_extension` (`PKI_X509_REQ * x`, `PKI_X509_EXTENSION * ext`)

Adds an extension to a Certificate Request.

#### 1.45.1.3 `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.

#### 1.45.1.4 `int PKI_X509_REQ_clear_attributes` (`PKI_X509_REQ * req`)

Clears the stack of attributes from a `PKI_X509_REQ`.

#### 1.45.1.5 `int PKI_X509_REQ_delete_attribute` (`PKI_X509_REQ * req`, `PKI_ID id`)

Deletes an attribute by using its `PKI_ID`.

**1.45.1.6 int PKI\_X509\_REQ\_delete\_attribute\_by\_name (PKI\_X509\_REQ \* *req*, char \* *name*)**

Deletes an attribute by using its name.

**1.45.1.7 int PKI\_X509\_REQ\_delete\_attribute\_by\_num (PKI\_X509\_REQ \* *req*, int *pos*)**

Deletes an attribute by using its number (position).

**1.45.1.8 void PKI\_X509\_REQ\_free (PKI\_X509\_REQ \* *x*)**

Frees the memory associated with a Certificate Request object.

**1.45.1.9 void PKI\_X509\_REQ\_free\_void (void \* *x*)****1.45.1.10 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).

**1.45.1.11 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.

**1.45.1.12 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).

**1.45.1.13 int PKI\_X509\_REQ\_get\_attributes\_num (PKI\_X509\_REQ \* *req*)**

Gets the number of attributes present in a PKI\_X509\_REQ.

**1.45.1.14 void\* PKI\_X509\_REQ\_get\_data (PKI\_X509\_REQ \* *req*, PKI\_X509\_DATA *type*)**

Returns an attribute of the Certificate Request.

**1.45.1.15 int PKI\_X509\_REQ\_get\_extension\_by\_name (PKI\_X509\_REQ \* *req*, char \* *name*)****1.45.1.16 int PKI\_X509\_REQ\_get\_extension\_by\_num (PKI\_X509\_REQ \* *req*, int *num*)****1.45.1.17 int PKI\_X509\_REQ\_get\_extension\_by\_oid (PKI\_X509\_REQ \* *req*, PKI\_OID \* *oid*)****1.45.1.18 int PKI\_X509\_REQ\_get\_keysize (PKI\_X509\_REQ \* *x*)**

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

**1.45.1.19 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.

**1.45.1.20** `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.

**1.45.1.21** `PKI_X509_REQ* PKI_X509_REQ_new_null (void)`

**1.45.1.22** `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.

## 1.46 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.*

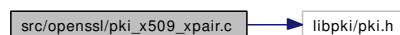
### 1.46.1 Function Documentation

**1.46.1.1** `char* PKI_X509_SIGNATURE_get_parsed (PKI_X509_SIGNATURE * sig)`

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

## 1.47 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.*

### 1.47.1 Function Documentation

**1.47.1.1** `void PKI_X509_XPAIR_free` (`PKI_X509_XPAIR *x`)

**1.47.1.2** `void PKI_X509_XPAIR_free_void` (`void *x`)

**1.47.1.3** `PKI_X509_CERT* PKI_X509_XPAIR_get_forward` (`PKI_X509_XPAIR *xp`)

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

**1.47.1.4** `PKI_X509_CERT* PKI_X509_XPAIR_get_reverse` (`PKI_X509_XPAIR *xp`)

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

**1.47.1.5** `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.

**1.47.1.6** `PKI_X509_XPAIR* PKI_X509_XPAIR_new_null` (`void`)

**1.47.1.7** `int PKI_X509_XPAIR_set_forward` (`PKI_X509_XPAIR *xp`, `PKI_X509_CERT *cert`)

Sets the forward certificate in a Cross Cert data structure.

**1.47.1.8** `int PKI_X509_XPAIR_set_reverse` (`PKI_X509_XPAIR *xp`, `PKI_X509_CERT *cert`)

Sets the reverse certificate in a Cross Cert data structure.

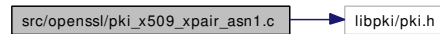
**1.47.1.9** `PKI_XPAIR* PKI_XPAIR_new_null` (`void`)

Create an empty cross certificate data structure.



## 1.48 src/openssl/pki\_x509\_xpair\_asn1.c File Reference

Include dependency graph for pki\_x509\_xpair\_asn1.c:



## 1.49 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`

### 1.49.1 Define Documentation

#### 1.49.1.1 `#define CRYPTO_LOCK 0x01`

#### 1.49.1.2 `#define CRYPTO_READ 0x04`

#### 1.49.1.3 `#define CRYPTO_UNLOCK 0x02`

#### 1.49.1.4 `#define CRYPTO_WRITE 0x08`

### 1.49.2 Function Documentation

#### 1.49.2.1 `struct CRYPTO_dynlock_value* _dyn_create_callback` (const char \*file, int line)

1.49.2.2 void `_dyn_destroy_callback` (struct CRYPTO\_dynlock\_value \* *l*, const char \* *file*, int *line*)

1.49.2.3 void `_dyn_lock_callback` (int *mode*, struct CRYPTO\_dynlock\_value \* *l*, const char \* *file*, int *line*)

1.49.2.4 void `thread_cleanup` (void)

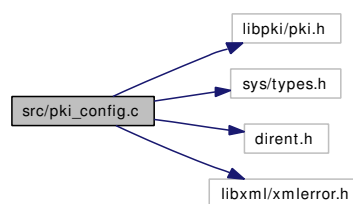
1.49.2.5 void `win32_locking_callback` (int *mode*, int *type*, char \* *file*, int *line*)

### 1.49.3 Variable Documentation

1.49.3.1 struct CRYPTO\_dynlock\_value\* [lock\\_cs](#)

## 1.50 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_CONFIG * PKI_CONFIG_OID_load (char *oidFile)`  
*Loads an OID file and creates internal OIDs.*
- `PKI_OID * PKI_CONFIG_OID_search (PKI_CONFIG *doc, char *searchName)`  
*Searches for a specific OID inside a PKI\_CONFIG object.*

### 1.50.1 Define Documentation

1.50.1.1 `#define LIBXML_MIN_VERSION 20600`

1.50.1.2 `#define PKI_DEF_CONF_DIRS_SIZE 2`

1.50.1.3 `#define xmlErrorPtr void *`

### 1.50.2 Function Documentation

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

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

1.50.2.3 `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.

**1.50.2.4** `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.

**1.50.2.5** `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.

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

Create a new Node for a PKI\_X509\_PROFILE.

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

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

**1.50.2.8** `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'.

**1.50.2.9** `int PKI_CONFIG_free (PKI_CONFIG * doc)`

Frees the memory associated with a PKI\_CONFIG object.

**1.50.2.10** `void PKI_CONFIG_free_void (void * doc)`

**1.50.2.11** `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.

**1.50.2.12** `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.

**1.50.2.13** `PKI_CONFIG_ELEMENT* PKI_CONFIG_get_element_child (PKI_CONFIG_ELEMENT * e)`

Returns the child of a PKI\_CONFIG\_ELEMENT.

**1.50.2.14** `PKI_CONFIG_ELEMENT_STACK* PKI_CONFIG_get_element_children (PKI_CONFIG_ELEMENT * e)`

Returns the stack of a PKI\_CONFIG\_ELEMENT's children.

**1.50.2.15** `char* PKI_CONFIG_get_element_name (PKI_CONFIG_ELEMENT * e)`

Returns the name of a PKI\_CONFIG\_ELEMENT.

**1.50.2.16** `PKI_CONFIG_ELEMENT*` `PKI_CONFIG_get_element_next` (`PKI_CONFIG_ELEMENT * e`)

Returns the next `PKI_CONFIG_ELEMENT`.

**1.50.2.17** `PKI_CONFIG_ELEMENT*` `PKI_CONFIG_get_element_prev` (`PKI_CONFIG_ELEMENT * e`)

Returns the previous `PKI_CONFIG_ELEMENT`.

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

Returns the stack of elements identified by the search path.

**1.50.2.19** `char*` `PKI_CONFIG_get_element_value` (`PKI_CONFIG_ELEMENT * e`)

Returns the value of a `PKI_CONFIG_ELEMENT`.

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

Returns the number of items identified by the search path.

**1.50.2.21** `PKI_CONFIG_ELEMENT*` `PKI_CONFIG_get_root` (`PKI_CONFIG * doc`)

Gets the root element of a `PKI_CONFIG` document.

**1.50.2.22** `PKI_STACK*` `PKI_CONFIG_get_search_paths` (`char * dir`)

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

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

Returns a stack of values for the selected search path.

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

Returns the first value found via the provided search path.

**1.50.2.25** `PKI_CONFIG*` `PKI_CONFIG_load` (`char * urlPath`)

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

**1.50.2.26** `PKI_CONFIG_STACK*` `PKI_CONFIG_load_all` (`char * dir`)

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

**1.50.2.27** `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.

**1.50.2.28 PKI\_CONFIG\* PKI\_CONFIG\_OID\_load (char \* oidFile)**

Loads an OID file and creates internal OIDs.

**1.50.2.29 PKI\_OID\* PKI\_CONFIG\_OID\_search (PKI\_CONFIG \* doc, char \* searchName)**

Searches for a specific OID inside a PKI\_CONFIG object.

**1.51 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.*

**1.51.1 Function Documentation****1.51.1.1 PKI\_CRED\* PKI\_CRED\_dup (PKI\_CRED \* cred)**

Duplicates a PKI\_CRED data structure.

**1.51.1.2 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).

**1.51.1.3 struct pki\_ssl\_t\* PKI\_CRED\_get\_ssl (PKI\_CRED \* cred)**

Gets the pointer to the PKI\_SSL structure inside a CRED.

**1.51.1.4 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.

**1.51.1.5 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.

**1.51.1.6 int PKI\_CRED\_set\_ssl (PKI\_CRED \* cred, struct pki\_ssl\_t \* ssl)**

Sets the SSL configuration for Creds.

**1.52 src/pki\_err.c File Reference**

Include dependency graph for pki\_err.c:

**Defines**

- `#define __LIBPKI_ERR__`

**Functions**

- `int __pki_error (const char *file, int line, int err, const char *info,...)`  
*Set and logs library errors.*

**Variables**

- `PKI_STACK * pki_err_stack = NULL`

**1.52.1 Define Documentation****1.52.1.1 #define \_\_LIBPKI\_ERR\_\_**



## 1.52.2 Function Documentation

### 1.52.2.1 int \_\_pki\_error (const char \*file, int line, int err, const char \*info, ...)

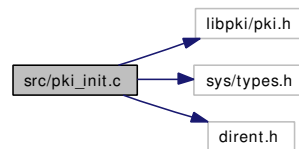
Set and logs library errors.

## 1.52.3 Variable Documentation

### 1.52.3.1 PKI\_STACK\* pki\_err\_stack = NULL

## 1.53 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

## 1.53.1 Function Documentation

### 1.53.1.1 PKI\_TOKEN\_STACK\* PKI\_get\_all\_tokens (char \* dir)

### 1.53.1.2 PKI\_TOKEN\_STACK\* PKI\_get\_all\_tokens\_dir (char \* dir, PKI\_TOKEN\_STACK \* list)

**1.53.1.3 int PKI\_get\_init\_status (void)**

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

**1.53.1.4 int PKI\_init\_all (void)**

Initialize libpki internal structures.

**1.53.1.5 PKI\_ID\_INFO\_STACK\* PKI\_list\_all\_id (void)**

Returns a stack of all the available Identities.

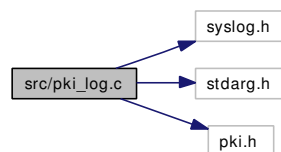
**1.53.1.6 PKI\_STACK\* PKI\_list\_all\_tokens (char \* dir)****1.53.1.7 PKI\_STACK\* PKI\_list\_all\_tokens\_dir (char \* dir, PKI\_STACK \* list)****1.53.2 Variable Documentation****1.53.2.1 const long LIBPKI\_OS\_DETAILS**

Initial value:

```
LIBPKI_OS_CLASS |
LIBPKI_OS_BITS | LIBPKI_OS_VENDOR
```

**1.53.2.2 int NID\_proxyCertInfo = -1****1.54 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.*

### 1.54.1 Function Documentation

#### 1.54.1.1 void PKI\_log (int level, const char \*fmt, ...)

Add an entry in the log.

#### 1.54.1.2 void PKI\_log\_debug\_simple (const char \*fmt, ...)

Add an entry in the Debug log.

#### 1.54.1.3 int PKI\_log\_end (void)

Finalize the log subsystem.

#### 1.54.1.4 void PKI\_log\_err\_simple (const char \*fmt, ...)

Add an entry in the Debug log.

#### 1.54.1.5 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.

## 1.55 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.*

### 1.55.1 Function Documentation

#### 1.55.1.1 void PKI\_Free (void \*ret)

Frees memory associated with a pointer (allocated with PKI\_Malloc).

#### 1.55.1.2 void\* PKI\_Malloc (size\_t size)

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

#### 1.55.1.3 int PKI\_MEM\_add (PKI\_MEM \*buf, char \*data, size\_t data\_size)

Adds the passed data to a PKI\_MEM.

#### 1.55.1.4 int PKI\_MEM\_B64\_decode (PKI\_MEM \*b64\_mem)

Decodes a PKI\_MEM from B64.

#### 1.55.1.5 int PKI\_MEM\_B64\_encode (PKI\_MEM \*der)

Encodes the contents of a PKI\_MEM in B64 format.

#### 1.55.1.6 PKI\_MEM\* PKI\_MEM\_dup (PKI\_MEM \*mem)

Duplicates a PKI\_MEM.

#### 1.55.1.7 ssize\_t PKI\_MEM\_fprintf (FILE \*file, PKI\_MEM \*buf)

Prints the content of a PKI\_MEM to a FILE pointer.

#### 1.55.1.8 int PKI\_MEM\_free (PKI\_MEM \*buf)

Frees the memory associated with a PKI\_MEM, returns 1.

#### 1.55.1.9 void PKI\_MEM\_free\_void (void \*buf)

#### 1.55.1.10 unsigned char\* PKI\_MEM\_get\_data (PKI\_MEM \*buf)

Returns the pointer to the data within a PKI\_MEM datastructure.

**1.55.1.11 size\_t PKI\_MEM\_get\_size (PKI\_MEM \* buf)**

Returns the size of the data within a PKI\_MEM datastructure.

**1.55.1.12 int PKI\_MEM\_grow (PKI\_MEM \* buf, size\_t data\_size)**

Grows the allocated size of data\_size bytes.

**1.55.1.13 PKI\_MEM\* PKI\_MEM\_new (size\_t size)**

Returns a new PKI\_MEM object with size allocated data.

**1.55.1.14 PKI\_MEM\* PKI\_MEM\_new\_bio (PKI\_IO \* io, PKI\_MEM \*\* mem)**

Returns a PKI\_MEM with the contents read from the PKI\_IO.

**1.55.1.15 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.

**1.55.1.16 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 renderer function from func which accepts BIO \* and data \* as its input.

**1.55.1.17 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\*).

**1.55.1.18 PKI\_MEM\* PKI\_MEM\_new\_membio (PKI\_IO \* io)**

Returns a PKI\_MEM with the contents of a memory PKI\_IO.

**1.55.1.19 PKI\_MEM\* PKI\_MEM\_new\_null (void)**

Returns a new PKI\_MEM object with no data associated with it.

**1.55.1.20 ssize\_t PKI\_MEM\_printf (PKI\_MEM \* buf)**

Prints the content of a PKI\_MEM to stdout.

**1.55.1.21 PKI\_MEM\* PKI\_MEM\_url\_decode (PKI\_MEM \* mem, int skip\_newlines)**

Decodes a URL-encoded version of a PKI\_MEM.

**1.55.1.22 PKI\_MEM\* PKI\_MEM\_url\_encode (PKI\_MEM \* mem, int skip\_newlines)**

Returns a URL-safe encoded version of a PKI\_MEM.

**1.55.1.23 void PKI\_ZFree (void \* *pnt*, size\_t *size*)**

Frees and Zeroizes memory associated with a pointer.

**1.55.1.24 void PKI\_ZFree\_str (char \* *str*)**

Frees and Zeroizes memory associated with a string.

**1.56 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.*

### 1.56.1 Function Documentation

#### 1.56.1.1 int PKI\_MSG\_REQ\_add\_data (PKI\_MSG\_REQ \* msg, unsigned char \* data, size\_t size)

Adds data to the Message body.

#### 1.56.1.2 int PKI\_MSG\_REQ\_add\_recipient (PKI\_MSG\_REQ \* msg, PKI\_X509\_CERT \* x)

Adds a certificate to the list of recipients.

#### 1.56.1.3 int PKI\_MSG\_REQ\_clear\_data (PKI\_MSG\_REQ \* msg)

Clears the data of the Message body.

#### 1.56.1.4 int PKI\_MSG\_REQ\_clear\_recipients (PKI\_MSG\_REQ \* msg)

Clears the list of recipients in a PKI\_MSG\_REQ.

#### 1.56.1.5 int PKI\_MSG\_REQ\_encode (PKI\_MSG\_REQ \* msg, PKI\_MSG\_PROTO proto)

Encodes the message according to the selected PKI\_MSG\_PROTO.

#### 1.56.1.6 void PKI\_MSG\_REQ\_free (PKI\_MSG\_REQ \* msg)

Free a PKI\_MSG\_REQ data structure.

#### 1.56.1.7 PKI\_MSG\_REQ\_ACTION PKI\_MSG\_REQ\_get\_action (PKI\_MSG\_REQ \* msg)

Gets the basic action in a PKI\_MSG\_REQ message.

#### 1.56.1.8 PKI\_X509\_CERT\* PKI\_MSG\_REQ\_get\_cacert (PKI\_MSG\_REQ \* msg)

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

#### 1.56.1.9 void\* PKI\_MSG\_REQ\_get\_encoded (PKI\_MSG\_REQ \* msg)

Gets the encoded version of the message.

**1.56.1.10 PKI\_X509\_KEYPAIR\* PKI\_MSG\_REQ\_get\_keypair (PKI\_MSG\_REQ \* msg)**

Gets the Keypair to be used when generating the request.

**1.56.1.11 char\* PKI\_MSG\_REQ\_get\_loa (PKI\_MSG\_REQ \* msg)**

Gets the requested certificate template.

**1.56.1.12 PKI\_MSG\_PROTO PKI\_MSG\_REQ\_get\_proto (PKI\_MSG\_REQ \* msg)**

Returns the PKI\_MSG\_PROTO from a PKI\_MSG\_REQUEST object.

**1.56.1.13 PKI\_X509\_CERT\_STACK\* PKI\_MSG\_REQ\_get\_recipients (PKI\_MSG\_REQ \* msg)**

Gets the list of recipients in a PKI\_MSG\_REQ.

**1.56.1.14 PKI\_X509\_CERT\* PKI\_MSG\_REQ\_get\_signer (PKI\_MSG\_REQ \* msg)**

Gets the Signer Certificate.

**1.56.1.15 char\* PKI\_MSG\_REQ\_get\_subject (PKI\_MSG\_REQ \* msg)**

Gets the Subject to be used in the certificate request.

**1.56.1.16 char\* PKI\_MSG\_REQ\_get\_template (PKI\_MSG\_REQ \* msg)**

Gets the requested certificate template.

**1.56.1.17 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.

**1.56.1.18 PKI\_MSG\_REQ\* PKI\_MSG\_REQ\_new\_null (void)**

Returns an empty generic PKI request message.

**1.56.1.19 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

**1.56.1.20 int PKI\_MSG\_REQ\_replace\_data (PKI\_MSG\_REQ \* msg, unsigned char \* data, size\_t size)**

Replaces data in a PKI\_MSG\_REQ message.

**1.56.1.21 int PKI\_MSG\_REQ\_SCEP\_new (PKI\_MSG\_REQ \* msg)**

Returns a SCEP\_MSG from the passed PKI\_MSG\_REQ

**1.56.1.22** `PKI_MSG_RESP* PKI_MSG_REQ_SCEP_send (PKI_MSG_REQ * msg, PKI_STACK * sk, PKI_TOKEN * tk)`

**1.56.1.23** `PKI_MSG_RESP* PKI_MSG_REQ_send (PKI_MSG_REQ * msg, PKI_TOKEN * tk, char * url_s)`

Sends a message and retrieves the response.

**1.56.1.24** `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.

**1.56.1.25** `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.

**1.56.1.26** `int PKI_MSG_REQ_set_keypair (PKI_MSG_REQ * msg, PKI_X509_KEYPAIR * pkey)`

Sets the Keypair to be used when generating the request.

**1.56.1.27** `int PKI_MSG_REQ_set_loa (PKI_MSG_REQ * msg, char * loa)`

Sets the requested certificate level of assurance (LOA).

**1.56.1.28** `int PKI_MSG_REQ_set_proto (PKI_MSG_REQ * msg, PKI_MSG_PROTO proto)`

Sets the messaging protocol to be used.

**1.56.1.29** `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.

**1.56.1.30** `int PKI_MSG_REQ_set_signer (PKI_MSG_REQ * msg, PKI_X509_CERT * signer)`

Sets the Signer Certificate.

**1.56.1.31** `int PKI_MSG_REQ_set_subject (PKI_MSG_REQ * msg, char * subject)`

Sets the Subject to be used in the certificate request.

**1.56.1.32** `int PKI_MSG_REQ_set_template (PKI_MSG_REQ * msg, char * name)`

Sets the requested certificate template.

## 1.57 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.*

### 1.57.1 Function Documentation

#### 1.57.1.1 `int PKI_MSG_RESP_add_data` (`PKI_MSG_RESP *msg`, unsigned char \*data, size\_t size)

Adds data to the Message body.

#### 1.57.1.2 `int PKI_MSG_RESP_add_recipient` (`PKI_MSG_RESP *msg`, `PKI_X509_CERT *x`)

Adds a certificate to the list of recipients.

**1.57.1.3 int PKI\_MSG\_RESP\_clear\_data (PKI\_MSG\_RESP \* msg)**

Clears the data of the Message body.

**1.57.1.4 int PKI\_MSG\_RESP\_clear\_recipients (PKI\_MSG\_RESP \* msg)**

Clears the list of recipients in a PKI\_MSG\_RESP.

**1.57.1.5 void\* PKI\_MSG\_RESP\_encode (PKI\_MSG\_RESP \* msg, PKI\_MSG\_PROTO proto)**

Encodes the message according to the selected PKI\_MSG\_PROTO.

**1.57.1.6 void PKI\_MSG\_RESP\_free (PKI\_MSG\_RESP \* msg)**

Free a PKI\_MSG\_REQ data structure.

**1.57.1.7 PKI\_MSG\_RESP\_ACTION PKI\_MSG\_RESP\_get\_action (PKI\_MSG\_RESP \* msg)**

Gets the basic action in a PKI\_MSG\_RESP message.

**1.57.1.8 PKI\_X509\_CERT\* PKI\_MSG\_RESP\_get\_cacert (PKI\_MSG\_RESP \* msg)**

Gets the CA certificate from the Response.

**1.57.1.9 void\* PKI\_MSG\_RESP\_get\_encoded (PKI\_MSG\_RESP \* msg)**

Gets the encoded version of the Response message.

**1.57.1.10 PKI\_X509\_CERT\* PKI\_MSG\_RESP\_get\_issued\_cert (PKI\_MSG\_RESP \* msg)**

Gets the certificate from the Response.

**1.57.1.11 PKI\_X509\_KEYPAIR\* PKI\_MSG\_RESP\_get\_keypair (PKI\_MSG\_RESP \* msg)**

Gets the Keypair to be used when generating the response.

**1.57.1.12 PKI\_MSG\_PROTO PKI\_MSG\_RESP\_get\_proto (PKI\_MSG\_RESP \* msg)**

Returns the PKI\_MSG\_PROTO from a PKI\_MSG\_RESP object.

**1.57.1.13 PKI\_X509\_CERT\_STACK\* PKI\_MSG\_RESP\_get\_recipients (PKI\_MSG\_RESP \* msg)**

Gets the list of recipients in a PKI\_MSG\_RESP.

**1.57.1.14 PKI\_X509\_CERT\* PKI\_MSG\_RESP\_get\_signer (PKI\_MSG\_RESP \* msg)**

Gets the Signer Certificate.

**1.57.1.15 PKI\_MSG\_STATUS PKI\_MSG\_RESP\_get\_status (PKI\_MSG\_RESP \* msg)**

Gets the status in a PKI\_MSG\_RESP message.

**1.57.1.16** `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.

**1.57.1.17** `PKI_MSG_RESP*` `PKI_MSG_RESP_new_null` (`void`)

Returns an empty generic PKI response message.

**1.57.1.18** `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

**1.57.1.19** `int` `PKI_MSG_RESP_replace_data` (`PKI_MSG_RESP` \* *msg*, `unsigned char` \* *data*, `size_t` *size*)

Replaces data in a PKI\_MSG\_RESP message.

**1.57.1.20** `PKI_X509_SCEP_MSG*` `PKI_MSG_RESP_SCEP_new` (`PKI_MSG_RESP` \* *msg*)

Returns a SCEP\_MSG from the passed PKI\_MSG\_RESP

**1.57.1.21** `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.

**1.57.1.22** `int` `PKI_MSG_RESP_set_cacert` (`PKI_MSG_RESP` \* *msg*, `PKI_X509_CERT` \* *x*)

Sets the CA certificate in the Response.

**1.57.1.23** `int` `PKI_MSG_RESP_set_issued_cert` (`PKI_MSG_RESP` \* *msg*, `PKI_X509_CERT` \* *x*)

Sets the certificate from the Response.

**1.57.1.24** `int` `PKI_MSG_RESP_set_keypair` (`PKI_MSG_RESP` \* *msg*, `PKI_X509_KEYPAIR` \* *pkey*)

Sets the Keypair to be used when generating the response.

**1.57.1.25** `int` `PKI_MSG_RESP_set_proto` (`PKI_MSG_RESP` \* *msg*, `PKI_MSG_PROTO` *proto*)

Sets the messaging protocol to be used.

**1.57.1.26** `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.

**1.57.1.27 int PKI\_MSG\_RESP\_set\_signer (PKI\_MSG\_RESP \* msg, PKI\_X509\_CERT \* signer)**

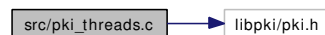
Sets the Signer Certificate.

**1.57.1.28 int PKI\_MSG\_RESP\_set\_status (PKI\_MSG\_RESP \* msg, PKI\_MSG\_STATUS status)**

Sets the status in a PKI\_MSG\_RESP message.

**1.58 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.*

**1.58.1 Function Documentation****1.58.1.1 int PKI\_THREAD\_create (PKI\_THREAD \* th, PKI\_THREAD\_ATTR \* attr, void \*(\*)(void \*) func, void \* arg)**

Spawns a new Thread.

**1.58.1.2 PKI\_THREAD\* PKI\_THREAD\_new (void \*(\*)(void \*arg) func, void \* arg)**

Creates and Spawns a new thread.

**1.58.1.3 PKI\_THREAD\_ID PKI\_THREAD\_self (void)**

Returns the identifier for current thread.

**1.59 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).*

### 1.59.1 Function Documentation

#### 1.59.1.1 struct timespec\* [PKI\\_clock\\_gettime](#) (void)

Returns the current time in a timespec structure.

#### 1.59.1.2 int [PKI\\_COND\\_broadcast](#) (PKI\_COND \* var)

Broadcasts on a condition variable (wakes up all waiting threads ).

#### 1.59.1.3 int [PKI\\_COND\\_destroy](#) (PKI\_COND \* var)

Destroys (but do not free memory) a condition variable.

#### 1.59.1.4 void [PKI\\_COND\\_free](#) (PKI\_COND \* var)

Frees the memory associated with a Condition Variable.

#### 1.59.1.5 int [PKI\\_COND\\_init](#) (PKI\_COND \* var)

Initializes an already allocated (or destroyed) condition variable.

#### 1.59.1.6 PKI\_COND\* [PKI\\_COND\\_new](#) ()

Creates a new Condition Variable used for thread management.

#### 1.59.1.7 int [PKI\\_COND\\_signal](#) (PKI\_COND \* var)

Signal on a condition variable (wakes up the first waiting thread).

**1.59.1.8 int PKI\_COND\_timedwait (PKI\_COND \* *var*, PKI\_Mutex \* *mutex*, struct timespec \* *t*)**

Waits on a condition variable for timespec time only.

**1.59.1.9 int PKI\_COND\_wait (PKI\_COND \* *var*, PKI\_Mutex \* *mutex*)**

Waits on a condition variable.

**1.59.1.10 int PKI\_Mutex\_acquire (PKI\_Mutex \* *var*)**

Acquires access for a mutex.

**1.59.1.11 int PKI\_Mutex\_destroy (PKI\_Mutex \* *var*)**

Destroys (but does not free the memory) a mutex structure.

**1.59.1.12 void PKI\_Mutex\_free (PKI\_Mutex \* *var*)**

Frees the memory associated with a PKI\_Mutex variable.

**1.59.1.13 int PKI\_Mutex\_init (PKI\_Mutex \* *var*)**

Initializes an already allocated mutex structure.

**1.59.1.14 PKI\_Mutex\* PKI\_Mutex\_new ()**

Creates a mutex variable to be used for critical sections.

**1.59.1.15 int PKI\_Mutex\_release (PKI\_Mutex \* *var*)**

Releases a mutex.

**1.59.1.16 int PKI\_RWLOCK\_destroy (PKI\_RWLOCK \* *l*)**

Destroys a R/W Lock (needed before re-initialization).

**1.59.1.17 void PKI\_RWLOCK\_free (PKI\_RWLOCK \* *l*)**

Destroys a R/W lock and Frees its memory.

**1.59.1.18 int PKI\_RWLOCK\_init (PKI\_RWLOCK \* *l*)**

Initializes a R/W Lock.

**1.59.1.19 PKI\_RWLOCK\* PKI\_RWLOCK\_new ()**

Allocates a new R/W Lock and Initializes it.

**1.59.1.20 int PKI\_RWLOCK\_read\_lock (PKI\_RWLOCK \* *l*)**

Locks a R/W lock in READ mode (SHARED).

**1.59.1.21 int PKI\_RWLOCK\_release (PKI\_RWLOCK \* l)**

Release a PKI\_RWLOCK.

**1.59.1.22 int PKI\_RWLOCK\_try\_read\_lock (PKI\_RWLOCK \* l)****1.59.1.23 int PKI\_RWLOCK\_try\_write\_lock (PKI\_RWLOCK \* l)****1.59.1.24 int PKI\_RWLOCK\_write\_lock (PKI\_RWLOCK \* l)**

Locks a R/W lock in WRITE mode (Exclusive).

**1.60 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.*

### 1.60.1 Function Documentation

#### 1.60.1.1 `const PKI_X509_CALLBACKS* PKI_X509_CALLBACKS_get (PKI_DATATYPE type, struct hsm_st * hsm)`

Returns the callbacks for a specific PKI\_DATATYPE.

#### 1.60.1.2 `int PKI_X509_delete (PKI_X509 * x)`

Deletes the hard copy (eg., file, hsm file, etc.) of the PKI\_X509 object.

#### 1.60.1.3 `PKI_X509* PKI_X509_dup (PKI_X509 * x)`

Duplicates a PKI\_X509 object.

#### 1.60.1.4 `void* PKI_X509_dup_value (PKI_X509 * x)`

Duplicates the PKI\_X509\_XXX\_VALUE from the passed PKI\_X509 object.

**1.60.1.5** void **PKI\_X509\_free** (PKI\_X509 \* *x*)

**1.60.1.6** void **PKI\_X509\_free\_void** (void \* *x*)

Frees the memory associated with a PKI\_X509 object.

**1.60.1.7** 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.

**1.60.1.8** struct hsm\_st\* **PKI\_X509\_get\_hsm** (PKI\_X509 \* *x*)

Retrieves the HSM reference from a PKI\_X509 object.

**1.60.1.9** 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.

**1.60.1.10** URL\* **PKI\_X509\_get\_reference** (PKI\_X509 \* *x*)

Retrieves the reference URL from a PKI\_X509 object.

**1.60.1.11** PKI\_DATATYPE **PKI\_X509\_get\_type** (PKI\_X509 \* *x*)

Returns the type of a PKI\_X509 object.

**1.60.1.12** void\* **PKI\_X509\_get\_value** (PKI\_X509 \* *x*)

Returns the reference to the PKI\_X509\_XXX\_VALUE withing a PKI\_X509 object.

**1.60.1.13** int **PKI\_X509\_is\_signed** (PKI\_X509 \* *obj*)

Returns PKI\_OK if the PKI\_X509 object is signed.

**1.60.1.14** PKI\_X509\* **PKI\_X509\_new** (PKI\_DATATYPE *type*, struct hsm\_st \* *hsm*)

Allocs the memory associated with an empty PKI\_X509 object.

**1.60.1.15** 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.

**1.60.1.16** 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.

**1.60.1.17** 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.

**1.60.1.18 int PKI\_X509\_set\_hsm (PKI\_X509 \* x, struct hsm\_st \* hsm)**

Sets the HSM reference in a PKI\_X509 object.

**1.60.1.19 int PKI\_X509\_set\_reference (PKI\_X509 \* x, URL \* url)**

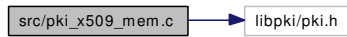
Sets (duplicates) the reference URL of a PKI\_X509 object.

**1.60.1.20 int PKI\_X509\_set\_value (PKI\_X509 \* x, void \* data)**

Sets the pointer to the internal value in a PKI\_X509.

**1.61 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.*

**1.61.1 Function Documentation****1.61.1.1 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.

**1.61.1.2** void\* **PKI\_X509\_get\_mem\_value** (PKI\_MEM \* *mem*, PKI\_DATATYPE *type*, PKI\_CRED \* *cred*, HSM \* *hsm*)

**1.61.1.3** 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.

**1.61.1.4** 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.

**1.61.1.5** 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.

**1.61.1.6** 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.

## 1.62 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*)

### 1.62.1 Function Documentation

**1.62.1.1** const char\* **PKI\_X509\_get\_mimetype** (PKI\_DATATYPE *type*)

## 1.63 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\_TOKEN \*tk)
- 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)

### 1.63.1 Function Documentation

**1.63.1.1 PKI\_CONFIG\_ELEMENT\* PKI\_X509\_PROFILE\_add\_extension (PKI\_X509\_PROFILE \* doc, char \* name, char \* value, char \* type, int crit)**

**1.63.1.2 int PKI\_X509\_PROFILE\_free (PKI\_X509\_PROFILE \* doc)**

**1.63.1.3 void PKI\_X509\_PROFILE\_free\_void (void \* doc)**

**1.63.1.4 PKI\_X509\_PROFILE\* PKI\_X509\_PROFILE\_get\_default (PKI\_X509\_PROFILE\_TYPE profile\_id)**

**1.63.1.5 PKI\_X509\_EXTENSION\* PKI\_X509\_PROFILE\_get\_ext\_by\_num (PKI\_X509\_PROFILE \* doc, int num, PKI\_TOKEN \* tk)**

**1.63.1.6 PKI\_CONFIG\_ELEMENT\* PKI\_X509\_PROFILE\_get\_extensions (PKI\_X509\_PROFILE \* doc)**

**1.63.1.7 int PKI\_X509\_PROFILE\_get\_exts\_num (PKI\_X509\_PROFILE \* doc)**

**1.63.1.8 char\* PKI\_X509\_PROFILE\_get\_name (PKI\_X509\_PROFILE \* doc)**

**1.63.1.9 char\* PKI\_X509\_PROFILE\_get\_value (PKI\_X509\_PROFILE \* doc, char \* path)**

**1.63.1.10 PKI\_X509\_PROFILE\* PKI\_X509\_PROFILE\_load (char \* urlPath)**

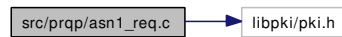
**1.63.1.11 PKI\_X509\_PROFILE\* PKI\_X509\_PROFILE\_new (char \* name)**

Create a new PKI\_X509\_PROFILE.

1.63.1.12 int `PKI_X509_PROFILE_put_file` (`PKI_X509_PROFILE * doc`, char \* *url*)

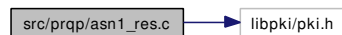
## 1.64 src/prqp/asn1\_req.c File Reference

Include dependency graph for `asn1_req.c`:



## 1.65 src/prqp/asn1\_res.c File Reference

Include dependency graph for `asn1_res.c`:



## 1.66 src/prqp/http\_client.c File Reference

Include dependency graph for `http_client.c`:



### Functions

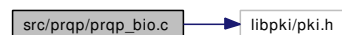
- `PKI_X509_PRQP_RESP *` [PKI\\_X509\\_PRQP\\_RESP\\_get\\_http](#) (URL \**url*, `PKI_X509_PRQP_REQ *`*req*, unsigned long *max\_size*)

### 1.66.1 Function Documentation

1.66.1.1 `PKI_X509_PRQP_RESP *` `PKI_X509_PRQP_RESP_get_http` (URL \* *url*, `PKI_X509_PRQP_REQ *` *req*, unsigned long *max\_size*)

## 1.67 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)`

### 1.67.1 Function Documentation

**1.67.1.1** `PKI_PRQP_REQ* d2i_PRQP_REQ_bio (BIO *bp, PKI_PRQP_REQ *p)`

**1.67.1.2** `PKI_PRQP_RESP* d2i_PRQP_RESP_bio (BIO *bp, PKI_PRQP_RESP *p)`

**1.67.1.3** `int i2d_PRQP_REQ_bio (BIO *bp, PKI_PRQP_REQ *o)`

**1.67.1.4** `int i2d_PRQP_RESP_bio (BIO *bp, PKI_PRQP_RESP *o)`

**1.67.1.5** `PKI_PRQP_REQ* PEM_read_bio_PRQP_REQ (BIO *bp)`

**1.67.1.6** `PKI_PRQP_RESP* PEM_read_bio_PRQP_RESP (BIO *bp)`

**1.67.1.7** `int PEM_write_bio_PRQP_REQ (BIO *bp, PKI_PRQP_REQ *o)`

**1.67.1.8** `int PEM_write_bio_PRQP_RESP (BIO *bp, PKI_PRQP_RESP *o)`

## 1.68 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](#) []

**1.68.1 Define Documentation****1.68.1.1 #define \_\_PKI\_PRQP\_LIB\_C\_\_****1.68.2 Function Documentation****1.68.2.1 int CERT\_IDENTIFIER\_cmp (CERT\_IDENTIFIER \* a, CERT\_IDENTIFIER \* b)**

**1.68.2.2 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)**

**1.68.2.3 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.

**1.68.2.4 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)**

**1.68.2.5 PKI\_INTEGER\* PKI\_X509\_PRQP\_NONCE\_new (int bits)****1.68.2.6 int PKI\_X509\_PRQP\_REQ\_add\_service (PKI\_X509\_PRQP\_REQ \* p, char \* ss)**

Adds a service identifier to a PRQP REQUEST.

**1.68.2.7 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.

**1.68.2.8 void PKI\_X509\_PRQP\_REQ\_free (PKI\_X509\_PRQP\_REQ \* x)****1.68.2.9 void PKI\_X509\_PRQP\_REQ\_free\_void (void \* x)**

**1.68.2.10 void\* PKI\_X509\_PRQP\_REQ\_get\_data (PKI\_X509\_PRQP\_REQ \* obj, PKI\_X509\_DATA type)**

Returns data pointers from a PRQP request.

**1.68.2.11** `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)`

**1.68.2.12** `void* PKI_X509_PRQP_REQ_new_null (void)`

**1.68.2.13** `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)`

**1.68.2.14** `int PKI_X509_PRQP_REQ_print (PKI_X509_PRQP_REQ * req)`

Prints out the contents of a PRQP\_REQ on stdout.

**1.68.2.15** `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.

**1.68.2.16** `int PKI_X509_PRQP_REQ_VALUE_print_bio (PKI_PRQP_REQ * req, BIO * bio)`

**1.68.2.17** `int PKI_X509_PRQP_REQ_verify (PKI_X509_PRQP_REQ * r)`

Verify Signature on the PRQP REQUEST.

**1.68.2.18** `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.

**1.68.2.19** `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.

**1.68.2.20** `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.

**1.68.2.21** `void PKI_X509_PRQP_RESP_free (PKI_X509_PRQP_RESP * x)`

Releases the memory associated with a PRQP\_RESP object.

**1.68.2.22** `void PKI_X509_PRQP_RESP_free_void (void * x)`

**1.68.2.23** `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.

**1.68.2.24 int PKI\_X509\_PRQP\_RESP\_get\_status (PKI\_X509\_PRQP\_RESP \* *obj*)**

Returns the PKI\_X509\_PRQP\_STATUS associated to a PRQP response.

**1.68.2.25 void\* PKI\_X509\_PRQP\_RESP\_new\_null (void)**

Creates a new PRQP\_RESP empty object.

**1.68.2.26 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*)**

Created a new PRQP\_RESP from the contents of a PRQP\_REQ.

**1.68.2.27 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.

**1.68.2.28 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.

**1.68.2.29 int PKI\_X509\_PRQP\_RESP\_print (PKI\_X509\_PRQP\_RESP \* *resp*)**

Prints out the contents of a PRQP\_RESP to stdout.

**1.68.2.30 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.

**1.68.2.31 PKI\_STACK\* PKI\_X509\_PRQP\_RESP\_url\_sk (PKI\_X509\_PRQP\_RESP \* *r*)**

Returns a PKI\_STACK of URLs from a PRQP\_RESP object.

**1.68.2.32 int PKI\_X509\_PRQP\_RESP\_VALUE\_print\_bio (PKI\_X509\_PRQP\_RESP\_VALUE \* *resp*, BIO \* *bio*)****1.68.2.33 int PKI\_X509\_PRQP\_RESP\_verify (PKI\_X509\_PRQP\_RESP \* *r*)****1.68.2.34 int PKI\_X509\_PRQP\_RESP\_version\_set (PKI\_X509\_PRQP\_RESP \* *resp*, int *ver*)**

Sets the protocol version of a PRQP\_RESP object.

**1.68.2.35 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.



**1.68.2.36** `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.

**1.68.2.37** `int PKI_X509_PRQP_verify (PKI_X509 *r)`

Verifies that the signature on a PRQP object is correct.

**1.68.2.38** `int PRQP_init_all_services (void)`

**1.68.2.39** `PKI_OID* PRQP_RESOURCE_RESPONSE_TOKEN_get_oid (RESOURCE_RESPONSE_TOKEN *rrt)`

**1.68.2.40** `PKI_STACK* PRQP_RESOURCE_RESPONSE_TOKEN_get_services (RESOURCE_RESPONSE_TOKEN *rrt)`

### 1.68.3 Variable Documentation

**1.68.3.1** `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
}
```

## 1.69 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`)

### 1.69.1 Function Documentation

**1.69.1.1** `PKI_X509_PRQP_REQ * PKI_X509_PRQP_REQ_get` (`char *url_s`, `PKI_CRED *cred`, `HSM *hsm`)

**1.69.1.2** `PKI_X509_PRQP_REQ * PKI_X509_PRQP_REQ_get_mem` (`PKI_MEM *mem`, `PKI_CRED *cred`, `HSM *hsm`)

**1.69.1.3** `PKI_X509_PRQP_REQ * PKI_X509_PRQP_REQ_get_url` (`URL *url`, `PKI_CRED *cred`, `HSM *hsm`)

**1.69.1.4** `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`)

**1.69.1.5** `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`)

**1.69.1.6** `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`)

**1.69.1.7** `PKI_X509_PRQP_REQ_STACK * PKI_X509_PRQP_REQ_STACK_get` (`char *url_s`, `PKI_CRED *cred`, `HSM *hsm`)

**1.69.1.8** `PKI_X509_PRQP_REQ_STACK * PKI_X509_PRQP_REQ_STACK_get_mem` (`PKI_MEM *mem`, `PKI_CRED *cred`, `HSM *hsm`)

**1.69.1.9** `PKI_X509_PRQP_REQ_STACK * PKI_X509_PRQP_REQ_STACK_get_url` (`URL *url`, `PKI_CRED *cred`, `HSM *hsm`)

**1.69.1.10** `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`)

**1.69.1.11** `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)`

**1.69.1.12** `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)`

## 1.70 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)`

### 1.70.1 Function Documentation

**1.70.1.1** `PKI_X509_PRQP_RESP* PKI_X509_PRQP_RESP_get (char * url_s, PKI_CRED * cred, HSM * hsm)`

**1.70.1.2** `PKI_X509_PRQP_RESP* PKI_X509_PRQP_RESP_get_mem (PKI_MEM * mem, PKI_CRED * cred, HSM * hsm)`

**1.70.1.3** `PKI_X509_PRQP_RESP*` `PKI_X509_PRQP_RESP_get_url` (`URL * url`, `PKI_CRED *` `cred`, `HSM *` `hsm`)

**1.70.1.4** `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`)

**1.70.1.5** `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`)

**1.70.1.6** `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`)

**1.70.1.7** `PKI_X509_PRQP_RESP_STACK*` `PKI_X509_PRQP_RESP_STACK_get` (`char *` `url_s`, `PKI_CRED *` `cred`, `HSM *` `hsm`)

**1.70.1.8** `PKI_X509_PRQP_RESP_STACK*` `PKI_X509_PRQP_RESP_STACK_get_mem` (`PKI_MEM *` `mem`, `PKI_CRED *` `cred`, `HSM *` `hsm`)

**1.70.1.9** `PKI_X509_PRQP_RESP_STACK*` `PKI_X509_PRQP_RESP_STACK_get_url` (`URL *` `url`, `PKI_CRED *` `cred`, `HSM *` `hsm`)

**1.70.1.10** `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`)

**1.70.1.11** `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`)

**1.70.1.12** `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`)

## 1.71 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)`

*Retrieve a PKI\_STACK of addresses from a PRQP server.*

- `char * PKI\_get\_ca\_service (PKI_X509_CERT *caCert, char *srv, char *url_s)`

*Retrieve 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)`

*Retrieve a stack of configured URL for a PKI service from a PRQP server.*

- `PKI_STACK * PKI\_get\_cert\_service\_sk (PKI_X509_CERT *cert, char *srv, char *url_s)`

### 1.71.1 Function Documentation

#### 1.71.1.1 `PKI_X509_PRQP_RESP* PKI_DISCOVER_get_resp (PKI_X509_PRQP_REQ * p, char * url_s)`

Retrieve a PRQP response from the passed url or from one of the configured RQAs in /etc/pki.conf.

#### 1.71.1.2 `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.

#### 1.71.1.3 `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)`

Retrieve 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).

#### 1.71.1.4 `char* PKI_get_ca_service (PKI_X509_CERT * caCert, char * srv, char * url_s)`

Retrieve 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).

#### 1.71.1.5 `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.

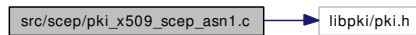
Retrieve information about a specific service provided by a CA. The function returns a `PKI_STACK` containing the URLs of the requested service (if available).

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).

#### 1.71.1.6 `PKI_STACK*` `PKI_get_cert_service_sk` (`PKI_X509_CERT` \* *cert*, char \* *srv*, char \* *url\_s*)

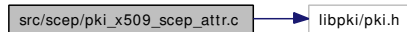
## 1.72 src/scep/pki\_x509\_scep\_asn1.c File Reference

Include dependency graph for `pki_x509_scep_asn1.c`:



## 1.73 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]

### 1.73.1 Define Documentation

#### 1.73.1.1 #define SCEP\_CONF\_LIST\_SIZE 8

### 1.73.2 Function Documentation

#### 1.73.2.1 PKI\_ID PKI\_X509\_SCEP\_ATTRIBUTE\_get\_nid (SCEP\_ATTRIBUTE\_TYPE *num*)

Returns the PKI\_ID of the specified SCEP\_ATTRIBUTE\_TYPE.

#### 1.73.2.2 SCEP\_ATTRIBUTE\_TYPE PKI\_X509\_SCEP\_ATTRIBUTE\_get\_txt (char \* *txt*)

Returns the SCEP\_ATTRIBUTE\_TYPE from the attribute name.

#### 1.73.2.3 void PKI\_X509\_SCEP\_init (void)

#### 1.73.2.4 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.

#### 1.73.2.5 int PKI\_X509\_SCEP\_MSG\_get\_attr\_value\_int (PKI\_X509\_SCEP\_MSG \* *msg*, SCEP\_ATTRIBUTE\_TYPE *type*)

#### 1.73.2.6 SCEP\_FAILURE PKI\_X509\_SCEP\_MSG\_get\_failinfo (PKI\_X509\_PKCS7 \* *msg*)

Returns the failInfo attribute from a SCEP message.

#### 1.73.2.7 PKI\_OID\* PKI\_X509\_SCEP\_MSG\_get\_oid (SCEP\_ATTRIBUTE\_TYPE *scep\_attribute*)

Returns the PKI\_OID for the specified SCEP attribute.

#### 1.73.2.8 int PKI\_X509\_SCEP\_MSG\_get\_proxy (PKI\_X509\_PKCS7 \* *msg*)

Returns the proxyAuthenticator attribute from a SCEP message.

#### 1.73.2.9 PKI\_MEM\* PKI\_X509\_SCEP\_MSG\_get\_recipient\_nonce (PKI\_X509\_PKCS7 \* *msg*)

Returns the recipientNonce attribute from a SCEP message.

#### 1.73.2.10 PKI\_MEM\* PKI\_X509\_SCEP\_MSG\_get\_sender\_nonce (PKI\_X509\_PKCS7 \* *msg*)

Returns the senderNonce attribute from a SCEP message.



**1.73.2.11** `SCEP_STATUS` `PKI_X509_SCEP_MSG_get_status (PKI_X509_PKCS7 * msg)`

Returns the pkiStatus attribute from a SCEP message.

**1.73.2.12** `char*` `PKI_X509_SCEP_MSG_get_trans_id (PKI_X509_SCEP_MSG * msg)`**1.73.2.13** `SCEP_MESSAGE_TYPE` `PKI_X509_SCEP_MSG_get_type (PKI_X509_PKCS7 * msg)`

Returns the messageType attribute from a SCEP message.

**1.73.2.14** `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.

**1.73.2.15** `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).

**1.73.2.16** `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.

**1.73.2.17** `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.

**1.73.2.18** `int` `PKI_X509_SCEP_MSG_set_failinfo (PKI_X509_PKCS7 * msg, int fail)`

Sets the failInfo attribute in a SCEP message.

**1.73.2.19** `int` `PKI_X509_SCEP_MSG_set_proxy (PKI_X509_PKCS7 * msg, int auth)`

Sets the proxyAuthenticator attribute from a SCEP message.

**1.73.2.20** `int` `PKI_X509_SCEP_MSG_set_recipient_nonce (PKI_X509_PKCS7 * msg, PKI_MEM * mem)`

Sets the recipientNonce attribute from a SCEP message.

**1.73.2.21** `int` `PKI_X509_SCEP_MSG_set_sender_nonce (PKI_X509_PKCS7 * msg, PKI_MEM * mem)`

Sets the senderNonce attribute in a SCEP message.

**1.73.2.22** `int` `PKI_X509_SCEP_MSG_set_status (PKI_X509_PKCS7 * msg, SCEP_STATUS status)`

Sets the pkiStatus attribute in a SCEP message.

**1.73.2.23 int PKI\_X509\_SCEP\_MSG\_set\_trans\_id (PKI\_X509\_PKCS7 \* msg, PKI\_MEM \* mem)**

Sets the transactionId attribute in a SCEP message.

**1.73.2.24 int PKI\_X509\_SCEP\_MSG\_set\_type (PKI\_X509\_PKCS7 \* msg, SCEP\_MESSAGE\_TYPE type)**

Sets the messageType attribute in a SCEP message.

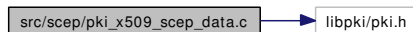
**1.73.3 Variable Documentation****1.73.3.1 SCEP\_CONF\_ATTRIBUTE SCEP\_ATTRIBUTE\_list[SCEP\_CONF\_LIST\_SIZE]**

**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 },
}
```

**1.74 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.*

### 1.74.1 Function Documentation

#### 1.74.1.1 int PKI\_X509\_SCEP\_DATA\_add\_recipient (PKI\_X509\_SCEP\_DATA \* data, PKI\_X509\_CERT \* recipient)

Adds a recipient to a SCEP\_DATA.

#### 1.74.1.2 void PKI\_X509\_SCEP\_DATA\_free (PKI\_X509\_SCEP\_DATA \* data)

Frees the memory associated with a PKI\_X509\_SCEP\_DATA.

#### 1.74.1.3 PKI\_X509\_SCEP\_DATA\* PKI\_X509\_SCEP\_DATA\_new (void)

Generates a new SCEP\_DATA.

#### 1.74.1.4 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.

#### 1.74.1.5 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).

#### 1.74.1.6 int PKI\_X509\_SCEP\_DATA\_set\_recipients (PKI\_X509\_SCEP\_DATA \* data, PKI\_X509\_CERT\_STACK \* sk)

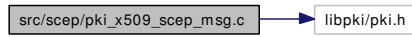
Adds a stack of recipients for a SCEP\_DATA.

#### 1.74.1.7 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.

## 1.75 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.*

### 1.75.1 Function Documentation

**1.75.1.1 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.

**1.75.1.2 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.

**1.75.1.3 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.

#### 1.75.1.4 int PKI\_X509\_SCEP\_MSG\_encode (PKI\_X509\_SCEP\_MSG \* msg, PKI\_X509\_SCEP\_DATA \* data)

Encodes a SCEP\_MSG to a PKCS7 structure.

#### 1.75.1.5 void PKI\_X509\_SCEP\_MSG\_free (PKI\_X509\_SCEP\_MSG \* msg)

Frees the memory associated with a PKI\_X509\_SCEP\_MSG.

#### 1.75.1.6 PKI\_X509\_SCEP\_MSG\* PKI\_X509\_SCEP\_MSG\_new (SCEP\_MESSAGE\_TYPE type)

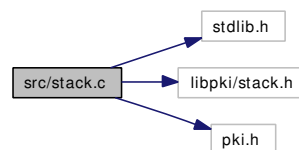
Generates a new PKI\_X509\_SCEP message.

#### 1.75.1.7 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.

## 1.76 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.*

### 1.76.1 Function Documentation

#### 1.76.1.1 `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.

#### 1.76.1.2 `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.

#### 1.76.1.3 `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).

#### 1.76.1.4 `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.

#### 1.76.1.5 `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`.

#### 1.76.1.6 `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.

#### 1.76.1.7 `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`.

#### 1.76.1.8 `PKI_STACK* PKI_STACK_new_null (void)`

#### 1.76.1.9 `PKI_STACK* PKI_STACK_new_type (int type)`

#### 1.76.1.10 `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.

#### 1.76.1.11 `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.

#### 1.76.1.12 `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`.

## 1.77 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)

### 1.77.1 Function Documentation

#### 1.77.1.1 char\* get\_env\_string (const char \* str)

#### 1.77.1.2 char\* PKI\_get\_env (char \* name)

Returns the value of the ENV variable 'name'.

#### 1.77.1.3 int PKI\_set\_env (char \* name, char \* value)

Set the ENV variable 'name' with the value 'value'.

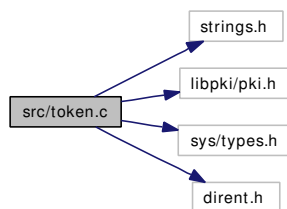
#### 1.77.1.4 int strcmp\_nocase (char \* st1, char \* st2)

#### 1.77.1.5 int strncmp\_nocase (char \* st1, char \* st2, int n)

#### 1.77.1.6 char\* strstr\_nocase (char \* buf, char \* string)

## 1.78 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`)  
*Exports the Token's certificate to a url passed as a string.*
- `int PKI_TOKEN_export_keypair` (`PKI_TOKEN *tk`, `char *url_string`, `int format`)  
*Exports the Token's KeyPair to the passed external URI (string).*
- `int PKI_TOKEN_export_keypair_url` (`PKI_TOKEN *tk`, `URL *url`, `int format`)  
*Exports the Token's keypair to the passed external URI.*
- `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_ALGOR * PKI_TOKEN_get_algor` (`PKI_TOKEN *tk`)  
*Returns the PKI\_ALGORITHM pointer set for the token.*
- `int PKI_TOKEN_get_algor_id` (`PKI_TOKEN *tk`)  
*Returns the algorithm id set for this token.*
- `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\\_ex](#) (PKI\_TOKEN \*tk, PKI\_KEYPARAMS \*kp, char \*label, char \*profile\_s)  
*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.*
- int [PKI\\_TOKEN\\_new\\_keypair\\_url\\_ex](#) (PKI\_TOKEN \*tk, PKI\_KEYPARAMS \*kp, URL \*label, char \*profile\_s)  
*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)  
*Generates a new PKI\_X509\_REQ object.*
- 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)  
*Returns a named profile from the loaded ones.*
- 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, const 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.*

### 1.78.1 Function Documentation

#### 1.78.1.1 int [PKI\\_TOKEN\\_add\\_profile](#) (PKI\_TOKEN \*tk, PKI\_X509\_PROFILE \*profile)

#### 1.78.1.2 int [PKI\\_TOKEN\\_check](#) (PKI\_TOKEN \*tk)

Checks the integrity of a PKI\_TOKEN.

#### 1.78.1.3 PKI\_CRED\* [PKI\\_TOKEN\\_cred\\_cb\\_env](#) (char \*env)

**1.78.1.4 PKI\_CRED\* PKI\_TOKEN\_cred\_cb\_stdin (char \* *prompt*)****1.78.1.5 PKI\_CRED\* PKI\_TOKEN\_cred\_get (PKI\_TOKEN \* *tk*, char \* *st*)**

Returns credentials from the registered callback function.

**1.78.1.6 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.

**1.78.1.7 int PKI\_TOKEN\_del\_url (PKI\_TOKEN \* *tk*, URL \* *url*, int *datatype*)****1.78.1.8 int PKI\_TOKEN\_export\_cert (PKI\_TOKEN \* *tk*, char \* *url\_string*, int *format*)**

Exports the Token's certificate to a url passed as a string.

**1.78.1.9 int PKI\_TOKEN\_export\_keypair (PKI\_TOKEN \* *tk*, char \* *url\_string*, int *format*)**

Exports the Token's KeyPair to the passed external URI (string).

**1.78.1.10 int PKI\_TOKEN\_export\_keypair\_url (PKI\_TOKEN \* *tk*, URL \* *url*, int *format*)**

Exports the Token's keypair to the passed external URI.

**1.78.1.11 int PKI\_TOKEN\_export\_otherCerts (PKI\_TOKEN \* *tk*, char \* *url\_string*, int *format*)**

Export the TOKEN otherCerts to an external URI.

**1.78.1.12 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.

**1.78.1.13 int PKI\_TOKEN\_export\_req (PKI\_TOKEN \* *tk*, char \* *url\_string*, int *format*)**

Export the TOKEN request from the Token to an external URI.

**1.78.1.14 int PKI\_TOKEN\_export\_trustedCerts (PKI\_TOKEN \* *tk*, char \* *url\_string*, int *format*)**

Export the TOKEN trustedCerts to an external URI.

**1.78.1.15 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.

**1.78.1.16 void PKI\_TOKEN\_free\_void (void \* *tk*)****1.78.1.17 PKI\_ALGOR\* PKI\_TOKEN\_get\_algor (PKI\_TOKEN \* *tk*)**

Returns the PKI\_ALGORITHM pointer set for the token.

**1.78.1.18 int PKI\_TOKEN\_get\_algor\_id (PKI\_TOKEN \* *tk*)**

Returns the algorithm id set for this token.

**1.78.1.19 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.

**1.78.1.20 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.

**1.78.1.21 char\* PKI\_TOKEN\_get\_config\_dir (PKI\_TOKEN \* *tk*)**

Get the configuration directory used for the TOKEN operations.

**1.78.1.22 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 or if no credentials has been assigned to the PKI\_TOKEN yet.

**1.78.1.23 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.

**1.78.1.24 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 or if no PKI\_X509\_KEYPAIR has been assigned to the PKI\_TOKEN yet.

**1.78.1.25 char\* PKI\_TOKEN\_get\_name (PKI\_TOKEN \* tk)**

Get TOKEN name.

**1.78.1.26 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.

**1.78.1.27 PKI\_X509\_PKCS12\* PKI\_TOKEN\_get\_p12 (PKI\_TOKEN \* tk, PKI\_CRED \* cred)**

Returns a PKI\_X509\_PKCS12 data structure from the PKI\_TOKEN.

**1.78.1.28 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.

**1.78.1.29 int PKI\_TOKEN\_import\_cert (PKI\_TOKEN \* tk, PKI\_X509\_CERT \* cert, int type, char \* url\_s)**

Imports a certificate into the Token.

**1.78.1.30 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.

**1.78.1.31 int PKI\_TOKEN\_import\_keypair (PKI\_TOKEN \* tk, PKI\_X509\_KEYPAIR \* key, char \* url\_s)**

Imports a Keypair into the Token.

**1.78.1.32 int PKI\_TOKEN\_init (PKI\_TOKEN \* tk, char \* conf\_dir, char \* tk\_name)**

Initialize Token properties (load OIDs and PROFILES).

**1.78.1.33 PKI\_X509\_CERT\* PKI\_TOKEN\_issue\_cert (PKI\_TOKEN \* tk, char \* subject, char \* serial, unsigned long validity, PKI\_X509\_REQ \* req, char \* profile\_s)****1.78.1.34 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.

**1.78.1.35** `PKI_TOKEN* PKI_TOKEN_issue_proxy (PKI_TOKEN * tk, char * subject, char * serial, unsigned long validity, char * profile_s, PKI_TOKEN * px_tk)`

**1.78.1.36** `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.

**1.78.1.37** `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.

**1.78.1.38** `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.

**1.78.1.39** `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.

**1.78.1.40** `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.

**1.78.1.41** `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.



**1.78.1.42 int PKI\_TOKEN\_load\_profiles (PKI\_TOKEN \* tk, char \* urlStr)****1.78.1.43 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.

**1.78.1.44 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.

**1.78.1.45 int PKI\_TOKEN\_login (PKI\_TOKEN \* tk)**

Login into the token (triggers keypair loading).

**1.78.1.46 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.

**1.78.1.47 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 \_ex version allows to specify additional parameters for the keypair generation. For EC keys, the param is a pointer to an integer (nid of the requested curve).

The label (a url string) is needed by some HSM(s).

**1.78.1.48 int PKI\_TOKEN\_new\_keypair\_ex (PKI\_TOKEN \* tk, PKI\_KEYPARAMS \* kp, char \* label, char \* profile\_s)**

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 \_ex version allows to specify additional parameters for the keypair generation. For EC keys, the param is a pointer to an integer (nid of the requested curve).

The label (a url string) is needed by some HSM(s).

**1.78.1.49 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).

**1.78.1.50** `int PKI_TOKEN_new_keypair_url_ex (PKI_TOKEN * tk, PKI_KEYPARAMS * kp, URL * label, char * profile_s)`

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).

**1.78.1.51** `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.

**1.78.1.52** `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.

**1.78.1.53** `int PKI_TOKEN_new_req (PKI_TOKEN * tk, char * subject, char * profile_s)`

Generates a new `PKI_X509_REQ` object.

**1.78.1.54** `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.

**1.78.1.55** `int PKI_TOKEN_print_info (PKI_TOKEN * tk)`

**1.78.1.56** `PKI_X509_PROFILE* PKI_TOKEN_search_profile (PKI_TOKEN * tk, char * profile_s)`

Returns a named profile from the loaded ones.

**1.78.1.57** `int PKI_TOKEN_self_sign (PKI_TOKEN * tk, char * subject, char * serial, unsigned long validity, char * profile_s)`

**1.78.1.58** `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_RIPEND160 -PKI_ALGOR_ECDSA_SHA1`

#### 1.78.1.59 `int PKI_TOKEN_set_algor_by_name (PKI_TOKEN * tk, const 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-RIPEND160`

#### 1.78.1.60 `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.

#### 1.78.1.61 `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.

#### 1.78.1.62 `int PKI_TOKEN_set_config_dir (PKI_TOKEN * tk, char * dir)`

Set the configuration directory to be used for the TOKEN operations.

#### 1.78.1.63 `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.

#### 1.78.1.64 `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.

#### 1.78.1.65 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.

#### 1.78.1.66 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.

#### 1.78.1.67 int PKI\_TOKEN\_set\_req (PKI\_TOKEN \* *tk*, PKI\_X509\_REQ \* *req*)

#### 1.78.1.68 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.

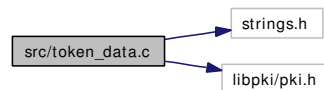
The function returns PKI\_OK in case of success or PKI\_ERR otherwise.

#### 1.78.1.69 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.

## 1.79 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)

### 1.79.1 Function Documentation

**1.79.1.1 PKI\_MEM\* PKI\_TOKEN\_get\_cacert\_data** (PKI\_TOKEN \* *tk*, PKI\_DATA\_FORMAT *format*)

**1.79.1.2 PKI\_MEM\* PKI\_TOKEN\_get\_cert\_data** (PKI\_TOKEN \* *tk*, PKI\_DATA\_FORMAT *format*)

**1.79.1.3 PKI\_MEM\* PKI\_TOKEN\_get\_identity\_data** (PKI\_TOKEN \* *tk*, PKI\_DATA\_FORMAT *format*)

**1.79.1.4 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... ).

**1.79.1.5 PKI\_MEM\_STACK\* PKI\_TOKEN\_get\_otherCerts\_data** (PKI\_TOKEN \* *tk*, PKI\_DATA\_FORMAT *format*)

**1.79.1.6 PKI\_MEM\* PKI\_TOKEN\_get\_privkey\_data** (PKI\_TOKEN \* *tk*, PKI\_DATA\_FORMAT *format*)

**1.79.1.7 PKI\_MEM\* PKI\_TOKEN\_get\_pubkey\_data** (PKI\_TOKEN \* *tk*, PKI\_DATA\_FORMAT *format*)

**1.79.1.8 PKI\_MEM\_STACK\* PKI\_TOKEN\_get\_trustedCerts\_data** (PKI\_TOKEN \* *tk*, PKI\_DATA\_FORMAT *format*)

## 1.80 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`)

### 1.80.1 Function Documentation

**1.80.1.1** `PKI_ID_INFO *` [PKI\\_TOKEN\\_ID\\_INFO\\_get](#) (`PKI_TOKEN * tk`, `int num`)

**1.80.1.2** `PKI_ID_INFO_STACK *` [PKI\\_TOKEN\\_ID\\_INFO\\_list](#) (`PKI_TOKEN * tk`)

**1.80.1.3** `int` [PKI\\_TOKEN\\_ID\\_num](#) (`PKI_TOKEN * tk`)

**1.80.1.4** `int` [PKI\\_TOKEN\\_ID\\_set](#) (`PKI_TOKEN * tk`, `int id`)

## Index

- \_Close
    - sock.c, [31](#)
  - \_Connect
    - sock.c, [31](#)
  - \_Listen
    - sock.c, [31](#)
  - \_Read
    - sock.c, [31](#)
  - \_Select
    - sock.c, [31](#)
  - \_Shutdown
    - sock.c, [32](#)
  - \_Socket
    - sock.c, [32](#)
  - \_Write
    - sock.c, [32](#)
- \_\_LIBPKI\_ERR\_\_
  - pki\_err.c, [95](#)
- \_\_PKI\_PRQP\_LIB\_C\_\_
  - prqp\_lib.c, [125](#)
- \_\_pki\_error
  - pki\_err.c, [96](#)
- \_\_pki\_ssl\_start\_ssl
  - ssl.c, [34](#)
- \_dyn\_create\_callback
  - pthread\_init.c, [88](#)
- \_dyn\_destroy\_callback
  - pthread\_init.c, [88](#)
- \_dyn\_lock\_callback
  - pthread\_init.c, [89](#)
- BAG\_DATATYPE\_ALL
  - pki\_x509\_p12.c, [76](#)
- BAG\_DATATYPE\_CACERT
  - pki\_x509\_p12.c, [76](#)
- BAG\_DATATYPE\_CERT
  - pki\_x509\_p12.c, [76](#)
- BAG\_DATATYPE\_KEYPAIR
  - pki\_x509\_p12.c, [76](#)
- BAG\_DATATYPE\_OTHERCERTS
  - pki\_x509\_p12.c, [76](#)
- bag\_datatype\_st
  - pki\_x509\_p12.c, [76](#)
- BAG\_DATATYPE\_UNKNOWN
  - pki\_x509\_p12.c, [76](#)
- BUFF\_MAX\_SIZE
  - http\_s.c, [24](#)
  - ssl.c, [34](#)
  - url.c, [37](#)
- CERT\_IDENTIFIER\_cmp
  - prqp\_lib.c, [125](#)
- 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, [88](#)
- CRYPTO\_READ
  - pthread\_init.c, [88](#)
- CRYPTO\_UNLOCK
  - pthread\_init.c, [88](#)
- CRYPTO\_WRITE
  - pthread\_init.c, [88](#)
- d2i\_CERT\_REQ\_MSG\_bio
  - cms\_cert\_req.c, [2](#)
- d2i\_OCSP\_REQ\_bio
  - pki\_ocsp\_req.c, [52](#)
- d2i\_PKI\_OCSP\_RESP\_bio
  - pki\_ocsp\_resp.c, [55](#)
- d2i\_PRQP\_REQ\_bio
  - prqp\_bio.c, [122](#)

- d2i\_PRQP\_RESP\_bio
  - prqp\_bio.c, 122
- 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, 143
- Gethostbyname
  - sock.c, 32
- h\_errno
  - sock.c, 33
- http\_client.c
  - PKI\_X509\_PRQP\_RESP\_get\_http, 121
- http\_s.c
  - BUFF\_MAX\_SIZE, 24
  - PKI\_HTTP\_free, 24
  - PKI\_HTTP\_GET\_data, 24
  - PKI\_HTTP\_GET\_data\_socket, 24
  - PKI\_HTTP\_GET\_data\_url, 24
  - PKI\_HTTP\_get\_header, 24
  - PKI\_HTTP\_get\_header\_txt, 25
  - PKI\_HTTP\_get\_message, 25
  - PKI\_HTTP\_get\_socket, 25
  - PKI\_HTTP\_get\_url, 25
  - PKI\_HTTP\_new, 25
  - PKI\_HTTP\_POST\_data, 25
  - PKI\_HTTP\_POST\_data\_socket, 25
  - PKI\_HTTP\_POST\_data\_url, 25
- i2d\_CERT\_REQ\_MSG\_bio
  - cms\_cert\_req.c, 2
- i2d\_OCSP\_REQ\_bio
  - pki\_ocsp\_req.c, 52
- i2d\_PKI\_OCSP\_RESP\_bio
  - pki\_ocsp\_resp.c, 55
- i2d\_PRQP\_REQ\_bio
  - prqp\_bio.c, 122
- i2d\_PRQP\_RESP\_bio
  - prqp\_bio.c, 122
- inet\_close
  - sock.c, 32
- inet\_connect
  - sock.c, 32
- LIBPKI\_OS\_DETAILS
  - pki\_init.c, 97
- LIBXML\_MIN\_VERSION
  - pki\_config.c, 91
- LISTENQ
  - sock.c, 31
- lock\_cs
  - pthread\_init.c, 89
- logXmlMessages
  - pki\_config.c, 91
- 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, 97
  - pki\_x509\_cert.c, 68
- 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, 52
- PEM\_read\_bio\_PKCS12
  - pki\_x509\_p12.c, 76
- PEM\_read\_bio\_PKI\_OCSP\_RESP
  - pki\_ocsp\_resp.c, 55
- PEM\_read\_bio\_PRQP\_REQ
  - prqp\_bio.c, 122
- PEM\_read\_bio\_PRQP\_RESP
  - prqp\_bio.c, 122
- PEM\_write\_bio\_CERT\_REQ\_MSG
  - cms\_cert\_req.c, 3
- PEM\_write\_bio\_OCSP\_REQ
  - pki\_ocsp\_req.c, 52
- PEM\_write\_bio\_PKCS12
  - pki\_x509\_p12.c, 76
- PEM\_write\_bio\_PKI\_OCSP\_RESP
  - pki\_ocsp\_resp.c, 55
- PEM\_write\_bio\_PRQP\_REQ
  - prqp\_bio.c, 122
- PEM\_write\_bio\_PRQP\_RESP
  - prqp\_bio.c, 122
- 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, 40
  - PKI\_ALGOR\_get\_by\_name, 40
  - PKI\_ALGOR\_get\_digest, 40
  - PKI\_ALGOR\_get\_digest\_id, 40
  - PKI\_ALGOR\_get\_id, 41
  - PKI\_ALGOR\_get\_parsed, 41
  - PKI\_ALGOR\_get\_scheme, 41
  - PKI\_ALGOR\_ID\_txt, 41
  - PKI\_ALGOR\_list, 41
  - PKI\_ALGOR\_LIST\_DSA, 42
  - PKI\_ALGOR\_LIST\_ECDSA, 42
  - PKI\_ALGOR\_LIST\_RSA, 42
  - PKI\_ALGOR\_list\_size, 41
  - PKI\_DIGEST\_ALG\_get, 41
  - PKI\_DIGEST\_ALG\_get\_by\_key, 41
  - PKI\_DIGEST\_ALG\_get\_by\_name, 41
  - PKI\_DIGEST\_ALG\_get\_parsed, 41
  - PKI\_DIGEST\_ALG\_LIST, 42
  - PKI\_DIGEST\_ALG\_list, 41
- PKI\_ALGOR\_get
  - pki\_algor.c, 40
- PKI\_ALGOR\_get\_by\_name
  - pki\_algor.c, 40
- PKI\_ALGOR\_get\_digest
  - pki\_algor.c, 40
- PKI\_ALGOR\_get\_digest\_id
  - pki\_algor.c, 40
- PKI\_ALGOR\_get\_id
  - pki\_algor.c, 41
- PKI\_ALGOR\_get\_parsed
  - pki\_algor.c, 41
- PKI\_ALGOR\_get\_scheme
  - pki\_algor.c, 41
- PKI\_ALGOR\_ID\_txt
  - pki\_algor.c, 41
- PKI\_ALGOR\_list
  - pki\_algor.c, 41
- PKI\_ALGOR\_LIST\_DSA
  - pki\_algor.c, 42
- PKI\_ALGOR\_LIST\_ECDSA
  - pki\_algor.c, 42
- PKI\_ALGOR\_LIST\_RSA
  - pki\_algor.c, 42
- PKI\_ALGOR\_list\_size
  - pki\_algor.c, 41
- pki\_algorithm.c
  - PKI\_ALGORITHM\_free, 43
  - PKI\_ALGORITHM\_new, 43
  - PKI\_ALGORITHM\_new\_digest, 43
  - PKI\_ALGORITHM\_new\_type, 43
- PKI\_ALGORITHM\_free
  - pki\_algorithm.c, 43
- PKI\_ALGORITHM\_new
  - pki\_algorithm.c, 43
- PKI\_ALGORITHM\_new\_digest
  - pki\_algorithm.c, 43
- PKI\_ALGORITHM\_new\_type
  - pki\_algorithm.c, 43
- PKI\_clock\_gettime
  - pki\_threads\_vars.c, 113
- PKI\_CMS\_REQ
  - cms\_simple.c, 3
- PKI\_CMS\_RESP
  - cms\_simple.c, 3
- PKI\_COND\_broadcast
  - pki\_threads\_vars.c, 113
- PKI\_COND\_destroy
  - pki\_threads\_vars.c, 113
- PKI\_COND\_free
  - pki\_threads\_vars.c, 113
- PKI\_COND\_init
  - pki\_threads\_vars.c, 113
- PKI\_COND\_new
  - pki\_threads\_vars.c, 113
- PKI\_COND\_signal
  - pki\_threads\_vars.c, 113
- PKI\_COND\_timedwait
  - pki\_threads\_vars.c, 113
- PKI\_COND\_wait
  - pki\_threads\_vars.c, 114
- pki\_config.c
  - LIBXML\_MIN\_VERSION, 91
  - logXmlMessages, 91
  - PKI\_CONFIG\_add\_node, 91
  - PKI\_CONFIG\_ELEMENT\_add\_attribute, 91
  - PKI\_CONFIG\_ELEMENT\_add\_child, 91

- PKI\_CONFIG\_ELEMENT\_add\_child\_el, 92
- PKI\_CONFIG\_ELEMENT\_new, 92
- PKI\_CONFIG\_find, 92
- PKI\_CONFIG\_find\_all, 92
- PKI\_CONFIG\_free, 92
- PKI\_CONFIG\_free\_void, 92
- PKI\_CONFIG\_get\_attribute\_value, 92
- PKI\_CONFIG\_get\_element, 92
- PKI\_CONFIG\_get\_element\_child, 92
- PKI\_CONFIG\_get\_element\_children, 92
- PKI\_CONFIG\_get\_element\_name, 92
- PKI\_CONFIG\_get\_element\_next, 92
- PKI\_CONFIG\_get\_element\_prev, 93
- PKI\_CONFIG\_get\_element\_stack, 93
- PKI\_CONFIG\_get\_element\_value, 93
- PKI\_CONFIG\_get\_elements\_num, 93
- PKI\_CONFIG\_get\_root, 93
- PKI\_CONFIG\_get\_search\_paths, 93
- PKI\_CONFIG\_get\_stack\_value, 93
- PKI\_CONFIG\_get\_value, 93
- PKI\_CONFIG\_load, 93
- PKI\_CONFIG\_load\_all, 93
- PKI\_CONFIG\_load\_dir, 93
- PKI\_CONFIG\_OID\_load, 93
- PKI\_CONFIG\_OID\_search, 94
- PKI\_DEF\_CONF\_DIRS\_SIZE, 91
- xmlErrorPtr, 91
- PKI\_CONFIG\_add\_node
  - pki\_config.c, 91
- PKI\_CONFIG\_ELEMENT\_add\_attribute
  - pki\_config.c, 91
- PKI\_CONFIG\_ELEMENT\_add\_child
  - pki\_config.c, 91
- PKI\_CONFIG\_ELEMENT\_add\_child\_el
  - pki\_config.c, 92
- PKI\_CONFIG\_ELEMENT\_new
  - pki\_config.c, 92
- PKI\_CONFIG\_find
  - pki\_config.c, 92
- PKI\_CONFIG\_find\_all
  - pki\_config.c, 92
- PKI\_CONFIG\_free
  - pki\_config.c, 92
- PKI\_CONFIG\_free\_void
  - pki\_config.c, 92
- PKI\_CONFIG\_get\_attribute\_value
  - pki\_config.c, 92
- PKI\_CONFIG\_get\_element
  - pki\_config.c, 92
- PKI\_CONFIG\_get\_element\_child
  - pki\_config.c, 92
- PKI\_CONFIG\_get\_element\_children
  - pki\_config.c, 92
- PKI\_CONFIG\_get\_element\_name
  - pki\_config.c, 92
- PKI\_CONFIG\_get\_element\_next
  - pki\_config.c, 92
- PKI\_CONFIG\_get\_element\_prev
  - pki\_config.c, 93
- PKI\_CONFIG\_get\_element\_stack
  - pki\_config.c, 93
- PKI\_CONFIG\_get\_element\_value
  - pki\_config.c, 93
- PKI\_CONFIG\_get\_elements\_num
  - pki\_config.c, 93
- PKI\_CONFIG\_get\_root
  - pki\_config.c, 93
- PKI\_CONFIG\_get\_search\_paths
  - pki\_config.c, 93
- PKI\_CONFIG\_get\_stack\_value
  - pki\_config.c, 93
- PKI\_CONFIG\_get\_value
  - pki\_config.c, 93
- PKI\_CONFIG\_load
  - pki\_config.c, 93
- PKI\_CONFIG\_load\_all
  - pki\_config.c, 93
- PKI\_CONFIG\_load\_dir
  - pki\_config.c, 93
- PKI\_CONFIG\_OID\_load
  - pki\_config.c, 93
- PKI\_CONFIG\_OID\_search
  - pki\_config.c, 94
- pki\_cred.c
  - PKI\_CRED\_dup, 94
  - PKI\_CRED\_free, 94
  - PKI\_CRED\_get\_ssl, 94
  - PKI\_CRED\_new, 95
  - PKI\_CRED\_new\_null, 95
  - PKI\_CRED\_set\_ssl, 95
- PKI\_CRED\_dup
  - pki\_cred.c, 94
- PKI\_CRED\_free
  - pki\_cred.c, 94
- PKI\_CRED\_get\_ssl
  - pki\_cred.c, 94
- PKI\_CRED\_new
  - pki\_cred.c, 95
- PKI\_CRED\_new\_null
  - pki\_cred.c, 95
- PKI\_CRED\_set\_ssl
  - pki\_cred.c, 95
- PKI\_DEF\_CONF\_DIRS\_SIZE
  - pki\_config.c, 91
- pki\_digest.c
  - PKI\_DIGEST\_free, 44
  - PKI\_DIGEST\_get\_parsed, 44
  - PKI\_DIGEST\_get\_size, 44

- PKI\_DIGEST\_MEM\_new, 44
- PKI\_DIGEST\_MEM\_new\_by\_name, 44
- PKI\_DIGEST\_new, 44
- PKI\_DIGEST\_new\_by\_name, 44
- PKI\_DIGEST\_URL\_new, 45
- PKI\_DIGEST\_URL\_new\_by\_name, 45
- PKI\_DIGEST\_ALG\_get
  - pki\_algor.c, 41
- PKI\_DIGEST\_ALG\_get\_by\_key
  - pki\_algor.c, 41
- PKI\_DIGEST\_ALG\_get\_by\_name
  - pki\_algor.c, 41
- PKI\_DIGEST\_ALG\_get\_parsed
  - pki\_algor.c, 41
- PKI\_DIGEST\_ALG\_LIST
  - pki\_algor.c, 42
- PKI\_DIGEST\_ALG\_list
  - pki\_algor.c, 41
- PKI\_DIGEST\_free
  - pki\_digest.c, 44
- PKI\_DIGEST\_get\_parsed
  - pki\_digest.c, 44
- PKI\_DIGEST\_get\_size
  - pki\_digest.c, 44
- PKI\_DIGEST\_MEM\_new
  - pki\_digest.c, 44
- PKI\_DIGEST\_MEM\_new\_by\_name
  - pki\_digest.c, 44
- PKI\_DIGEST\_new
  - pki\_digest.c, 44
- PKI\_DIGEST\_new\_by\_name
  - pki\_digest.c, 44
- PKI\_DIGEST\_URL\_new
  - pki\_digest.c, 45
- PKI\_DIGEST\_URL\_new\_by\_name
  - pki\_digest.c, 45
- PKI\_DISCOVER\_get\_resp
  - prqp\_srv.c, 132
- PKI\_DISCOVER\_get\_resp\_url
  - prqp\_srv.c, 132
- pki\_err.c
  - \_\_LIBPKI\_ERR\_\_, 95
  - \_\_pki\_error, 96
  - pki\_err\_stack, 96
- pki\_err\_stack
  - pki\_err.c, 96
- PKI\_Free
  - pki\_mem.c, 100
- PKI\_get\_all\_tokens
  - pki\_init.c, 96
- PKI\_get\_all\_tokens\_dir
  - pki\_init.c, 96
- PKI\_get\_ca\_resources
  - prqp\_srv.c, 132
- PKI\_get\_ca\_service
  - prqp\_srv.c, 132
- PKI\_get\_ca\_service\_sk
  - prqp\_srv.c, 132
- PKI\_get\_cert\_service\_sk
  - prqp\_srv.c, 133
- PKI\_get\_env
  - support.c, 143
- PKI\_get\_init\_status
  - pki\_init.c, 96
- PKI\_get\_value
  - pki\_x509\_io.c, 15
- PKI\_HTTP\_free
  - http\_s.c, 24
- PKI\_HTTP\_GET\_data
  - http\_s.c, 24
- PKI\_HTTP\_GET\_data\_socket
  - http\_s.c, 24
- PKI\_HTTP\_GET\_data\_url
  - http\_s.c, 24
- PKI\_HTTP\_get\_header
  - http\_s.c, 24
- PKI\_HTTP\_get\_header\_txt
  - http\_s.c, 25
- PKI\_HTTP\_get\_message
  - http\_s.c, 25
- PKI\_HTTP\_get\_socket
  - http\_s.c, 25
- PKI\_HTTP\_get\_url
  - http\_s.c, 25
- PKI\_HTTP\_new
  - http\_s.c, 25
- PKI\_HTTP\_POST\_data
  - http\_s.c, 25
- PKI\_HTTP\_POST\_data\_socket
  - http\_s.c, 25
- PKI\_HTTP\_POST\_data\_url
  - http\_s.c, 25
- pki\_id.c
  - PKI\_ID\_get, 45
  - PKI\_ID\_get\_by\_name, 45
  - PKI\_ID\_get\_txt, 45
- PKI\_ID\_get
  - pki\_id.c, 45
- PKI\_ID\_get\_by\_name
  - pki\_id.c, 45
- PKI\_ID\_get\_txt
  - pki\_id.c, 45
- pki\_init.c
  - LIBPKI\_OS\_DETAILS, 97
  - NID\_proxyCertInfo, 97
  - PKI\_get\_all\_tokens, 96
  - PKI\_get\_all\_tokens\_dir, 96
  - PKI\_get\_init\_status, 96

- PKI\_init\_all, 97
- PKI\_list\_all\_id, 97
- PKI\_list\_all\_tokens, 97
- PKI\_list\_all\_tokens\_dir, 97
- PKI\_init\_all
  - pki\_init.c, 97
- pki\_integer.c
  - PKI\_INTEGER\_cmp, 46
  - PKI\_INTEGER\_dup, 46
  - PKI\_INTEGER\_free, 46
  - PKI\_INTEGER\_free\_void, 46
  - PKI\_INTEGER\_get\_parsed, 46
  - PKI\_INTEGER\_new, 47
  - PKI\_INTEGER\_new\_bin, 47
  - PKI\_INTEGER\_new\_char, 47
  - PKI\_INTEGER\_print, 47
  - PKI\_INTEGER\_print\_fp, 47
- PKI\_INTEGER\_cmp
  - pki\_integer.c, 46
- PKI\_INTEGER\_dup
  - pki\_integer.c, 46
- PKI\_INTEGER\_free
  - pki\_integer.c, 46
- PKI\_INTEGER\_free\_void
  - pki\_integer.c, 46
- PKI\_INTEGER\_get\_parsed
  - pki\_integer.c, 46
- PKI\_INTEGER\_new
  - pki\_integer.c, 47
- PKI\_INTEGER\_new\_bin
  - pki\_integer.c, 47
- PKI\_INTEGER\_new\_char
  - pki\_integer.c, 47
- PKI\_INTEGER\_print
  - pki\_integer.c, 47
- PKI\_INTEGER\_print\_fp
  - pki\_integer.c, 47
- pki\_keypair.c
  - PKI\_X509\_KEYPAIR\_free, 48
  - PKI\_X509\_KEYPAIR\_free\_void, 48
  - PKI\_X509\_KEYPAIR\_get\_algor, 48
  - PKI\_X509\_KEYPAIR\_get\_p8, 48
  - PKI\_X509\_KEYPAIR\_get\_parsed, 49
  - PKI\_X509\_KEYPAIR\_get\_scheme, 49
  - PKI\_X509\_KEYPAIR\_get\_size, 49
  - PKI\_X509\_KEYPAIR\_new, 49
  - PKI\_X509\_KEYPAIR\_new\_kp, 49
  - PKI\_X509\_KEYPAIR\_new\_null, 49
  - PKI\_X509\_KEYPAIR\_new\_p8, 49
  - PKI\_X509\_KEYPAIR\_new\_url, 49
  - PKI\_X509\_KEYPAIR\_new\_url\_kp, 49
  - PKI\_X509\_KEYPAIR\_pub\_digest, 49
  - PKI\_X509\_KEYPAIR\_VALUE\_get\_algor, 49
  - PKI\_X509\_KEYPAIR\_VALUE\_get\_scheme, 49
  - PKI\_X509\_KEYPAIR\_VALUE\_get\_size, 50
  - PKI\_X509\_KEYPAIR\_VALUE\_pub\_digest, 50
- 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\_keyparams.c
  - PKI\_KEYPARAMS\_free, 50
  - PKI\_KEYPARAMS\_get\_type, 50
  - PKI\_KEYPARAMS\_new, 50
- PKI\_KEYPARAMS\_free
  - pki\_keyparams.c, 50
- PKI\_KEYPARAMS\_get\_type
  - pki\_keyparams.c, 50
- PKI\_KEYPARAMS\_new
  - pki\_keyparams.c, 50
- PKI\_list\_all\_id
  - pki\_init.c, 97
- PKI\_list\_all\_tokens
  - pki\_init.c, 97
- PKI\_list\_all\_tokens\_dir
  - pki\_init.c, 97
- PKI\_log
  - pki\_log.c, 98
- pki\_log.c
  - PKI\_log, 98
  - PKI\_log\_debug\_simple, 98
  - PKI\_log\_end, 98
  - PKI\_log\_err\_simple, 98
  - PKI\_log\_init, 98
- PKI\_log\_debug\_simple
  - pki\_log.c, 98
- PKI\_log\_end
  - pki\_log.c, 98
- PKI\_log\_err\_simple
  - pki\_log.c, 98
- PKI\_log\_init
  - pki\_log.c, 98
- PKI\_Malloc
  - pki\_mem.c, 100
- pki\_mem.c
  - PKI\_Free, 100
  - PKI\_Malloc, 100
  - PKI\_MEM\_add, 100
  - PKI\_MEM\_B64\_decode, 100
  - PKI\_MEM\_B64\_encode, 100

- PKI\_MEM\_dup, [100](#)
- PKI\_MEM\_fprintf, [100](#)
- PKI\_MEM\_free, [100](#)
- PKI\_MEM\_free\_void, [100](#)
- PKI\_MEM\_get\_data, [100](#)
- PKI\_MEM\_get\_size, [100](#)
- PKI\_MEM\_grow, [101](#)
- PKI\_MEM\_new, [101](#)
- PKI\_MEM\_new\_bio, [101](#)
- PKI\_MEM\_new\_data, [101](#)
- PKI\_MEM\_new\_func, [101](#)
- PKI\_MEM\_new\_func\_bio, [101](#)
- PKI\_MEM\_new\_membio, [101](#)
- PKI\_MEM\_new\_null, [101](#)
- PKI\_MEM\_printf, [101](#)
- PKI\_MEM\_url\_decode, [101](#)
- PKI\_MEM\_url\_encode, [101](#)
- PKI\_ZFree, [101](#)
- PKI\_ZFree\_str, [102](#)
- PKI\_MEM\_add
  - pki\_mem.c, [100](#)
- PKI\_MEM\_B64\_decode
  - pki\_mem.c, [100](#)
- PKI\_MEM\_B64\_encode
  - pki\_mem.c, [100](#)
- PKI\_MEM\_dup
  - pki\_mem.c, [100](#)
- PKI\_MEM\_fprintf
  - pki\_mem.c, [100](#)
- PKI\_MEM\_free
  - pki\_mem.c, [100](#)
- PKI\_MEM\_free\_void
  - pki\_mem.c, [100](#)
- PKI\_MEM\_get\_data
  - pki\_mem.c, [100](#)
- PKI\_MEM\_get\_size
  - pki\_mem.c, [100](#)
- PKI\_MEM\_grow
  - pki\_mem.c, [101](#)
- PKI\_MEM\_new
  - pki\_mem.c, [101](#)
- PKI\_MEM\_new\_bio
  - pki\_mem.c, [101](#)
- PKI\_MEM\_new\_data
  - pki\_mem.c, [101](#)
- PKI\_MEM\_new\_func
  - pki\_mem.c, [101](#)
- PKI\_MEM\_new\_func\_bio
  - pki\_mem.c, [101](#)
- PKI\_MEM\_new\_membio
  - pki\_mem.c, [101](#)
- PKI\_MEM\_new\_null
  - pki\_mem.c, [101](#)
- PKI\_MEM\_printf
  - pki\_mem.c, [101](#)
- PKI\_MEM\_url\_decode
  - pki\_mem.c, [101](#)
- PKI\_MEM\_url\_encode
  - pki\_mem.c, [101](#)
- PKI\_MSG\_CMS\_write
  - cms\_simple.c, [3](#)
- pki\_msg\_req.c
  - PKI\_MSG\_REQ\_add\_data, [104](#)
  - PKI\_MSG\_REQ\_add\_recipient, [104](#)
  - PKI\_MSG\_REQ\_clear\_data, [104](#)
  - PKI\_MSG\_REQ\_clear\_recipients, [104](#)
  - PKI\_MSG\_REQ\_encode, [104](#)
  - PKI\_MSG\_REQ\_free, [104](#)
  - PKI\_MSG\_REQ\_get\_action, [104](#)
  - PKI\_MSG\_REQ\_get\_cacert, [104](#)
  - PKI\_MSG\_REQ\_get\_encoded, [104](#)
  - PKI\_MSG\_REQ\_get\_keypair, [104](#)
  - PKI\_MSG\_REQ\_get\_loa, [105](#)
  - PKI\_MSG\_REQ\_get\_proto, [105](#)
  - PKI\_MSG\_REQ\_get\_recipients, [105](#)
  - PKI\_MSG\_REQ\_get\_signer, [105](#)
  - PKI\_MSG\_REQ\_get\_subject, [105](#)
  - PKI\_MSG\_REQ\_get\_template, [105](#)
  - PKI\_MSG\_REQ\_new, [105](#)
  - PKI\_MSG\_REQ\_new\_null, [105](#)
  - PKI\_MSG\_REQ\_new\_tk, [105](#)
  - PKI\_MSG\_REQ\_replace\_data, [105](#)
  - PKI\_MSG\_REQ\_SCEP\_new, [105](#)
  - PKI\_MSG\_REQ\_SCEP\_send, [105](#)
  - PKI\_MSG\_REQ\_send, [106](#)
  - PKI\_MSG\_REQ\_set\_action, [106](#)
  - PKI\_MSG\_REQ\_set\_cacert, [106](#)
  - PKI\_MSG\_REQ\_set\_keypair, [106](#)
  - PKI\_MSG\_REQ\_set\_loa, [106](#)
  - PKI\_MSG\_REQ\_set\_proto, [106](#)
  - PKI\_MSG\_REQ\_set\_recipients, [106](#)
  - PKI\_MSG\_REQ\_set\_signer, [106](#)
  - PKI\_MSG\_REQ\_set\_subject, [106](#)
  - PKI\_MSG\_REQ\_set\_template, [106](#)
- PKI\_MSG\_REQ\_add\_data
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_add\_recipient
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_clear\_data
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_clear\_recipients
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_encode
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_free
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_get\_action
  - pki\_msg\_req.c, [104](#)

- PKI\_MSG\_REQ\_get\_cacert
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_get\_encoded
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_get\_keypair
  - pki\_msg\_req.c, [104](#)
- PKI\_MSG\_REQ\_get\_loa
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_get\_proto
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_get\_recipients
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_get\_signer
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_get\_subject
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_get\_template
  - pki\_msg\_req.c, [105](#)
- pki\_msg\_req\_io.c
  - PKI\_MSG\_REQ\_put, [6](#)
  - PKI\_MSG\_REQ\_put\_mem, [6](#)
- PKI\_MSG\_REQ\_new
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_new\_null
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_new\_tk
  - pki\_msg\_req.c, [105](#)
- 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, [105](#)
- PKI\_MSG\_REQ\_SCEP\_new
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_SCEP\_send
  - pki\_msg\_req.c, [105](#)
- PKI\_MSG\_REQ\_send
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_action
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_cacert
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_keypair
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_loa
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_proto
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_recipients
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_signer
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_subject
  - pki\_msg\_req.c, [106](#)
- PKI\_MSG\_REQ\_set\_template
  - pki\_msg\_req.c, [106](#)
- pki\_msg\_resp.c
  - PKI\_MSG\_RESP\_add\_data, [108](#)
  - PKI\_MSG\_RESP\_add\_recipient, [108](#)
  - PKI\_MSG\_RESP\_clear\_data, [108](#)
  - PKI\_MSG\_RESP\_clear\_recipients, [109](#)
  - PKI\_MSG\_RESP\_encode, [109](#)
  - PKI\_MSG\_RESP\_free, [109](#)
  - PKI\_MSG\_RESP\_get\_action, [109](#)
  - PKI\_MSG\_RESP\_get\_cacert, [109](#)
  - PKI\_MSG\_RESP\_get\_encoded, [109](#)
  - PKI\_MSG\_RESP\_get\_issued\_cert, [109](#)
  - PKI\_MSG\_RESP\_get\_keypair, [109](#)
  - PKI\_MSG\_RESP\_get\_proto, [109](#)
  - PKI\_MSG\_RESP\_get\_recipients, [109](#)
  - PKI\_MSG\_RESP\_get\_signer, [109](#)
  - PKI\_MSG\_RESP\_get\_status, [109](#)
  - PKI\_MSG\_RESP\_new, [109](#)
  - PKI\_MSG\_RESP\_new\_null, [110](#)
  - PKI\_MSG\_RESP\_new\_tk, [110](#)
  - PKI\_MSG\_RESP\_replace\_data, [110](#)
  - PKI\_MSG\_RESP\_SCEP\_new, [110](#)
  - PKI\_MSG\_RESP\_set\_action, [110](#)
  - PKI\_MSG\_RESP\_set\_cacert, [110](#)
  - PKI\_MSG\_RESP\_set\_issued\_cert, [110](#)
  - PKI\_MSG\_RESP\_set\_keypair, [110](#)
  - PKI\_MSG\_RESP\_set\_proto, [110](#)
  - PKI\_MSG\_RESP\_set\_recipients, [110](#)
  - PKI\_MSG\_RESP\_set\_signer, [110](#)
  - PKI\_MSG\_RESP\_set\_status, [111](#)
- PKI\_MSG\_RESP\_add\_data
  - pki\_msg\_resp.c, [108](#)
- PKI\_MSG\_RESP\_add\_recipient
  - pki\_msg\_resp.c, [108](#)
- PKI\_MSG\_RESP\_clear\_data
  - pki\_msg\_resp.c, [108](#)
- PKI\_MSG\_RESP\_clear\_recipients
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_encode
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_free
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_get\_action
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_get\_cacert
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_get\_encoded
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_get\_issued\_cert
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_get\_keypair
  - pki\_msg\_resp.c, [109](#)



- PKI\_MSG\_RESP\_get\_proto
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_get\_recipients
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_get\_signer
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_get\_status
  - pki\_msg\_resp.c, [109](#)
- pki\_msg\_resp\_io.c
  - PKI\_MSG\_RESP\_put, [6](#)
  - PKI\_MSG\_RESP\_put\_mem, [6](#)
- PKI\_MSG\_RESP\_new
  - pki\_msg\_resp.c, [109](#)
- PKI\_MSG\_RESP\_new\_null
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_new\_tk
  - pki\_msg\_resp.c, [110](#)
- 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, [110](#)
- PKI\_MSG\_RESP\_SCEP\_new
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_set\_action
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_set\_cacert
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_set\_issued\_cert
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_set\_keypair
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_set\_proto
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_set\_recipients
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_set\_signer
  - pki\_msg\_resp.c, [110](#)
- PKI\_MSG\_RESP\_set\_status
  - pki\_msg\_resp.c, [111](#)
- PKI\_MUTEX\_acquire
  - pki\_threads\_vars.c, [114](#)
- PKI\_MUTEX\_destroy
  - pki\_threads\_vars.c, [114](#)
- PKI\_MUTEX\_free
  - pki\_threads\_vars.c, [114](#)
- PKI\_MUTEX\_init
  - pki\_threads\_vars.c, [114](#)
- PKI\_MUTEX\_new
  - pki\_threads\_vars.c, [114](#)
- PKI\_MUTEX\_release
  - pki\_threads\_vars.c, [114](#)
- PKI\_NET\_accept
  - sock.c, [32](#)
- PKI\_NET\_close
  - sock.c, [32](#)
- PKI\_NET\_get\_data
  - sock.c, [32](#)
- PKI\_NET\_listen
  - sock.c, [32](#)
- PKI\_NET\_open
  - sock.c, [32](#)
- PKI\_NET\_read
  - sock.c, [32](#)
- PKI\_NET\_socket
  - sock.c, [32](#)
- PKI\_NET\_write
  - sock.c, [32](#)
- PKI\_OCSP\_nonce\_check
  - pki\_ocsp\_req.c, [52](#)
- pki\_ocsp\_req.c
  - d2i\_OCSP\_REQ\_bio, [52](#)
  - i2d\_OCSP\_REQ\_bio, [52](#)
  - PEM\_read\_bio\_OCSP\_REQ, [52](#)
  - PEM\_write\_bio\_OCSP\_REQ, [52](#)
  - PKI\_OCSP\_nonce\_check, [52](#)
  - PKI\_X509\_OCSP\_REQ\_add\_cert, [52](#)
  - PKI\_X509\_OCSP\_REQ\_add\_longlong, [53](#)
  - PKI\_X509\_OCSP\_REQ\_add\_nonce, [53](#)
  - PKI\_X509\_OCSP\_REQ\_add\_serial, [53](#)
  - PKI\_X509\_OCSP\_REQ\_add\_txt, [53](#)
  - PKI\_X509\_OCSP\_REQ\_DATA\_sign, [53](#)
  - PKI\_X509\_OCSP\_REQ\_elements, [53](#)
  - PKI\_X509\_OCSP\_REQ\_free, [53](#)
  - PKI\_X509\_OCSP\_REQ\_free\_void, [53](#)
  - PKI\_X509\_OCSP\_REQ\_get\_cid, [53](#)
  - PKI\_X509\_OCSP\_REQ\_get\_data, [53](#)
  - PKI\_X509\_OCSP\_REQ\_get\_parsed, [53](#)
  - PKI\_X509\_OCSP\_REQ\_get\_serial, [54](#)
  - PKI\_X509\_OCSP\_REQ\_new, [54](#)
  - PKI\_X509\_OCSP\_REQ\_new\_null, [54](#)
  - PKI\_X509\_OCSP\_REQ\_print\_parsed, [54](#)
  - PKI\_X509\_OCSP\_REQ\_sign, [54](#)
  - PKI\_X509\_OCSP\_REQ\_sign\_tk, [54](#)
- 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, 55
  - i2d\_PKI\_OCSP\_RESP\_bio, 55
  - PEM\_read\_bio\_PKI\_OCSP\_RESP, 55
  - PEM\_write\_bio\_PKI\_OCSP\_RESP, 55
  - PKI\_OCSP\_RESP\_free, 56
  - PKI\_OCSP\_RESP\_new, 56
  - PKI\_X509\_OCSP\_RESP\_add, 56
  - PKI\_X509\_OCSP\_RESP\_copy\_nonce, 56
  - PKI\_X509\_OCSP\_RESP\_DATA\_sign, 56
  - PKI\_X509\_OCSP\_RESP\_free, 56
  - PKI\_X509\_OCSP\_RESP\_free\_void, 56
  - PKI\_X509\_OCSP\_RESP\_get\_data, 56
  - PKI\_X509\_OCSP\_RESP\_get\_parsed, 56
  - PKI\_X509\_OCSP\_RESP\_new, 56
  - PKI\_X509\_OCSP\_RESP\_new\_null, 56
  - PKI\_X509\_OCSP\_RESP\_print\_parsed, 56
  - PKI\_X509\_OCSP\_RESP\_set\_status, 56
  - PKI\_X509\_OCSP\_RESP\_sign, 57
  - PKI\_X509\_OCSP\_RESP\_sign\_tk, 57
- PKI\_OCSP\_RESP\_free
  - pki\_ocsp\_resp.c, 56
- 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, 56
- pki\_oid.c
  - PKI\_OID\_cmp, 58
  - PKI\_OID\_dup, 58
  - PKI\_OID\_free, 58
  - PKI\_OID\_free\_void, 58
  - PKI\_OID\_get, 58
  - PKI\_OID\_get\_descr, 58
  - PKI\_OID\_get\_id, 58
  - PKI\_OID\_get\_str, 58
  - PKI\_OID\_load, 58
  - PKI\_OID\_new, 58
  - PKI\_OID\_new\_id, 58
  - PKI\_OID\_new\_text, 59
- PKI\_OID\_cmp
  - pki\_oid.c, 58
- PKI\_OID\_dup
  - pki\_oid.c, 58
- PKI\_OID\_free
  - pki\_oid.c, 58
- PKI\_OID\_free\_void
  - pki\_oid.c, 58
- PKI\_OID\_get
  - pki\_oid.c, 58
- PKI\_OID\_get\_descr
  - pki\_oid.c, 58
- PKI\_OID\_get\_id
  - pki\_oid.c, 58
- PKI\_OID\_get\_str
  - pki\_oid.c, 58
- PKI\_OID\_load
  - pki\_oid.c, 58
- PKI\_OID\_new
  - pki\_oid.c, 58
- PKI\_OID\_new\_id
  - pki\_oid.c, 58
- PKI\_OID\_new\_text
  - pki\_oid.c, 59
- PKI\_PRQP\_CERTID\_new
  - prqp\_lib.c, 125
- PKI\_PRQP\_CERTID\_new\_cert
  - prqp\_lib.c, 125
- PKI\_PRQP\_REQ\_new\_cert
  - prqp\_lib.c, 125
- PKI\_RWLOCK\_destroy
  - pki\_threads\_vars.c, 114
- PKI\_RWLOCK\_free
  - pki\_threads\_vars.c, 114
- PKI\_RWLOCK\_init
  - pki\_threads\_vars.c, 114
- PKI\_RWLOCK\_new
  - pki\_threads\_vars.c, 114
- PKI\_RWLOCK\_read\_lock
  - pki\_threads\_vars.c, 114
- PKI\_RWLOCK\_release
  - pki\_threads\_vars.c, 114
- PKI\_RWLOCK\_try\_read\_lock
  - pki\_threads\_vars.c, 115
- PKI\_RWLOCK\_try\_write\_lock
  - pki\_threads\_vars.c, 115
- PKI\_RWLOCK\_write\_lock
  - pki\_threads\_vars.c, 115
- PKI\_set\_env
  - support.c, 143
- pki\_socket.c
  - PKI\_SOCKET\_close, 29
  - PKI\_SOCKET\_connect, 29
  - PKI\_SOCKET\_connect\_ssl, 29
  - PKI\_SOCKET\_free, 29
  - PKI\_SOCKET\_get\_fd, 29



- PKI\_SOCKET\_get\_ssl, [29](#)
- PKI\_SOCKET\_get\_url, [29](#)
- PKI\_SOCKET\_new, [29](#)
- PKI\_SOCKET\_new\_ssl, [29](#)
- PKI\_SOCKET\_open, [30](#)
- PKI\_SOCKET\_open\_url, [30](#)
- PKI\_SOCKET\_read, [30](#)
- PKI\_SOCKET\_set\_fd, [30](#)
- PKI\_SOCKET\_set\_ssl, [30](#)
- PKI\_SOCKET\_start\_ssl, [30](#)
- PKI\_SOCKET\_write, [30](#)
- PKI\_SOCKET\_close
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_connect
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_connect\_ssl
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_free
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_get\_fd
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_get\_ssl
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_get\_url
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_new
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_new\_ssl
  - pki\_socket.c, [29](#)
- PKI\_SOCKET\_open
  - pki\_socket.c, [30](#)
- PKI\_SOCKET\_open\_url
  - pki\_socket.c, [30](#)
- PKI\_SOCKET\_read
  - pki\_socket.c, [30](#)
- PKI\_SOCKET\_set\_fd
  - pki\_socket.c, [30](#)
- PKI\_SOCKET\_set\_ssl
  - pki\_socket.c, [30](#)
- PKI\_SOCKET\_start\_ssl
  - pki\_socket.c, [30](#)
- PKI\_SOCKET\_write
  - pki\_socket.c, [30](#)
- PKI\_SSL\_check\_verify
  - ssl.c, [34](#)
- PKI\_SSL\_close
  - ssl.c, [34](#)
- PKI\_SSL\_connect
  - ssl.c, [34](#)
- PKI\_SSL\_connect\_url
  - ssl.c, [34](#)
- PKI\_SSL\_dup
  - ssl.c, [35](#)
- PKI\_SSL\_free
  - ssl.c, [35](#)
- PKI\_SSL\_get\_fd
  - ssl.c, [35](#)
- PKI\_SSL\_get\_peer\_cert
  - ssl.c, [35](#)
- PKI\_SSL\_get\_peer\_chain
  - ssl.c, [35](#)
- PKI\_SSL\_get\_servername
  - ssl.c, [35](#)
- PKI\_SSL\_new
  - ssl.c, [35](#)
- PKI\_SSL\_read
  - ssl.c, [35](#)
- PKI\_SSL\_set\_algor
  - ssl.c, [35](#)
- PKI\_SSL\_set\_cipher
  - ssl.c, [35](#)
- PKI\_SSL\_set\_fd
  - ssl.c, [35](#)
- PKI\_SSL\_set\_flags
  - ssl.c, [35](#)
- PKI\_SSL\_set\_token
  - ssl.c, [35](#)
- PKI\_SSL\_set\_trusted
  - ssl.c, [36](#)
- PKI\_SSL\_set\_verify
  - ssl.c, [36](#)
- PKI\_SSL\_start\_ssl
  - ssl.c, [36](#)
- PKI\_SSL\_write
  - ssl.c, [36](#)
- PKI\_STACK\_del\_num
  - stack.c, [141](#)
- PKI\_STACK\_elements
  - stack.c, [141](#)
- PKI\_STACK\_free
  - stack.c, [141](#)
- PKI\_STACK\_free\_all
  - stack.c, [141](#)
- PKI\_STACK\_get\_num
  - stack.c, [141](#)
- PKI\_STACK\_ins\_num
  - stack.c, [142](#)
- PKI\_STACK\_new
  - stack.c, [142](#)
- PKI\_STACK\_new\_null
  - stack.c, [142](#)
- PKI\_STACK\_new\_type
  - stack.c, [142](#)
- PKI\_STACK\_pop
  - stack.c, [142](#)
- PKI\_STACK\_pop\_free
  - stack.c, [142](#)
- PKI\_STACK\_push

- stack.c, [142](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_add
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_delete
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_delete\_by\_  
name
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_delete\_by\_num
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_free
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_free\_all
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_get
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_get\_by\_name
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_get\_by\_num
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_num
  - pki\_x509\_attribute.c, [63](#)
- PKI\_STACK\_X509\_ATTRIBUTE\_replace
  - pki\_x509\_attribute.c, [63](#)
- pki\_string.c
  - PKI\_STRING\_dup, [60](#)
  - PKI\_STRING\_free, [60](#)
  - PKI\_STRING\_get\_digest, [60](#)
  - PKI\_STRING\_get\_parsed, [60](#)
  - PKI\_STRING\_get\_type, [60](#)
  - PKI\_STRING\_get\_utf8, [60](#)
  - PKI\_STRING\_new, [60](#)
  - PKI\_STRING\_new\_null, [60](#)
  - PKI\_STRING\_print, [60](#)
  - PKI\_STRING\_print\_fp, [60](#)
  - PKI\_STRING\_set, [60](#)
- PKI\_STRING\_dup
  - pki\_string.c, [60](#)
- PKI\_STRING\_free
  - pki\_string.c, [60](#)
- PKI\_STRING\_get\_digest
  - pki\_string.c, [60](#)
- PKI\_STRING\_get\_parsed
  - pki\_string.c, [60](#)
- PKI\_STRING\_get\_type
  - pki\_string.c, [60](#)
- PKI\_STRING\_get\_utf8
  - pki\_string.c, [60](#)
- PKI\_STRING\_new
  - pki\_string.c, [60](#)
- PKI\_STRING\_new\_null
  - pki\_string.c, [60](#)
- PKI\_STRING\_print
  - pki\_string.c, [60](#)
- PKI\_STRING\_print\_fp
  - pki\_string.c, [60](#)
- PKI\_STRING\_set\_fp
  - pki\_string.c, [60](#)
- PKI\_STRING\_set
  - pki\_string.c, [60](#)
- PKI\_THREAD\_create
  - pki\_threads.c, [111](#)
- PKI\_THREAD\_new
  - pki\_threads.c, [111](#)
- PKI\_THREAD\_self
  - pki\_threads.c, [111](#)
- pki\_threads.c
  - PKI\_THREAD\_create, [111](#)
  - PKI\_THREAD\_new, [111](#)
  - PKI\_THREAD\_self, [111](#)
- pki\_threads\_vars.c
  - PKI\_clock\_gettime, [113](#)
  - PKI\_COND\_broadcast, [113](#)
  - PKI\_COND\_destroy, [113](#)
  - PKI\_COND\_free, [113](#)
  - PKI\_COND\_init, [113](#)
  - PKI\_COND\_new, [113](#)
  - PKI\_COND\_signal, [113](#)
  - PKI\_COND\_timedwait, [113](#)
  - PKI\_COND\_wait, [114](#)
  - PKI\_MUTEX\_acquire, [114](#)
  - PKI\_MUTEX\_destroy, [114](#)
  - PKI\_MUTEX\_free, [114](#)
  - PKI\_MUTEX\_init, [114](#)
  - PKI\_MUTEX\_new, [114](#)
  - PKI\_MUTEX\_release, [114](#)
  - PKI\_RWLOCK\_destroy, [114](#)
  - PKI\_RWLOCK\_free, [114](#)
  - PKI\_RWLOCK\_init, [114](#)
  - PKI\_RWLOCK\_new, [114](#)
  - PKI\_RWLOCK\_read\_lock, [114](#)
  - PKI\_RWLOCK\_release, [114](#)
  - PKI\_RWLOCK\_try\_read\_lock, [115](#)
  - PKI\_RWLOCK\_try\_write\_lock, [115](#)
  - PKI\_RWLOCK\_write\_lock, [115](#)
- pki\_time.c
  - PKI\_TIME\_adj, [61](#)
  - PKI\_TIME\_dup, [61](#)
  - PKI\_TIME\_free, [61](#)
  - PKI\_TIME\_free\_void, [61](#)
  - PKI\_TIME\_get\_parsed, [61](#)
  - PKI\_TIME\_new, [61](#)
  - PKI\_TIME\_print, [62](#)
  - PKI\_TIME\_print\_fp, [62](#)
- PKI\_TIME\_adj
  - pki\_time.c, [61](#)
- PKI\_TIME\_dup
  - pki\_time.c, [61](#)
- PKI\_TIME\_free
  - pki\_time.c, [61](#)

- PKI\_TIME\_free\_void
  - pki\_time.c, [61](#)
- PKI\_TIME\_get\_parsed
  - pki\_time.c, [61](#)
- PKI\_TIME\_new
  - pki\_time.c, [61](#)
- PKI\_TIME\_print
  - pki\_time.c, [62](#)
- PKI\_TIME\_print\_fp
  - pki\_time.c, [62](#)
- PKI\_TOKEN\_add\_profile
  - token.c, [147](#)
- PKI\_TOKEN\_check
  - token.c, [147](#)
- PKI\_TOKEN\_cred\_cb\_env
  - token.c, [147](#)
- PKI\_TOKEN\_cred\_cb\_stdin
  - token.c, [147](#)
- PKI\_TOKEN\_cred\_get
  - token.c, [148](#)
- PKI\_TOKEN\_cred\_set\_cb
  - token.c, [148](#)
- PKI\_TOKEN\_del\_url
  - token.c, [148](#)
- PKI\_TOKEN\_export\_cert
  - token.c, [148](#)
- PKI\_TOKEN\_export\_keypair
  - token.c, [148](#)
- PKI\_TOKEN\_export\_keypair\_url
  - token.c, [148](#)
- PKI\_TOKEN\_export\_otherCerts
  - token.c, [148](#)
- PKI\_TOKEN\_export\_p12
  - token.c, [148](#)
- PKI\_TOKEN\_export\_req
  - token.c, [148](#)
- PKI\_TOKEN\_export\_trustedCerts
  - token.c, [148](#)
- PKI\_TOKEN\_free
  - token.c, [148](#)
- PKI\_TOKEN\_free\_void
  - token.c, [148](#)
- PKI\_TOKEN\_get\_algor
  - token.c, [149](#)
- PKI\_TOKEN\_get\_algor\_id
  - token.c, [149](#)
- PKI\_TOKEN\_get\_cacert
  - token.c, [149](#)
- PKI\_TOKEN\_get\_cacert\_data
  - token\_data.c, [156](#)
- PKI\_TOKEN\_get\_cert
  - token.c, [149](#)
- PKI\_TOKEN\_get\_cert\_data
  - token\_data.c, [156](#)
- PKI\_TOKEN\_get\_config\_dir
  - token.c, [149](#)
- PKI\_TOKEN\_get\_cred
  - token.c, [149](#)
- PKI\_TOKEN\_get\_crls
  - token.c, [149](#)
- PKI\_TOKEN\_get\_identity\_data
  - token\_data.c, [156](#)
- PKI\_TOKEN\_get\_keypair
  - token.c, [149](#)
- PKI\_TOKEN\_get\_keypair\_data
  - token\_data.c, [156](#)
- PKI\_TOKEN\_get\_name
  - token.c, [149](#)
- PKI\_TOKEN\_get\_otherCerts
  - token.c, [150](#)
- PKI\_TOKEN\_get\_otherCerts\_data
  - token\_data.c, [156](#)
- PKI\_TOKEN\_get\_p12
  - token.c, [150](#)
- PKI\_TOKEN\_get\_privkey\_data
  - token\_data.c, [156](#)
- PKI\_TOKEN\_get\_pubkey\_data
  - token\_data.c, [156](#)
- PKI\_TOKEN\_get\_trustedCerts
  - token.c, [150](#)
- PKI\_TOKEN\_get\_trustedCerts\_data
  - token\_data.c, [156](#)
- PKI\_TOKEN\_ID\_INFO\_get
  - token\_id.c, [157](#)
- PKI\_TOKEN\_ID\_INFO\_list
  - token\_id.c, [157](#)
- PKI\_TOKEN\_ID\_num
  - token\_id.c, [157](#)
- PKI\_TOKEN\_ID\_set
  - token\_id.c, [157](#)
- PKI\_TOKEN\_import\_cert
  - token.c, [150](#)
- PKI\_TOKEN\_import\_cert\_stack
  - token.c, [150](#)
- PKI\_TOKEN\_import\_keypair
  - token.c, [150](#)
- PKI\_TOKEN\_init
  - token.c, [150](#)
- PKI\_TOKEN\_issue\_cert
  - token.c, [150](#)
- PKI\_TOKEN\_issue\_crl
  - token.c, [150](#)
- PKI\_TOKEN\_issue\_proxy
  - token.c, [150](#)
- PKI\_TOKEN\_load\_cacert
  - token.c, [151](#)
- PKI\_TOKEN\_load\_cert
  - token.c, [151](#)

- PKI\_TOKEN\_load\_config
  - token.c, [151](#)
- PKI\_TOKEN\_load\_crls
  - token.c, [151](#)
- PKI\_TOKEN\_load\_keypair
  - token.c, [151](#)
- PKI\_TOKEN\_load\_otherCerts
  - token.c, [151](#)
- PKI\_TOKEN\_load\_profiles
  - token.c, [151](#)
- PKI\_TOKEN\_load\_req
  - token.c, [152](#)
- PKI\_TOKEN\_load\_trustedCerts
  - token.c, [152](#)
- PKI\_TOKEN\_login
  - token.c, [152](#)
- PKI\_TOKEN\_new
  - token.c, [152](#)
- PKI\_TOKEN\_new\_keypair
  - token.c, [152](#)
- PKI\_TOKEN\_new\_keypair\_ex
  - token.c, [152](#)
- PKI\_TOKEN\_new\_keypair\_url
  - token.c, [152](#)
- PKI\_TOKEN\_new\_keypair\_url\_ex
  - token.c, [153](#)
- PKI\_TOKEN\_new\_null
  - token.c, [153](#)
- PKI\_TOKEN\_new\_p12
  - token.c, [153](#)
- PKI\_TOKEN\_new\_req
  - token.c, [153](#)
- PKI\_TOKEN\_OID\_new
  - token.c, [153](#)
- PKI\_TOKEN\_print\_info
  - token.c, [153](#)
- PKI\_TOKEN\_search\_profile
  - token.c, [153](#)
- PKI\_TOKEN\_self\_sign
  - token.c, [153](#)
- PKI\_TOKEN\_set\_algor
  - token.c, [153](#)
- PKI\_TOKEN\_set\_algor\_by\_name
  - token.c, [154](#)
- PKI\_TOKEN\_set\_cacert
  - token.c, [154](#)
- PKI\_TOKEN\_set\_cert
  - token.c, [154](#)
- PKI\_TOKEN\_set\_config\_dir
  - token.c, [154](#)
- PKI\_TOKEN\_set\_cred
  - token.c, [154](#)
- PKI\_TOKEN\_set\_crls
  - token.c, [154](#)
- PKI\_TOKEN\_set\_keypair
  - token.c, [155](#)
- PKI\_TOKEN\_set\_otherCerts
  - token.c, [155](#)
- PKI\_TOKEN\_set\_req
  - token.c, [155](#)
- PKI\_TOKEN\_set\_trustedCerts
  - token.c, [155](#)
- PKI\_TOKEN\_use\_slot
  - token.c, [155](#)
- pki\_x509.c
  - PKI\_X509\_CALLBACKS\_get, [116](#)
  - PKI\_X509\_delete, [116](#)
  - PKI\_X509\_dup, [116](#)
  - PKI\_X509\_dup\_value, [116](#)
  - PKI\_X509\_free, [116](#)
  - PKI\_X509\_free\_void, [117](#)
  - PKI\_X509\_get\_data, [117](#)
  - PKI\_X509\_get\_hsm, [117](#)
  - PKI\_X509\_get\_parsed, [117](#)
  - PKI\_X509\_get\_reference, [117](#)
  - PKI\_X509\_get\_type, [117](#)
  - PKI\_X509\_get\_value, [117](#)
  - PKI\_X509\_is\_signed, [117](#)
  - PKI\_X509\_new, [117](#)
  - PKI\_X509\_new\_dup\_value, [117](#)
  - PKI\_X509\_new\_value, [117](#)
  - PKI\_X509\_print\_parsed, [117](#)
  - PKI\_X509\_set\_hsm, [117](#)
  - PKI\_X509\_set\_reference, [118](#)
  - PKI\_X509\_set\_value, [118](#)
- pki\_x509\_attribute.c
  - PKI\_STACK\_X509\_ATTRIBUTE\_add, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_delete, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_delete\_-  
by\_name, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_delete\_-  
by\_num, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_free, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_free\_all,  
[63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_get, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_get\_by\_-  
name, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_get\_by\_-  
num, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_num, [63](#)
  - PKI\_STACK\_X509\_ATTRIBUTE\_replace,  
[63](#)
  - PKI\_X509\_ATTRIBUTE\_free, [63](#)
  - PKI\_X509\_ATTRIBUTE\_free\_null, [63](#)
  - PKI\_X509\_ATTRIBUTE\_get\_descr, [64](#)
  - PKI\_X509\_ATTRIBUTE\_get\_parsed, [64](#)
  - PKI\_X509\_ATTRIBUTE\_get\_value, [64](#)

- PKI\_X509\_ATTRIBUTE\_new, [64](#)
- PKI\_X509\_ATTRIBUTE\_new\_name, [64](#)
- PKI\_X509\_ATTRIBUTE\_new\_null, [64](#)
- PKI\_X509\_ATTRIBUTE\_free
  - pki\_x509\_attribute.c, [63](#)
- PKI\_X509\_ATTRIBUTE\_free\_null
  - pki\_x509\_attribute.c, [63](#)
- PKI\_X509\_ATTRIBUTE\_get\_descr
  - pki\_x509\_attribute.c, [64](#)
- PKI\_X509\_ATTRIBUTE\_get\_parsed
  - pki\_x509\_attribute.c, [64](#)
- PKI\_X509\_ATTRIBUTE\_get\_value
  - pki\_x509\_attribute.c, [64](#)
- PKI\_X509\_ATTRIBUTE\_new
  - pki\_x509\_attribute.c, [64](#)
- PKI\_X509\_ATTRIBUTE\_new\_name
  - pki\_x509\_attribute.c, [64](#)
- PKI\_X509\_ATTRIBUTE\_new\_null
  - pki\_x509\_attribute.c, [64](#)
- PKI\_X509\_CALLBACKS\_get
  - pki\_x509.c, [116](#)
- pki\_x509\_cert.c
  - NID\_proxyCertInfo, [68](#)
  - PKI\_X509\_CERT\_add\_extension, [66](#)
  - PKI\_X509\_CERT\_add\_extension\_stack, [66](#)
  - PKI\_X509\_CERT\_check\_domain, [66](#)
  - PKI\_X509\_CERT\_dup, [66](#)
  - PKI\_X509\_CERT\_fingerprint, [66](#)
  - PKI\_X509\_CERT\_fingerprint\_by\_name, [66](#)
  - PKI\_X509\_CERT\_free, [66](#)
  - PKI\_X509\_CERT\_free\_void, [67](#)
  - PKI\_X509\_CERT\_get\_cdp, [67](#)
  - PKI\_X509\_CERT\_get\_data, [67](#)
  - PKI\_X509\_CERT\_get\_email, [67](#)
  - PKI\_X509\_CERT\_get\_extension\_by\_id, [67](#)
  - PKI\_X509\_CERT\_get\_extension\_by\_name, [67](#)
  - PKI\_X509\_CERT\_get\_extension\_by\_oid, [67](#)
  - PKI\_X509\_CERT\_get\_extensions, [67](#)
  - PKI\_X509\_CERT\_get\_keysize, [67](#)
  - PKI\_X509\_CERT\_get\_parsed, [67](#)
  - PKI\_X509\_CERT\_get\_type, [67](#)
  - PKI\_X509\_CERT\_is\_ca, [67](#)
  - PKI\_X509\_CERT\_is\_proxy, [67](#)
  - PKI\_X509\_CERT\_is\_selfsigned, [67](#)
  - PKI\_X509\_CERT\_key\_hash, [68](#)
  - PKI\_X509\_CERT\_key\_hash\_by\_name, [68](#)
  - PKI\_X509\_CERT\_new, [68](#)
  - PKI\_X509\_CERT\_new\_null, [68](#)
  - PKI\_X509\_CERT\_print\_parsed, [68](#)
  - PKI\_X509\_CERT\_set\_data, [68](#)
  - PKI\_X509\_CERT\_sign, [68](#)
  - PKI\_X509\_CERT\_sign\_tk, [68](#)
- PKI\_X509\_CERT\_add\_extension
  - pki\_x509\_cert.c, [66](#)
- PKI\_X509\_CERT\_add\_extension\_stack
  - pki\_x509\_cert.c, [66](#)
- PKI\_X509\_CERT\_check\_domain
  - pki\_x509\_cert.c, [66](#)
- PKI\_X509\_CERT\_dup
  - pki\_x509\_cert.c, [66](#)
- PKI\_X509\_CERT\_fingerprint
  - pki\_x509\_cert.c, [66](#)
- PKI\_X509\_CERT\_fingerprint\_by\_name
  - pki\_x509\_cert.c, [66](#)
- PKI\_X509\_CERT\_free
  - pki\_x509\_cert.c, [66](#)
- PKI\_X509\_CERT\_free\_void
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get
  - pki\_x509\_cert\_io.c, [11](#)
- PKI\_X509\_CERT\_get\_cdp
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_data
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_email
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_extension\_by\_id
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_extension\_by\_name
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_extension\_by\_oid
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_extensions
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_keysize
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_mem
  - pki\_x509\_cert\_io.c, [11](#)
- PKI\_X509\_CERT\_get\_parsed
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_get\_type
  - pki\_x509\_cert.c, [67](#)
- 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, [67](#)
- PKI\_X509\_CERT\_is\_proxy
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_is\_selfsigned
  - pki\_x509\_cert.c, [67](#)
- PKI\_X509\_CERT\_key\_hash
  - pki\_x509\_cert.c, [68](#)
- PKI\_X509\_CERT\_key\_hash\_by\_name
  - pki\_x509\_cert.c, [68](#)
- PKI\_X509\_CERT\_new
  - pki\_x509\_cert.c, [68](#)
- PKI\_X509\_CERT\_new\_null
  - pki\_x509\_cert.c, [68](#)
- PKI\_X509\_CERT\_print\_parsed
  - pki\_x509\_cert.c, [68](#)
- 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\_set\_data
  - pki\_x509\_cert.c, [68](#)
- PKI\_X509\_CERT\_sign
  - pki\_x509\_cert.c, [68](#)
- PKI\_X509\_CERT\_sign\_tk
  - pki\_x509\_cert.c, [68](#)
- 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, [70](#)
  - PKI\_X509\_CRL\_add\_extension\_stack, [70](#)
  - PKI\_X509\_CRL\_ENTRY\_free, [70](#)
  - PKI\_X509\_CRL\_ENTRY\_free\_void, [70](#)
  - PKI\_X509\_CRL\_ENTRY\_new, [70](#)
  - PKI\_X509\_CRL\_ENTRY\_new\_serial, [70](#)
  - PKI\_X509\_CRL\_free, [70](#)
  - PKI\_X509\_CRL\_free\_void, [70](#)
  - PKI\_X509\_CRL\_get\_data, [70](#)
  - PKI\_X509\_CRL\_get\_parsed, [70](#)
  - PKI\_X509\_CRL\_lookup, [71](#)
  - PKI\_X509\_CRL\_lookup\_cert, [71](#)
  - PKI\_X509\_CRL\_lookup\_long, [71](#)
  - PKI\_X509\_CRL\_lookup\_serial, [71](#)
  - PKI\_X509\_CRL\_new, [71](#)
  - PKI\_X509\_CRL\_print\_parsed, [71](#)
- PKI\_X509\_CRL\_lookup
  - pki\_x509\_crl.c, [71](#)
- PKI\_X509\_CRL\_lookup\_cert
  - pki\_x509\_crl.c, [71](#)
- PKI\_X509\_CRL\_lookup\_long
  - pki\_x509\_crl.c, [71](#)
- PKI\_X509\_CRL\_lookup\_serial
  - pki\_x509\_crl.c, [71](#)
- PKI\_X509\_CRL\_new
  - pki\_x509\_crl.c, [71](#)
- PKI\_X509\_CRL\_print\_parsed
  - pki\_x509\_crl.c, [71](#)

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, 116  
PKI\_X509\_dup  
    pki\_x509.c, 116  
PKI\_X509\_dup\_value  
    pki\_x509.c, 116  
pki\_x509\_extension.c  
    PKI\_X509\_EXTENSION\_free, 72  
    PKI\_X509\_EXTENSION\_free\_void, 72  
    PKI\_X509\_EXTENSION\_get\_list, 72  
    PKI\_X509\_EXTENSION\_new, 72  
    PKI\_X509\_EXTENSION\_value\_new\_profile,  
        72  
PKI\_X509\_EXTENSION\_free  
    pki\_x509\_extension.c, 72  
PKI\_X509\_EXTENSION\_free\_void  
    pki\_x509\_extension.c, 72  
PKI\_X509\_EXTENSION\_get\_list  
    pki\_x509\_extension.c, 72  
PKI\_X509\_EXTENSION\_new  
    pki\_x509\_extension.c, 72  
PKI\_X509\_EXTENSION\_value\_new\_profile  
    pki\_x509\_extension.c, 72  
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, 116  
PKI\_X509\_free\_void  
    pki\_x509.c, 117  
PKI\_X509\_get  
    pki\_x509\_io.c, 15  
PKI\_X509\_get\_data  
    pki\_x509.c, 117  
PKI\_X509\_get\_hsm  
    pki\_x509.c, 117  
PKI\_X509\_get\_mem  
    pki\_x509\_mem.c, 118  
PKI\_X509\_get\_mem\_value  
    pki\_x509\_mem.c, 118  
PKI\_X509\_get\_mimetype  
    pki\_x509\_mime.c, 119  
PKI\_X509\_get\_parsed  
    pki\_x509.c, 117  
PKI\_X509\_get\_reference  
    pki\_x509.c, 117  
PKI\_X509\_get\_type  
    pki\_x509.c, 117  
PKI\_X509\_get\_url  
    pki\_x509\_io.c, 15  
PKI\_X509\_get\_value  
    pki\_x509.c, 117  
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, 117  
PKI\_X509\_KEYPAIR\_free  
    pki\_keypair.c, 48  
PKI\_X509\_KEYPAIR\_free\_void  
    pki\_keypair.c, 48  
PKI\_X509\_KEYPAIR\_get  
    pki\_keypair\_io.c, 5  
PKI\_X509\_KEYPAIR\_get\_algor  
    pki\_keypair.c, 48  
PKI\_X509\_KEYPAIR\_get\_mem  
    pki\_keypair\_io.c, 5  
PKI\_X509\_KEYPAIR\_get\_p8  
    pki\_keypair.c, 48  
PKI\_X509\_KEYPAIR\_get\_parsed  
    pki\_keypair.c, 49  
PKI\_X509\_KEYPAIR\_get\_scheme  
    pki\_keypair.c, 49  
PKI\_X509\_KEYPAIR\_get\_size  
    pki\_keypair.c, 49  
PKI\_X509\_KEYPAIR\_get\_url  
    pki\_keypair\_io.c, 5  
PKI\_X509\_KEYPAIR\_new  
    pki\_keypair.c, 49



- PKI\_X509\_KEYPAIR\_new\_kp
  - pki\_keypair.c, [49](#)
- PKI\_X509\_KEYPAIR\_new\_null
  - pki\_keypair.c, [49](#)
- PKI\_X509\_KEYPAIR\_new\_p8
  - pki\_keypair.c, [49](#)
- PKI\_X509\_KEYPAIR\_new\_url
  - pki\_keypair.c, [49](#)
- PKI\_X509\_KEYPAIR\_new\_url\_kp
  - pki\_keypair.c, [49](#)
- PKI\_X509\_KEYPAIR\_pub\_digest
  - pki\_keypair.c, [49](#)
- 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\_get\_algor
  - pki\_keypair.c, [49](#)
- PKI\_X509\_KEYPAIR\_VALUE\_get\_scheme
  - pki\_keypair.c, [49](#)
- PKI\_X509\_KEYPAIR\_VALUE\_get\_size
  - pki\_keypair.c, [50](#)
- PKI\_X509\_KEYPAIR\_VALUE\_pub\_digest
  - pki\_keypair.c, [50](#)
- pki\_x509\_mem.c
  - PKI\_X509\_get\_mem, [118](#)
  - PKI\_X509\_get\_mem\_value, [118](#)
  - PKI\_X509\_put\_mem, [119](#)
  - PKI\_X509\_put\_mem\_value, [119](#)
  - PKI\_X509\_STACK\_get\_mem, [119](#)
  - PKI\_X509\_STACK\_put\_mem, [119](#)
- pki\_x509\_mime.c
  - PKI\_X509\_get\_mimetype, [119](#)
- pki\_x509\_name.c
  - PKI\_X509\_NAME\_add, [73](#)
  - PKI\_X509\_NAME\_cmp, [73](#)
  - PKI\_X509\_NAME\_dup, [73](#)
  - PKI\_X509\_NAME\_free, [73](#)
  - PKI\_X509\_NAME\_get\_digest, [73](#)
  - PKI\_X509\_NAME\_get\_list, [73](#)
  - PKI\_X509\_NAME\_get\_parsed, [74](#)
  - PKI\_X509\_NAME\_list\_free, [74](#)
  - PKI\_X509\_NAME\_new, [74](#)
  - PKI\_X509\_NAME\_new\_null, [74](#)
  - PKI\_X509\_NAME\_RDN\_type\_descr, [74](#)
  - PKI\_X509\_NAME\_RDN\_type\_id, [74](#)
  - PKI\_X509\_NAME\_RDN\_type\_text, [74](#)
  - PKI\_X509\_NAME\_RDN\_value, [74](#)
- PKI\_X509\_NAME\_add
  - pki\_x509\_name.c, [73](#)
- PKI\_X509\_NAME\_cmp
  - pki\_x509\_name.c, [73](#)
- PKI\_X509\_NAME\_dup
  - pki\_x509\_name.c, [73](#)
- PKI\_X509\_NAME\_free
  - pki\_x509\_name.c, [73](#)
- PKI\_X509\_NAME\_get\_digest
  - pki\_x509\_name.c, [73](#)
- PKI\_X509\_NAME\_get\_list
  - pki\_x509\_name.c, [73](#)
- PKI\_X509\_NAME\_get\_parsed
  - pki\_x509\_name.c, [74](#)
- PKI\_X509\_NAME\_list\_free
  - pki\_x509\_name.c, [74](#)
- PKI\_X509\_NAME\_new
  - pki\_x509\_name.c, [74](#)
- PKI\_X509\_NAME\_new\_null
  - pki\_x509\_name.c, [74](#)
- PKI\_X509\_NAME\_RDN\_type\_descr
  - pki\_x509\_name.c, [74](#)
- PKI\_X509\_NAME\_RDN\_type\_id
  - pki\_x509\_name.c, [74](#)
- PKI\_X509\_NAME\_RDN\_type\_text
  - pki\_x509\_name.c, [74](#)
- PKI\_X509\_NAME\_RDN\_value
  - pki\_x509\_name.c, [74](#)
- PKI\_X509\_new
  - pki\_x509.c, [117](#)
- PKI\_X509\_new\_dup\_value
  - pki\_x509.c, [117](#)
- PKI\_X509\_new\_value
  - pki\_x509.c, [117](#)
- PKI\_X509\_OCSP\_REQ\_add\_cert
  - pki\_ocsp\_req.c, [52](#)
- PKI\_X509\_OCSP\_REQ\_add\_longlong
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_add\_nonce
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_add\_serial
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_add\_txt
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_DATA\_sign
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_elements
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_free
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_free\_void
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_get
  - pki\_ocsp\_req\_io.c, [7](#)



- PKI\_X509\_OCSP\_REQ\_get\_cid
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_get\_data
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_get\_mem
  - pki\_ocsp\_req\_io.c, [7](#)
- PKI\_X509\_OCSP\_REQ\_get\_parsed
  - pki\_ocsp\_req.c, [53](#)
- PKI\_X509\_OCSP\_REQ\_get\_serial
  - pki\_ocsp\_req.c, [54](#)
- PKI\_X509\_OCSP\_REQ\_get\_url
  - pki\_ocsp\_req\_io.c, [7](#)
- PKI\_X509\_OCSP\_REQ\_new
  - pki\_ocsp\_req.c, [54](#)
- PKI\_X509\_OCSP\_REQ\_new\_null
  - pki\_ocsp\_req.c, [54](#)
- PKI\_X509\_OCSP\_REQ\_print\_parsed
  - pki\_ocsp\_req.c, [54](#)
- 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, [54](#)
- PKI\_X509\_OCSP\_REQ\_sign\_tk
  - pki\_ocsp\_req.c, [54](#)
- 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, [56](#)
- PKI\_X509\_OCSP\_RESP\_copy\_nonce
  - pki\_ocsp\_resp.c, [56](#)
- PKI\_X509\_OCSP\_RESP\_DATA\_sign
  - pki\_ocsp\_resp.c, [56](#)
- PKI\_X509\_OCSP\_RESP\_free
  - pki\_ocsp\_resp.c, [56](#)
- PKI\_X509\_OCSP\_RESP\_free\_void
  - pki\_ocsp\_resp.c, [56](#)
- PKI\_X509\_OCSP\_RESP\_get
  - pki\_ocsp\_resp\_io.c, [9](#)
- PKI\_X509\_OCSP\_RESP\_get\_data
  - pki\_ocsp\_resp.c, [56](#)
- PKI\_X509\_OCSP\_RESP\_get\_mem
  - pki\_ocsp\_resp\_io.c, [9](#)
- PKI\_X509\_OCSP\_RESP\_get\_parsed
  - pki\_ocsp\_resp.c, [56](#)
- PKI\_X509\_OCSP\_RESP\_get\_url
  - pki\_ocsp\_resp\_io.c, [9](#)
- PKI\_X509\_OCSP\_RESP\_new
  - pki\_ocsp\_resp.c, [56](#)
- PKI\_X509\_OCSP\_RESP\_new\_null
  - pki\_ocsp\_resp.c, [56](#)
- PKI\_X509\_OCSP\_RESP\_print\_parsed
  - pki\_ocsp\_resp.c, [56](#)
- 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, [56](#)
- PKI\_X509\_OCSP\_RESP\_sign
  - pki\_ocsp\_resp.c, [57](#)
- PKI\_X509\_OCSP\_RESP\_sign\_tk
  - pki\_ocsp\_resp.c, [57](#)
- 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, [76](#)
  - BAG\_DATATYPE\_CACERT, [76](#)
  - BAG\_DATATYPE\_CERT, [76](#)
  - BAG\_DATATYPE\_KEYPAIR, [76](#)
  - BAG\_DATATYPE\_OTHERCERTS, [76](#)
  - BAG\_DATATYPE\_UNKNOWN, [76](#)
- pki\_x509\_p12.c
  - bag\_datatype\_st, [76](#)
  - PEM\_read\_bio\_PKCS12, [76](#)
  - PEM\_write\_bio\_PKCS12, [76](#)
  - PKI\_X509\_PKCS12\_DATA\_add\_certs, [76](#)
  - PKI\_X509\_PKCS12\_DATA\_add\_keypair, [76](#)
  - PKI\_X509\_PKCS12\_DATA\_add\_other\_certs, [76](#)
  - PKI\_X509\_PKCS12\_DATA\_free, [76](#)
  - PKI\_X509\_PKCS12\_DATA\_new, [76](#)
  - PKI\_X509\_PKCS12\_free, [76](#)

- PKI\_X509\_PKCS12\_free\_void, [76](#)
- PKI\_X509\_PKCS12\_get\_cacert, [76](#)
- PKI\_X509\_PKCS12\_get\_cert, [77](#)
- PKI\_X509\_PKCS12\_get\_keypair, [77](#)
- PKI\_X509\_PKCS12\_get\_otherCerts, [77](#)
- PKI\_X509\_PKCS12\_new, [77](#)
- PKI\_X509\_PKCS12\_new\_null, [77](#)
- PKI\_X509\_PKCS12\_TOKEN\_export, [77](#)
- PKI\_X509\_PKCS12\_verify\_cred, [77](#)
- 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, [76](#)
- PKI\_X509\_PKCS12\_DATA\_add\_keypair
  - pki\_x509\_p12.c, [76](#)
- PKI\_X509\_PKCS12\_DATA\_add\_other\_certs
  - pki\_x509\_p12.c, [76](#)
- PKI\_X509\_PKCS12\_DATA\_free
  - pki\_x509\_p12.c, [76](#)
- PKI\_X509\_PKCS12\_DATA\_new
  - pki\_x509\_p12.c, [76](#)
- PKI\_X509\_PKCS12\_free
  - pki\_x509\_p12.c, [76](#)
- PKI\_X509\_PKCS12\_free\_void
  - pki\_x509\_p12.c, [76](#)
- PKI\_X509\_PKCS12\_get
  - pki\_x509\_p12\_io.c, [17](#)
- PKI\_X509\_PKCS12\_get\_cacert
  - pki\_x509\_p12.c, [76](#)
- PKI\_X509\_PKCS12\_get\_cert
  - pki\_x509\_p12.c, [77](#)
- PKI\_X509\_PKCS12\_get\_keypair
  - pki\_x509\_p12.c, [77](#)
- PKI\_X509\_PKCS12\_get\_mem
  - pki\_x509\_p12\_io.c, [17](#)
- PKI\_X509\_PKCS12\_get\_otherCerts
  - pki\_x509\_p12.c, [77](#)
- PKI\_X509\_PKCS12\_get\_url
  - pki\_x509\_p12\_io.c, [17](#)
- PKI\_X509\_PKCS12\_new
  - pki\_x509\_p12.c, [77](#)
- PKI\_X509\_PKCS12\_new\_null
  - pki\_x509\_p12.c, [77](#)
- 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, [77](#)
- PKI\_X509\_PKCS12\_verify\_cred
  - pki\_x509\_p12.c, [77](#)
- pki\_x509\_pkcs7.c
  - PKI\_X509\_PKCS7\_add\_attribute, [80](#)
  - PKI\_X509\_PKCS7\_add\_cert, [80](#)
  - PKI\_X509\_PKCS7\_add\_cert\_stack, [80](#)
  - PKI\_X509\_PKCS7\_add\_crl, [80](#)
  - PKI\_X509\_PKCS7\_add\_crl\_stack, [80](#)
  - PKI\_X509\_PKCS7\_add\_recipient, [80](#)
  - PKI\_X509\_PKCS7\_add\_signed\_attribute, [80](#)
  - PKI\_X509\_PKCS7\_add\_signer, [80](#)
  - PKI\_X509\_PKCS7\_add\_signer\_tk, [80](#)
  - PKI\_X509\_PKCS7\_clear\_certs, [80](#)
  - PKI\_X509\_PKCS7\_decode, [80](#)
  - PKI\_X509\_PKCS7\_delete\_attribute, [80](#)
  - PKI\_X509\_PKCS7\_delete\_signed\_attribute, [80](#)
  - PKI\_X509\_PKCS7\_encode, [81](#)
  - PKI\_X509\_PKCS7\_free, [81](#)
  - PKI\_X509\_PKCS7\_free\_void, [81](#)
  - PKI\_X509\_PKCS7\_get\_attribute, [81](#)
  - PKI\_X509\_PKCS7\_get\_attribute\_by\_name, [81](#)
  - PKI\_X509\_PKCS7\_get\_cert, [81](#)
  - PKI\_X509\_PKCS7\_get\_certs\_num, [81](#)
  - PKI\_X509\_PKCS7\_get\_crl, [81](#)
  - PKI\_X509\_PKCS7\_get\_crls\_num, [81](#)
  - PKI\_X509\_PKCS7\_get\_data, [81](#)
  - PKI\_X509\_PKCS7\_get\_data\_tk, [81](#)
  - PKI\_X509\_PKCS7\_get\_encode\_alg, [81](#)
  - PKI\_X509\_PKCS7\_get\_raw\_data, [81](#)
  - PKI\_X509\_PKCS7\_get\_recipient\_cert, [82](#)
  - PKI\_X509\_PKCS7\_get\_recipient\_info, [82](#)
  - PKI\_X509\_PKCS7\_get\_recipients\_num, [82](#)

- PKI\_X509\_PKCS7\_get\_signed\_attribute, 82
- PKI\_X509\_PKCS7\_get\_signed\_attribute\_  
by\_name, 82
- PKI\_X509\_PKCS7\_get\_signer\_info, 82
- PKI\_X509\_PKCS7\_get\_signers\_num, 82
- PKI\_X509\_PKCS7\_get\_type, 82
- PKI\_X509\_PKCS7\_has\_recipients, 82
- PKI\_X509\_PKCS7\_has\_signers, 82
- PKI\_X509\_PKCS7\_new, 82
- PKI\_X509\_PKCS7\_set\_cipher, 82
- PKI\_X509\_PKCS7\_set\_recipients, 82
- PKI\_X509\_PKCS7\_VALUE\_print\_bio, 83
- PKI\_X509\_PKCS7\_add\_attribute
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_add\_cert
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_add\_cert\_stack
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_add\_crl
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_add\_crl\_stack
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_add\_recipient
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_add\_signed\_attribute
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_add\_signer
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_add\_signer\_tk
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_clear\_certs
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_decode
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_delete\_attribute
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_delete\_signed\_attribute
  - pki\_x509\_pkcs7.c, 80
- PKI\_X509\_PKCS7\_encode
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_free
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_free\_void
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get
  - pki\_x509\_pkcs7\_io.c, 19
- PKI\_X509\_PKCS7\_get\_attribute
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_attribute\_by\_name
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_cert
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_certs\_num
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_crl
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_crls\_num
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_data
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_data\_tk
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_encode\_alg
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_mem
  - pki\_x509\_pkcs7\_io.c, 19
- PKI\_X509\_PKCS7\_get\_raw\_data
  - pki\_x509\_pkcs7.c, 81
- PKI\_X509\_PKCS7\_get\_recipient\_cert
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_get\_recipient\_info
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_get\_recipients\_num
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_get\_signed\_attribute
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_get\_signed\_attribute\_by\_name
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_get\_signer\_info
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_get\_signers\_num
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_get\_type
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_get\_url
  - pki\_x509\_pkcs7\_io.c, 19
- PKI\_X509\_PKCS7\_has\_recipients
  - pki\_x509\_pkcs7.c, 82
- PKI\_X509\_PKCS7\_has\_signers
  - pki\_x509\_pkcs7.c, 82
- 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, 82
- 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, [82](#)
- PKI\_X509\_PKCS7\_set\_recipients
  - pki\_x509\_pkcs7.c, [82](#)
- 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, [83](#)
- PKI\_X509\_print\_parsed
  - pki\_x509.c, [117](#)
- PKI\_X509\_PROFILE\_add\_extension
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_free
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_free\_void
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_get\_default
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_get\_ext\_by\_num
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_get\_extensions
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_get\_exts\_num
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_get\_name
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_get\_value
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_load
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_new
  - profile.c, [120](#)
- PKI\_X509\_PROFILE\_put\_file
  - profile.c, [120](#)
- PKI\_X509\_PRQP\_NONCE\_new
  - prqp\_lib.c, [125](#)
- PKI\_X509\_PRQP\_REQ\_add\_service
  - prqp\_lib.c, [125](#)
- PKI\_X509\_PRQP\_REQ\_add\_service\_stack
  - prqp\_lib.c, [125](#)
- PKI\_X509\_PRQP\_REQ\_free
  - prqp\_lib.c, [125](#)
- PKI\_X509\_PRQP\_REQ\_free\_void
  - prqp\_lib.c, [125](#)
- PKI\_X509\_PRQP\_REQ\_get
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_get\_data
  - prqp\_lib.c, [125](#)
- PKI\_X509\_PRQP\_REQ\_get\_mem
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_get\_url
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_new\_certs\_res
  - prqp\_lib.c, [125](#)
- PKI\_X509\_PRQP\_REQ\_new\_null
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_REQ\_new\_url
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_REQ\_print
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_REQ\_print\_fp
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_REQ\_put
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_put\_mem
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_put\_url
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_STACK\_get
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_STACK\_get\_mem
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_STACK\_get\_url
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_STACK\_put
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_STACK\_put\_mem
  - prqp\_req\_io.c, [129](#)
- PKI\_X509\_PRQP\_REQ\_STACK\_put\_url
  - prqp\_req\_io.c, [130](#)
- PKI\_X509\_PRQP\_REQ\_VALUE\_print\_bio
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_REQ\_verify
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_RESP\_add\_referrals
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_RESP\_add\_service
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_RESP\_add\_service\_stack
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_RESP\_free
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_RESP\_free\_void
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_RESP\_get

- prqp\_resp\_io.c, [130](#)
- PKI\_X509\_PRQP\_RESP\_get\_data
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_RESP\_get\_http
  - http\_client.c, [121](#)
- PKI\_X509\_PRQP\_RESP\_get\_mem
  - prqp\_resp\_io.c, [130](#)
- PKI\_X509\_PRQP\_RESP\_get\_status
  - prqp\_lib.c, [126](#)
- PKI\_X509\_PRQP\_RESP\_get\_url
  - prqp\_resp\_io.c, [130](#)
- PKI\_X509\_PRQP\_RESP\_new\_null
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_new\_req
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_nonce\_dup
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_pkistatus\_set
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_print
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_print\_fp
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_put
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_put\_mem
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_put\_url
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_STACK\_get
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_STACK\_get\_mem
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_STACK\_get\_url
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_STACK\_put
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_STACK\_put\_mem
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_STACK\_put\_url
  - prqp\_resp\_io.c, [131](#)
- PKI\_X509\_PRQP\_RESP\_url\_sk
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_VALUE\_print\_bio
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_verify
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_RESP\_version\_set
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_sign
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_sign\_tk
  - prqp\_lib.c, [127](#)
- PKI\_X509\_PRQP\_STATUS\_STRING
  - prqp\_lib.c, [128](#)
- PKI\_X509\_PRQP\_verify
  - prqp\_lib.c, [128](#)
- PKI\_X509\_put
  - pki\_x509\_io.c, [15](#)
- PKI\_X509\_put\_mem
  - pki\_x509\_mem.c, [119](#)
- PKI\_X509\_put\_mem\_value
  - pki\_x509\_mem.c, [119](#)
- 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, [84](#)
  - PKI\_X509\_REQ\_add\_extension, [84](#)
  - PKI\_X509\_REQ\_add\_extension\_stack, [84](#)
  - PKI\_X509\_REQ\_clear\_attributes, [84](#)
  - PKI\_X509\_REQ\_delete\_attribute, [84](#)
  - PKI\_X509\_REQ\_delete\_attribute\_by\_name, [84](#)
  - PKI\_X509\_REQ\_delete\_attribute\_by\_num, [85](#)
  - PKI\_X509\_REQ\_free, [85](#)
  - PKI\_X509\_REQ\_free\_void, [85](#)
  - PKI\_X509\_REQ\_get\_attribute, [85](#)
  - PKI\_X509\_REQ\_get\_attribute\_by\_name, [85](#)
  - PKI\_X509\_REQ\_get\_attribute\_by\_num, [85](#)
  - PKI\_X509\_REQ\_get\_attributes\_num, [85](#)
  - PKI\_X509\_REQ\_get\_data, [85](#)
  - PKI\_X509\_REQ\_get\_extension\_by\_name, [85](#)
  - PKI\_X509\_REQ\_get\_extension\_by\_num, [85](#)
  - PKI\_X509\_REQ\_get\_extension\_by\_oid, [85](#)
  - PKI\_X509\_REQ\_get\_keysize, [85](#)
  - PKI\_X509\_REQ\_get\_parsed, [85](#)
  - PKI\_X509\_REQ\_new, [85](#)
  - PKI\_X509\_REQ\_new\_null, [86](#)
  - PKI\_X509\_REQ\_print\_parsed, [86](#)
- PKI\_X509\_REQ\_add\_attribute
  - pki\_x509\_req.c, [84](#)
- PKI\_X509\_REQ\_add\_extension
  - pki\_x509\_req.c, [84](#)
- PKI\_X509\_REQ\_add\_extension\_stack
  - pki\_x509\_req.c, [84](#)
- PKI\_X509\_REQ\_clear\_attributes
  - pki\_x509\_req.c, [84](#)
- PKI\_X509\_REQ\_delete\_attribute
  - pki\_x509\_req.c, [84](#)
- PKI\_X509\_REQ\_delete\_attribute\_by\_name
  - pki\_x509\_req.c, [84](#)
- PKI\_X509\_REQ\_delete\_attribute\_by\_num
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_free
  - pki\_x509\_req.c, [85](#)



- PKI\_X509\_REQ\_free\_void
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get
  - pki\_x509\_req\_io.c, [20](#)
- PKI\_X509\_REQ\_get\_attribute
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_attribute\_by\_name
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_attribute\_by\_num
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_attributes\_num
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_data
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_extension\_by\_name
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_extension\_by\_num
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_extension\_by\_oid
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_keysize
  - pki\_x509\_req.c, [85](#)
- PKI\_X509\_REQ\_get\_mem
  - pki\_x509\_req\_io.c, [20](#)
- PKI\_X509\_REQ\_get\_parsed
  - pki\_x509\_req.c, [85](#)
- 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, [85](#)
- PKI\_X509\_REQ\_new\_null
  - pki\_x509\_req.c, [86](#)
- PKI\_X509\_REQ\_print\_parsed
  - pki\_x509\_req.c, [86](#)
- 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, [135](#)
  - PKI\_X509\_SCEP\_ATTRIBUTE\_get\_txt, [135](#)
  - PKI\_X509\_SCEP\_init, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_attr\_value, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_attr\_value\_int, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_failinfo, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_oid, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_proxy, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_recipient\_nonce, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_sender\_nonce, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_status, [135](#)
  - PKI\_X509\_SCEP\_MSG\_get\_trans\_id, [136](#)
  - PKI\_X509\_SCEP\_MSG\_get\_type, [136](#)
  - PKI\_X509\_SCEP\_MSG\_new\_trans\_id, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_attribute, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_attribute\_by\_name, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_attribute\_int, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_failinfo, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_proxy, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_recipient\_nonce, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_sender\_nonce, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_status, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_trans\_id, [136](#)
  - PKI\_X509\_SCEP\_MSG\_set\_type, [137](#)
  - SCEP\_ATTRIBUTE\_list, [137](#)
  - SCEP\_CONF\_LIST\_SIZE, [135](#)
- PKI\_X509\_SCEP\_ATTRIBUTE\_get\_nid
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_ATTRIBUTE\_get\_txt
  - pki\_x509\_scep\_attr.c, [135](#)
- pki\_x509\_scep\_data.c
  - PKI\_X509\_SCEP\_DATA\_add\_recipient, [138](#)
  - PKI\_X509\_SCEP\_DATA\_free, [138](#)
  - PKI\_X509\_SCEP\_DATA\_new, [138](#)
  - PKI\_X509\_SCEP\_DATA\_set\_ias, [138](#)

- PKI\_X509\_SCEP\_DATA\_set\_raw\_data, [138](#)
- PKI\_X509\_SCEP\_DATA\_set\_recipients, [138](#)
- PKI\_X509\_SCEP\_DATA\_set\_x509\_obj, [138](#)
- PKI\_X509\_SCEP\_DATA\_add\_recipient
  - pki\_x509\_scep\_data.c, [138](#)
- PKI\_X509\_SCEP\_DATA\_free
  - pki\_x509\_scep\_data.c, [138](#)
- PKI\_X509\_SCEP\_DATA\_new
  - pki\_x509\_scep\_data.c, [138](#)
- PKI\_X509\_SCEP\_DATA\_set\_ias
  - pki\_x509\_scep\_data.c, [138](#)
- PKI\_X509\_SCEP\_DATA\_set\_raw\_data
  - pki\_x509\_scep\_data.c, [138](#)
- PKI\_X509\_SCEP\_DATA\_set\_recipients
  - pki\_x509\_scep\_data.c, [138](#)
- PKI\_X509\_SCEP\_DATA\_set\_x509\_obj
  - pki\_x509\_scep\_data.c, [138](#)
- PKI\_X509\_SCEP\_init
  - pki\_x509\_scep\_attr.c, [135](#)
- pki\_x509\_scep\_msg.c
  - PKI\_X509\_SCEP\_MSG\_add\_signer, [139](#)
  - PKI\_X509\_SCEP\_MSG\_add\_signer\_tk, [139](#)
  - PKI\_X509\_SCEP\_MSG\_decode, [139](#)
  - PKI\_X509\_SCEP\_MSG\_encode, [139](#)
  - PKI\_X509\_SCEP\_MSG\_free, [140](#)
  - PKI\_X509\_SCEP\_MSG\_new, [140](#)
  - PKI\_X509\_SCEP\_MSG\_new\_certreq, [140](#)
- PKI\_X509\_SCEP\_MSG\_add\_signer
  - pki\_x509\_scep\_msg.c, [139](#)
- PKI\_X509\_SCEP\_MSG\_add\_signer\_tk
  - pki\_x509\_scep\_msg.c, [139](#)
- PKI\_X509\_SCEP\_MSG\_decode
  - pki\_x509\_scep\_msg.c, [139](#)
- PKI\_X509\_SCEP\_MSG\_encode
  - pki\_x509\_scep\_msg.c, [139](#)
- PKI\_X509\_SCEP\_MSG\_free
  - pki\_x509\_scep\_msg.c, [140](#)
- PKI\_X509\_SCEP\_MSG\_get\_attr\_value
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_MSG\_get\_attr\_value\_int
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_MSG\_get\_failinfo
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_MSG\_get\_oid
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_MSG\_get\_proxy
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_MSG\_get\_recipient\_nonce
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_MSG\_get\_sender\_nonce
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_MSG\_get\_status
  - pki\_x509\_scep\_attr.c, [135](#)
- PKI\_X509\_SCEP\_MSG\_get\_trans\_id
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_get\_type
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_new
  - pki\_x509\_scep\_msg.c, [140](#)
- PKI\_X509\_SCEP\_MSG\_new\_certreq
  - pki\_x509\_scep\_msg.c, [140](#)
- PKI\_X509\_SCEP\_MSG\_new\_trans\_id
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_attribute
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_attribute\_by\_name
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_attribute\_int
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_failinfo
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_proxy
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_recipient\_nonce
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_sender\_nonce
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_status
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_trans\_id
  - pki\_x509\_scep\_attr.c, [136](#)
- PKI\_X509\_SCEP\_MSG\_set\_type
  - pki\_x509\_scep\_attr.c, [137](#)
- PKI\_X509\_set\_hsm
  - pki\_x509.c, [117](#)
- PKI\_X509\_set\_reference
  - pki\_x509.c, [118](#)
- PKI\_X509\_set\_value
  - pki\_x509.c, [118](#)
- pki\_x509\_signature.c
  - PKI\_X509\_SIGNATURE\_get\_parsed, [86](#)
- PKI\_X509\_SIGNATURE\_get\_parsed
  - pki\_x509\_signature.c, [86](#)
- PKI\_X509\_STACK\_get
  - pki\_x509\_io.c, [16](#)
- PKI\_X509\_STACK\_get\_mem
  - pki\_x509\_mem.c, [119](#)
- 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, [119](#)
- PKI\_X509\_STACK\_put\_url
  - pki\_x509\_io.c, [16](#)
- pki\_x509\_xpair.c
  - PKI\_X509\_XPAIR\_free, [87](#)
  - PKI\_X509\_XPAIR\_free\_void, [87](#)

- PKI\_X509\_XPAIR\_get\_forward, [87](#)
- PKI\_X509\_XPAIR\_get\_reverse, [87](#)
- PKI\_X509\_XPAIR\_new\_certs, [87](#)
- PKI\_X509\_XPAIR\_new\_null, [87](#)
- PKI\_X509\_XPAIR\_set\_forward, [87](#)
- PKI\_X509\_XPAIR\_set\_reverse, [87](#)
- PKI\_XPAIR\_new\_null, [87](#)
- PKI\_X509\_XPAIR\_free
  - pki\_x509\_xpair.c, [87](#)
- PKI\_X509\_XPAIR\_free\_void
  - pki\_x509\_xpair.c, [87](#)
- PKI\_X509\_XPAIR\_get
  - pki\_x509\_xpair\_io.c, [22](#)
- PKI\_X509\_XPAIR\_get\_forward
  - pki\_x509\_xpair.c, [87](#)
- PKI\_X509\_XPAIR\_get\_reverse
  - pki\_x509\_xpair.c, [87](#)
- 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, [87](#)
- PKI\_X509\_XPAIR\_new\_null
  - pki\_x509\_xpair.c, [87](#)
- 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, [87](#)
- PKI\_X509\_XPAIR\_set\_reverse
  - pki\_x509\_xpair.c, [87](#)
- 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, [87](#)
- PKI\_XPAIR\_print
  - pki\_x509\_xpair\_io.c, [23](#)
- PKI\_ZFree
  - pki\_mem.c, [101](#)
- PKI\_ZFree\_str
  - pki\_mem.c, [102](#)
- profile.c
  - PKI\_X509\_PROFILE\_add\_extension, [120](#)
  - PKI\_X509\_PROFILE\_free, [120](#)
  - PKI\_X509\_PROFILE\_free\_void, [120](#)
  - PKI\_X509\_PROFILE\_get\_default, [120](#)
  - PKI\_X509\_PROFILE\_get\_ext\_by\_num, [120](#)
  - PKI\_X509\_PROFILE\_get\_extensions, [120](#)
  - PKI\_X509\_PROFILE\_get\_exts\_num, [120](#)
  - PKI\_X509\_PROFILE\_get\_name, [120](#)
  - PKI\_X509\_PROFILE\_get\_value, [120](#)
  - PKI\_X509\_PROFILE\_load, [120](#)
  - PKI\_X509\_PROFILE\_new, [120](#)
  - PKI\_X509\_PROFILE\_put\_file, [120](#)
- proto\_list
  - url.c, [39](#)
- prqp\_bio.c
  - d2i\_PRQP\_REQ\_bio, [122](#)
  - d2i\_PRQP\_RESP\_bio, [122](#)
  - i2d\_PRQP\_REQ\_bio, [122](#)
  - i2d\_PRQP\_RESP\_bio, [122](#)
  - PEM\_read\_bio\_PRQP\_REQ, [122](#)
  - PEM\_read\_bio\_PRQP\_RESP, [122](#)
  - PEM\_write\_bio\_PRQP\_REQ, [122](#)
  - PEM\_write\_bio\_PRQP\_RESP, [122](#)
- PRQP\_init\_all\_services
  - prqp\_lib.c, [128](#)
- prqp\_lib.c
  - \_\_PKI\_PRQP\_LIB\_C\_\_, [125](#)
  - CERT\_IDENTIFIER\_cmp, [125](#)
  - PKI\_PRQP\_CERTID\_new, [125](#)
  - PKI\_PRQP\_CERTID\_new\_cert, [125](#)
  - PKI\_PRQP\_REQ\_new\_cert, [125](#)
  - PKI\_X509\_PRQP\_NONCE\_new, [125](#)
  - PKI\_X509\_PRQP\_REQ\_add\_service, [125](#)
  - PKI\_X509\_PRQP\_REQ\_add\_service\_stack, [125](#)
  - PKI\_X509\_PRQP\_REQ\_free, [125](#)
  - PKI\_X509\_PRQP\_REQ\_free\_void, [125](#)
  - PKI\_X509\_PRQP\_REQ\_get\_data, [125](#)
  - PKI\_X509\_PRQP\_REQ\_new\_certs\_res, [125](#)
  - PKI\_X509\_PRQP\_REQ\_new\_null, [126](#)
  - PKI\_X509\_PRQP\_REQ\_new\_url, [126](#)
  - PKI\_X509\_PRQP\_REQ\_print, [126](#)
  - PKI\_X509\_PRQP\_REQ\_print\_fp, [126](#)



- PKI\_X509\_PRQP\_REQ\_VALUE\_print\_bio, 126
- PKI\_X509\_PRQP\_REQ\_verify, 126
- PKI\_X509\_PRQP\_RESP\_add\_referrals, 126
- PKI\_X509\_PRQP\_RESP\_add\_service, 126
- PKI\_X509\_PRQP\_RESP\_add\_service\_stack, 126
- PKI\_X509\_PRQP\_RESP\_free, 126
- PKI\_X509\_PRQP\_RESP\_free\_void, 126
- PKI\_X509\_PRQP\_RESP\_get\_data, 126
- PKI\_X509\_PRQP\_RESP\_get\_status, 126
- PKI\_X509\_PRQP\_RESP\_new\_null, 127
- PKI\_X509\_PRQP\_RESP\_new\_req, 127
- PKI\_X509\_PRQP\_RESP\_nonce\_dup, 127
- PKI\_X509\_PRQP\_RESP\_pkistatus\_set, 127
- PKI\_X509\_PRQP\_RESP\_print, 127
- PKI\_X509\_PRQP\_RESP\_print\_fp, 127
- PKI\_X509\_PRQP\_RESP\_url\_sk, 127
- PKI\_X509\_PRQP\_RESP\_VALUE\_print\_bio, 127
- PKI\_X509\_PRQP\_RESP\_verify, 127
- PKI\_X509\_PRQP\_RESP\_version\_set, 127
- PKI\_X509\_PRQP\_sign, 127
- PKI\_X509\_PRQP\_sign\_tk, 127
- PKI\_X509\_PRQP\_STATUS\_STRING, 128
- PKI\_X509\_PRQP\_verify, 128
- PRQP\_init\_all\_services, 128
- PRQP\_RESOURCE\_RESPONSE\_TOKEN\_-get\_oid, 128
- PRQP\_RESOURCE\_RESPONSE\_TOKEN\_-get\_services, 128
- prqp\_req\_io.c
  - PKI\_X509\_PRQP\_REQ\_get, 129
  - PKI\_X509\_PRQP\_REQ\_get\_mem, 129
  - PKI\_X509\_PRQP\_REQ\_get\_url, 129
  - PKI\_X509\_PRQP\_REQ\_put, 129
  - PKI\_X509\_PRQP\_REQ\_put\_mem, 129
  - PKI\_X509\_PRQP\_REQ\_put\_url, 129
  - PKI\_X509\_PRQP\_REQ\_STACK\_get, 129
  - PKI\_X509\_PRQP\_REQ\_STACK\_get\_mem, 129
  - PKI\_X509\_PRQP\_REQ\_STACK\_get\_url, 129
  - PKI\_X509\_PRQP\_REQ\_STACK\_put, 129
  - PKI\_X509\_PRQP\_REQ\_STACK\_put\_mem, 129
  - PKI\_X509\_PRQP\_REQ\_STACK\_put\_url, 130
- PRQP\_RESOURCE\_RESPONSE\_TOKEN\_get\_oid
  - prqp\_lib.c, 128
- PRQP\_RESOURCE\_RESPONSE\_TOKEN\_get\_services
  - prqp\_lib.c, 128
- prqp\_resp\_io.c
  - PKI\_X509\_PRQP\_RESP\_get, 130
  - PKI\_X509\_PRQP\_RESP\_get\_mem, 130
  - PKI\_X509\_PRQP\_RESP\_get\_url, 130
  - PKI\_X509\_PRQP\_RESP\_put, 131
  - PKI\_X509\_PRQP\_RESP\_put\_mem, 131
  - PKI\_X509\_PRQP\_RESP\_put\_url, 131
  - PKI\_X509\_PRQP\_RESP\_STACK\_get, 131
  - PKI\_X509\_PRQP\_RESP\_STACK\_get\_mem, 131
  - PKI\_X509\_PRQP\_RESP\_STACK\_get\_url, 131
  - PKI\_X509\_PRQP\_RESP\_STACK\_put, 131
  - PKI\_X509\_PRQP\_RESP\_STACK\_put\_mem, 131
  - PKI\_X509\_PRQP\_RESP\_STACK\_put\_url, 131
- prqp\_srv.c
  - PKI\_DISCOVER\_get\_resp, 132
  - PKI\_DISCOVER\_get\_resp\_url, 132
  - PKI\_get\_ca\_resources, 132
  - PKI\_get\_ca\_service, 132
  - PKI\_get\_ca\_service\_sk, 132
  - PKI\_get\_cert\_service\_sk, 133
- pthread\_init.c
  - \_dyn\_create\_callback, 88
  - \_dyn\_destroy\_callback, 88
  - \_dyn\_lock\_callback, 89
  - CRYPTO\_LOCK, 88
  - CRYPTO\_READ, 88
  - CRYPTO\_UNLOCK, 88
  - CRYPTO\_WRITE, 88
  - lock\_cs, 89
  - thread\_cleanup, 89
  - win32\_locking\_callback, 89
- SCEP\_ATTRIBUTE\_list
  - pki\_x509\_scep\_attr.c, 137
- SCEP\_CONF\_LIST\_SIZE
  - pki\_x509\_scep\_attr.c, 135
- sock.c
  - \_Close, 31
  - \_Connect, 31
  - \_Listen, 31
  - \_Read, 31
  - \_Select, 31
  - \_Shutdown, 32
  - \_Socket, 32
  - \_Write, 32
  - Gethostbyname, 32
  - h\_errno, 33
  - inet\_close, 32
  - inet\_connect, 32
  - LISTENQ, 31

- PKI\_NET\_accept, 32
- PKI\_NET\_close, 32
- PKI\_NET\_get\_data, 32
- PKI\_NET\_listen, 32
- PKI\_NET\_open, 32
- PKI\_NET\_read, 32
- PKI\_NET\_socket, 32
- PKI\_NET\_write, 32
- 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/pki\_socket.c, 28
- src/net/sock.c, 30
- src/net/ssl.c, 33
- src/net/url.c, 36
- src/openssl/pki\_algor.c, 39
- src/openssl/pki\_algorithm.c, 43
- src/openssl/pki\_digest.c, 43
- src/openssl/pki\_id.c, 45
- src/openssl/pki\_integer.c, 45
- src/openssl/pki\_keypair.c, 47
- src/openssl/pki\_keyparams.c, 50
- src/openssl/pki\_ocsp\_req.c, 51
- src/openssl/pki\_ocsp\_resp.c, 54
- src/openssl/pki\_oid.c, 57
- src/openssl/pki\_string.c, 59
- src/openssl/pki\_time.c, 60
- src/openssl/pki\_x509\_attribute.c, 62
- src/openssl/pki\_x509\_cert.c, 64
- src/openssl/pki\_x509\_crl.c, 68
- src/openssl/pki\_x509\_extension.c, 71
- src/openssl/pki\_x509\_name.c, 72
- src/openssl/pki\_x509\_p12.c, 74
- src/openssl/pki\_x509\_pkcs7.c, 77
- src/openssl/pki\_x509\_req.c, 83
- src/openssl/pki\_x509\_signature.c, 86
- src/openssl/pki\_x509\_xpair.c, 86
- src/openssl/pki\_x509\_xpair\_asn1.c, 88
- src/openssl/pthread\_init.c, 88
- src/pki\_config.c, 89
- src/pki\_cred.c, 94
- src/pki\_err.c, 95
- src/pki\_init.c, 96
- src/pki\_log.c, 97
- src/pki\_mem.c, 98
- src/pki\_msg\_req.c, 102
- src/pki\_msg\_resp.c, 106
- src/pki\_threads.c, 111
- src/pki\_threads\_vars.c, 111
- src/pki\_x509.c, 115
- src/pki\_x509\_mem.c, 118
- src/pki\_x509\_mime.c, 119
- src/profile.c, 119
- src/prqp/asn1\_req.c, 121
- src/prqp/asn1\_res.c, 121
- src/prqp/http\_client.c, 121
- src/prqp/prqp\_bio.c, 121
- src/prqp/prqp\_lib.c, 122
- src/prqp/prqp\_req\_io.c, 128
- src/prqp/prqp\_resp\_io.c, 130
- src/prqp/prqp\_srv.c, 131
- src/scep/pki\_x509\_scep\_asn1.c, 133
- src/scep/pki\_x509\_scep\_attr.c, 133
- src/scep/pki\_x509\_scep\_data.c, 137
- src/scep/pki\_x509\_scep\_msg.c, 139
- src/stack.c, 140
- src/support.c, 142
- src/token.c, 143
- src/token\_data.c, 155
- src/token\_id.c, 156
- ssl.c
  - \_\_pki\_ssl\_start\_ssl, 34
  - BUFF\_MAX\_SIZE, 34
  - PKI\_SSL\_check\_verify, 34
  - PKI\_SSL\_close, 34
  - PKI\_SSL\_connect, 34
  - PKI\_SSL\_connect\_url, 34
  - PKI\_SSL\_dup, 35
  - PKI\_SSL\_free, 35
  - PKI\_SSL\_get\_fd, 35
  - PKI\_SSL\_get\_peer\_cert, 35
  - PKI\_SSL\_get\_peer\_chain, 35
  - PKI\_SSL\_get\_servername, 35
  - PKI\_SSL\_new, 35
  - PKI\_SSL\_read, 35
  - PKI\_SSL\_set\_algor, 35
  - PKI\_SSL\_set\_cipher, 35
  - PKI\_SSL\_set\_fd, 35
  - PKI\_SSL\_set\_flags, 35
  - PKI\_SSL\_set\_token, 35

- PKI\_SSL\_set\_trusted, 36
- PKI\_SSL\_set\_verify, 36
- PKI\_SSL\_start\_ssl, 36
- PKI\_SSL\_write, 36
- stack.c
  - PKI\_STACK\_del\_num, 141
  - PKI\_STACK\_elements, 141
  - PKI\_STACK\_free, 141
  - PKI\_STACK\_free\_all, 141
  - PKI\_STACK\_get\_num, 141
  - PKI\_STACK\_ins\_num, 142
  - PKI\_STACK\_new, 142
  - PKI\_STACK\_new\_null, 142
  - PKI\_STACK\_new\_type, 142
  - PKI\_STACK\_pop, 142
  - PKI\_STACK\_pop\_free, 142
  - PKI\_STACK\_push, 142
- strcmp\_nocase
  - support.c, 143
- strncmp\_nocase
  - support.c, 143
- strstr\_nocase
  - support.c, 143
- support.c
  - get\_env\_string, 143
  - PKI\_get\_env, 143
  - PKI\_set\_env, 143
  - strcmp\_nocase, 143
  - strncmp\_nocase, 143
  - strstr\_nocase, 143
- thread\_cleanup
  - pthread\_init.c, 89
- token.c
  - PKI\_TOKEN\_add\_profile, 147
  - PKI\_TOKEN\_check, 147
  - PKI\_TOKEN\_cred\_cb\_env, 147
  - PKI\_TOKEN\_cred\_cb\_stdin, 147
  - PKI\_TOKEN\_cred\_get, 148
  - PKI\_TOKEN\_cred\_set\_cb, 148
  - PKI\_TOKEN\_del\_url, 148
  - PKI\_TOKEN\_export\_cert, 148
  - PKI\_TOKEN\_export\_keypair, 148
  - PKI\_TOKEN\_export\_keypair\_url, 148
  - PKI\_TOKEN\_export\_otherCerts, 148
  - PKI\_TOKEN\_export\_p12, 148
  - PKI\_TOKEN\_export\_req, 148
  - PKI\_TOKEN\_export\_trustedCerts, 148
  - PKI\_TOKEN\_free, 148
  - PKI\_TOKEN\_free\_void, 148
  - PKI\_TOKEN\_get\_algor, 149
  - PKI\_TOKEN\_get\_algor\_id, 149
  - PKI\_TOKEN\_get\_cacert, 149
  - PKI\_TOKEN\_get\_cert, 149
  - PKI\_TOKEN\_get\_config\_dir, 149
  - PKI\_TOKEN\_get\_cred, 149
  - PKI\_TOKEN\_get\_crls, 149
  - PKI\_TOKEN\_get\_keypair, 149
  - PKI\_TOKEN\_get\_name, 149
  - PKI\_TOKEN\_get\_otherCerts, 150
  - PKI\_TOKEN\_get\_p12, 150
  - PKI\_TOKEN\_get\_trustedCerts, 150
  - PKI\_TOKEN\_import\_cert, 150
  - PKI\_TOKEN\_import\_cert\_stack, 150
  - PKI\_TOKEN\_import\_keypair, 150
  - PKI\_TOKEN\_init, 150
  - PKI\_TOKEN\_issue\_cert, 150
  - PKI\_TOKEN\_issue\_crl, 150
  - PKI\_TOKEN\_issue\_proxy, 150
  - PKI\_TOKEN\_load\_cacert, 151
  - PKI\_TOKEN\_load\_cert, 151
  - PKI\_TOKEN\_load\_config, 151
  - PKI\_TOKEN\_load\_crls, 151
  - PKI\_TOKEN\_load\_keypair, 151
  - PKI\_TOKEN\_load\_otherCerts, 151
  - PKI\_TOKEN\_load\_profiles, 151
  - PKI\_TOKEN\_load\_req, 152
  - PKI\_TOKEN\_load\_trustedCerts, 152
  - PKI\_TOKEN\_login, 152
  - PKI\_TOKEN\_new, 152
  - PKI\_TOKEN\_new\_keypair, 152
  - PKI\_TOKEN\_new\_keypair\_ex, 152
  - PKI\_TOKEN\_new\_keypair\_url, 152
  - PKI\_TOKEN\_new\_keypair\_url\_ex, 153
  - PKI\_TOKEN\_new\_null, 153
  - PKI\_TOKEN\_new\_p12, 153
  - PKI\_TOKEN\_new\_req, 153
  - PKI\_TOKEN\_OID\_new, 153
  - PKI\_TOKEN\_print\_info, 153
  - PKI\_TOKEN\_search\_profile, 153
  - PKI\_TOKEN\_self\_sign, 153
  - PKI\_TOKEN\_set\_algor, 153
  - PKI\_TOKEN\_set\_algor\_by\_name, 154
  - PKI\_TOKEN\_set\_cacert, 154
  - PKI\_TOKEN\_set\_cert, 154
  - PKI\_TOKEN\_set\_config\_dir, 154
  - PKI\_TOKEN\_set\_cred, 154
  - PKI\_TOKEN\_set\_crls, 154
  - PKI\_TOKEN\_set\_keypair, 155
  - PKI\_TOKEN\_set\_otherCerts, 155
  - PKI\_TOKEN\_set\_req, 155
  - PKI\_TOKEN\_set\_trustedCerts, 155
  - PKI\_TOKEN\_use\_slot, 155
- token\_data.c
  - PKI\_TOKEN\_get\_cacert\_data, 156
  - PKI\_TOKEN\_get\_cert\_data, 156
  - PKI\_TOKEN\_get\_identity\_data, 156
  - PKI\_TOKEN\_get\_keypair\_data, 156

- PKI\_TOKEN\_get\_otherCerts\_data, 156
- PKI\_TOKEN\_get\_privkey\_data, 156
- PKI\_TOKEN\_get\_pubkey\_data, 156
- PKI\_TOKEN\_get\_trustedCerts\_data, 156
- token\_id.c
  - PKI\_TOKEN\_ID\_INFO\_get, 157
  - PKI\_TOKEN\_ID\_INFO\_list, 157
  - PKI\_TOKEN\_ID\_num, 157
  - PKI\_TOKEN\_ID\_set, 157
- URI\_PROTOCOL
  - url.c, 37
- url.c
  - BUFF\_MAX\_SIZE, 37
  - proto\_list, 39
  - URI\_PROTOCOL, 37
  - URL\_free, 37
  - URL\_get\_data, 37
  - URL\_get\_data\_fd, 37
  - URL\_get\_data\_file, 38
  - URL\_get\_data\_socket, 38
  - URL\_get\_data\_url, 38
  - URL\_get\_local\_addr, 38
  - URL\_new, 38
  - URL\_proto\_to\_string, 38
  - URL\_put\_data, 38
  - URL\_put\_data\_fd, 38
  - URL\_put\_data\_file, 39
  - URL\_put\_data\_socket, 39
  - URL\_put\_data\_url, 39
- URL\_free
  - url.c, 37
- URL\_get\_data
  - url.c, 37
- URL\_get\_data\_fd
  - url.c, 37
- URL\_get\_data\_file
  - url.c, 38
- 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\_socket
  - url.c, 38
- URL\_get\_data\_url
  - url.c, 38
- URL\_get\_local\_addr
  - url.c, 38
- URL\_new
  - url.c, 38
- URL\_proto\_to\_string
  - url.c, 38
- URL\_put\_data
  - url.c, 38
- URL\_put\_data\_fd
  - url.c, 38
- URL\_put\_data\_file
  - url.c, 39
- 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\_socket
  - url.c, 39
- URL\_put\_data\_url
  - url.c, 39
- win32\_locking\_callback
  - pthread\_init.c, 89
- xmlErrorPtr
  - pki\_config.c, 91