NAME

d2i_PrivateKey_ex, d2i_PrivateKey, d2i_PublicKey, d2i_KeyParams, d2i_AutoPrivateKey_ex, d2i_AutoPrivateKey, i2d_PrivateKey, i2d_PublicKey, i2d_KeyParams, i2d_KeyParams_bio, d2i_PrivateKey_ex_bio, d2i_PrivateKey_bio, d2i_PrivateKey_ex_fp, d2i_PrivateKey_fp, d2i_KeyParams_bio, i2d_PrivateKey_bio, i2d_PrivateKey_fp - decode and encode functions for reading and saving EVP_PKEY structures

SYNOPSIS

```
#include <openssl/evp.h>

EVP_PKEY *d2i_PrivateKey_ex(int type, EVP_PKEY **a, const unsigned char **pp,
                            long length, OSSL_LIB_CTX *libctx,
                            const char *propq);
EVP_PKEY *d2i_PrivateKey(int type, EVP_PKEY **a, const unsigned char **pp,
                         long length);
EVP_PKEY *d2i_PublicKey(int type, EVP_PKEY **a, const unsigned char **pp,
                        long length);
EVP_PKEY *d2i_KeyParams(int type, EVP_PKEY **a, const unsigned char **pp,
                        long length);
EVP_PKEY *d2i_AutoPrivateKey_ex(EVP_PKEY **a, const unsigned char **pp,
                                long length, OSSL_LIB_CTX *libctx,
                                const char *propq);
EVP_PKEY *d2i_AutoPrivateKey(EVP_PKEY **a, const unsigned char **pp,
                             long length);

int i2d_PrivateKey(const EVP_PKEY *a, unsigned char **pp);
int i2d_PublicKey(const EVP_PKEY *a, unsigned char **pp);
int i2d_KeyParams(const EVP_PKEY *a, unsigned char **pp);
int i2d_KeyParams_bio(BIO *bp, const EVP_PKEY *pkey);
EVP_PKEY *d2i_KeyParams_bio(int type, EVP_PKEY **a, BIO *in);


#include <openssl/x509.h>

EVP_PKEY *d2i_PrivateKey_ex_bio(BIO *bp, EVP_PKEY **a, OSSL_LIB_CTX *libctx,
                                const char *propq);
EVP_PKEY *d2i_PrivateKey_bio(BIO *bp, EVP_PKEY **a);
EVP_PKEY *d2i_PrivateKey_ex_fp(FILE *fp, EVP_PKEY **a, OSSL_LIB_CTX *libctx,
                               const char *propq);
EVP_PKEY *d2i_PrivateKey_fp(FILE *fp, EVP_PKEY **a);
```

```
int i2d_PrivateKey_bio(BIO *bp, const EVP_PKEY *pkey);
int i2d_PrivateKey_fp(FILE *fp, const EVP_PKEY *pkey);
```

DESCRIPTION

d2i_PrivateKey_ex() decodes a private key using algorithm *type*. It attempts to use any key-specific format or PKCS#8 unencrypted PrivateKeyInfo format. The *type* parameter should be a public key algorithm constant such as **EVP_PKEY_RSA**. An error occurs if the decoded key does not match *type*. Some private key decoding implementations may use cryptographic algorithms (for example to automatically derive the public key if it is not explicitly included in the encoding). In this case the supplied library context *libctx* and property query string *propq* are used. If successful and the *a* parameter is not NULL the function assigns the returned **EVP_PKEY** structure pointer to **a*, overwriting any previous value.

d2i_PrivateKey() does the same as **d2i_PrivateKey_ex()** except that the default library context and property query string are used. **d2i_PublicKey()** does the same for public keys. **d2i_KeyParams()** does the same for key parameters.

The **d2i_PrivateKey_ex_bio()** and **d2i_PrivateKey_bio()** functions are similar to **d2i_PrivateKey_ex()** and **d2i_PrivateKey()** respectively except that they decode the data read from the given BIO. The **d2i_PrivateKey_ex_fp()** and **d2i_PrivateKey_fp()** functions are the same except that they read the data from the given FILE.

d2i_AutoPrivateKey_ex() and **d2i_AutoPrivateKey()** are similar to **d2i_PrivateKey_ex()** and **d2i_PrivateKey()** respectively except that they attempt to automatically detect the private key format.

i2d_PrivateKey() encodes *a*. It uses a key specific format or, if none is defined for that key type, PKCS#8 unencrypted PrivateKeyInfo format. **i2d_PublicKey()** does the same for public keys. **i2d_KeyParams()** does the same for key parameters. These functions are similar to the **d2i_X509()** functions; see **d2i_X509**(3). **i2d_PrivateKey_bio()** and **i2d_PrivateKey_fp()** do the same thing except that they encode to a **BIO** or **FILE** respectively. Again, these work similarly to the functions described in **d2i_X509**(3).

NOTES

All the functions that operate on data in memory update the data pointer **pp* after a successful operation, just like the other d2i and i2d functions; see **d2i_X509**(3).

All these functions use DER format and unencrypted keys. Applications wishing to encrypt or decrypt private keys should use other functions such as **d2i_PKCS8PrivateKey()** instead.

To decode a key with type **EVP_PKEY_EC**, **d2i_PublicKey()** requires **a* to be a non-NULL

EVP_PKEY structure assigned an EC_KEY structure referencing the proper EC_GROUP.

RETURN VALUES

The **d2i_PrivateKey_ex()**, **d2i_PrivateKey()**, **d2i_AutoPrivateKey_ex()**, **d2i_AutoPrivateKey()**, **d2i_PrivateKey_ex_bio()**, **d2i_PrivateKey_bio()**, **d2i_PrivateKey_ex_fp()**, **d2i_PrivateKey_fp()**, **d2i_PublicKey()**, **d2i_KeyParams()** and **d2i_KeyParams_bio()** functions return a valid **EVP_PKEY** structure or NULL if an error occurs. The error code can be obtained by calling **ERR_get_error**(3).

i2d_PrivateKey(), **i2d_PublicKey()** and **i2d_KeyParams()** return the number of bytes successfully encoded or a negative value if an error occurs. The error code can be obtained by calling **ERR_get_error**(3).

i2d_PrivateKey_bio(), **i2d_PrivateKey_fp()** and **i2d_KeyParams_bio()** return 1 if successfully encoded or zero if an error occurs.

SEE ALSO

crypto(7), **d2i_PKCS8PrivateKey_bio**(3)

HISTORY

d2i_PrivateKey_ex(), **d2i_PrivateKey_ex_bio()**, **d2i_PrivateKey_ex_fp()**, and **d2i_AutoPrivateKey_ex()** were added in OpenSSL 3.0.

COPYRIGHT