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

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'd2i_PUBKEY_bio.3ossl' command

$ man d2i_PUBKEY_bio.3ossl

X509_PUBKEY_NEW(3ossl) OpenSSL X509_PUBKEY_NEW(3ossl)

NAME

X509_PUBKEY_new_ex, X509_PUBKEY_new, X509_PUBKEY_free, X509_PUBKEY_dup, X509_PUBKEY_set, X509_PUBKEY_get0, X509_PUBKEY_get, d2i_PUBKEY_ex, d2i_PUBKEY, i2d_PUBKEY, d2i_PUBKEY_bio, d2i_PUBKEY_fp, i2d_PUBKEY_fp, i2d_PUBKEY_bio, X509_PUBKEY_set0_param, X509_PUBKEY_get0_param, X509_PUBKEY_eq - SubjectPublicKeyInfo public key functions

SYNOPSIS

```
#include <openssl/x509.h>
X509_PUBKEY *X509_PUBKEY_new_ex(OSSL_LIB_CTX *libctx, const char *propq);
X509_PUBKEY *X509_PUBKEY_new(void);
void X509_PUBKEY_free(X509_PUBKEY *a);
X509_PUBKEY *X509_PUBKEY_dup(const X509_PUBKEY *a);
int X509_PUBKEY_set(X509_PUBKEY **x, EVP_PKEY *pkey);
EVP_PKEY *X509_PUBKEY_get0(const X509_PUBKEY *key);
EVP_PKEY *X509_PUBKEY_get(const X509_PUBKEY *key);
EVP_PKEY *d2i_PUBKEY_ex(EVP_PKEY **a, const unsigned char **pp, long length,
                        OSSL_LIB_CTX *libctx, const char *propq);
EVP_PKEY *d2i_PUBKEY(EVP_PKEY **a, const unsigned char **pp, long length);
int i2d_PUBKEY(const EVP_PKEY *a, unsigned char **pp);
EVP_PKEY *d2i_PUBKEY_bio(BIO *bp, EVP_PKEY **a);
EVP_PKEY *d2i_PUBKEY_fp(FILE *fp, EVP_PKEY **a);
int i2d_PUBKEY_fp(const FILE *fp, EVP_PKEY *pkey);
```

```
int i2d_PUBKEY_bio(BIO *bp, const EVP_PKEY *pkey);
int X509_PUBKEY_set0_param(X509_PUBKEY *pub, ASN1_OBJECT *aobj,
                           int ptype, void *pval,
                           unsigned char *penc, int penclen);
int X509_PUBKEY_get0_param(ASN1_OBJECT **ppkalg,
                           const unsigned char **pk, int *ppklen,
                           X509_ALGOR **pa, const X509_PUBKEY *pub);
int X509_PUBKEY_eq(X509_PUBKEY *a, X509_PUBKEY *b);
```

DESCRIPTION

The X509_PUBKEY structure represents the ASN.1 SubjectPublicKeyInfo structure defined in RFC5280 and used in certificates and certificate requests.

X509_PUBKEY_new_ex() allocates and initializes an X509_PUBKEY structure associated with the given OSSL_LIB_CTX in the libctx parameter. Any algorithm fetches associated with using the X509_PUBKEY object will use the property query string propq. See "ALGORITHM FETCHING" in crypto(7) for further information about algorithm fetching.

X509_PUBKEY_new() is the same as X509_PUBKEY_new_ex() except that the default (NULL) OSSL_LIB_CTX and a NULL property query string are used.

X509_PUBKEY_dup() creates a duplicate copy of the X509_PUBKEY object specified by a.

X509_PUBKEY_free() frees up X509_PUBKEY structure a. If a is NULL nothing is done.

X509_PUBKEY_set() sets the public key in *x to the public key contained in the EVP_PKEY structure pkey. If *x is not NULL any existing public key structure will be freed.

X509_PUBKEY_get0() returns the public key contained in key. The returned value is an internal pointer which MUST NOT be freed after use.

X509_PUBKEY_get() is similar to X509_PUBKEY_get0() except the reference count on the returned key is incremented so it MUST be freed using EVP_PKEY_free() after use.

d2i_PUBKEY_ex() decodes an EVP_PKEY structure using

SubjectPublicKeyInfo format. Some public key decoding implementations may use cryptographic algorithms. In this case the supplied library context libctx and property query string propq are used. d2i_PUBKEY() does the same as d2i_PUBKEY_ex() except that the default library context and property query string are used.

i2d_PUBKEY() encodes an EVP_PKEY structure using SubjectPublicKeyInfo format.

d2i_PUBKEY_bio(), d2i_PUBKEY_fp(), i2d_PUBKEY_bio() and i2d_PUBKEY_fp() are similar to d2i_PUBKEY() and i2d_PUBKEY() except they decode or encode using a BIO or FILE pointer.

X509_PUBKEY_set0_param() sets the public key parameters of pub. The OID associated with the algorithm is set to aobj. The type of the algorithm parameters is set to type using the structure pval. The encoding of the public key itself is set to the penclen bytes contained in buffer penc. On success ownership of all the supplied parameters is passed to pub so they must not be freed after the call.

X509_PUBKEY_get0_param() retrieves the public key parameters from pub, *ppkalg is set to the associated OID and the encoding consists of *ppklen bytes at *pk, *pa is set to the associated AlgorithmIdentifier for the public key. If the value of any of these parameters is not required it can be set to NULL. All of the retrieved pointers are internal and must not be freed after the call.

X509_PUBKEY_eq() compares two X509_PUBKEY values.

NOTES

The X509_PUBKEY functions can be used to encode and decode public keys in a standard format.

In many cases applications will not call the X509_PUBKEY functions directly: they will instead call wrapper functions such as X509_get0_pubkey().

RETURN VALUES

If the allocation fails, X509_PUBKEY_new() and X509_PUBKEY_dup() return NULL and set an error code that can be obtained by ERR_get_error(3).

Otherwise they return a pointer to the newly allocated structure.

X509_PUBKEY_free() does not return a value.

X509_PUBKEY_get0() and X509_PUBKEY_get() return a pointer to an EVP_PKEY structure or NULL if an error occurs.

X509_PUBKEY_set(), X509_PUBKEY_set0_param() and X509_PUBKEY_get0_param() return 1 for success and 0 if an error occurred.

X509_PUBKEY_eq() returns 1 for equal, 0 for different, and < 0 on error.

SEE ALSO

d2i_X509(3), ERR_get_error(3), X509_get_pubkey(3),

HISTORY

The X509_PUBKEY_new_ex() and X509_PUBKEY_eq() functions were added in OpenSSL 3.0.

COPYRIGHT