NAME

d2i_DSAPrivateKey, d2i_DSAPrivateKey_bio, d2i_DSAPrivateKey_fp, d2i_DSAPublicKey, d2i_DSA_PUBKEY, d2i_DSA_PUBKEY_bio, d2i_DSA_PUBKEY_fp, d2i_DSAparams, d2i_RSAPrivateKey, d2i_RSAPrivateKey_bio, d2i_RSAPrivateKey_fp, d2i_RSAPublicKey, d2i_RSAPublicKey_bio, d2i_RSAPublicKey_fp, d2i_RSA_PUBKEY, d2i_RSA_PUBKEY_bio, d2i_RSA_PUBKEY_fp, d2i_DHparams, d2i_DHparams_bio, d2i_DHparams_fp, d2i_ECParameters, d2i_ECPrivateKey, d2i_ECPrivateKey_bio, d2i_ECPrivateKey_fp, d2i_EC_PUBKEY, d2i_EC_PUBKEY_bio, d2i_EC_PUBKEY_fp, i2d_RSAPrivateKey, i2d_RSAPrivateKey_bio, i2d_RSAPrivateKey_fp, i2d_RSAPublicKey, i2d_RSAPublicKey_bio, i2d_RSAPublicKey_fp, i2d_RSA_PUBKEY, i2d_RSA_PUBKEY_bio, i2d_RSA_PUBKEY_fp, i2d_DHparams, i2d_DHparams_bio, i2d_DHparams_fp, i2d_DSAPrivateKey, i2d_DSAPrivateKey_bio, i2d_DSAPrivateKey_fp, i2d_DSAPublicKey, i2d_DSA_PUBKEY, i2d_DSA_PUBKEY_bio, i2d_DSA_PUBKEY_fp, i2d_DSAparams, i2d_ECParameters, i2d_ECPrivateKey, i2d_ECPrivateKey_bio, i2d_ECPrivateKey_fp, i2d_EC_PUBKEY, i2d_EC_PUBKEY_bio, i2d_EC_PUBKEY_fp - DEPRECATED

SYNOPSIS

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining **OPENSSL_API_COMPAT** with a suitable version value, see **openssl_user_macros**(7):

```
TYPE *d2i_TYPEPrivateKey(TYPE **a, const unsigned char **ppin, long length);
TYPE *d2i_TYPEPrivateKey_bio(BIO *bp, TYPE **a);
TYPE *d2i_TYPEPrivateKey_fp(FILE *fp, TYPE **a);
TYPE *d2i_TYPEPublicKey(TYPE **a, const unsigned char **ppin, long length);
TYPE *d2i_TYPEPublicKey_bio(BIO *bp, TYPE **a);
TYPE *d2i_TYPEPublicKey_fp(FILE *fp, TYPE **a);
TYPE *d2i_TYPEparams(TYPE **a, const unsigned char **ppin, long length);
TYPE *d2i_TYPEparams_bio(BIO *bp, TYPE **a);
TYPE *d2i_TYPEparams_fp(FILE *fp, TYPE **a);
TYPE *d2i_TYPE_PUBKEY(TYPE **a, const unsigned char **ppin, long length);
TYPE *d2i_TYPE_PUBKEY_bio(BIO *bp, TYPE **a);
TYPE *d2i_TYPE_PUBKEY_fp(FILE *fp, TYPE **a);

int i2d_TYPEPrivateKey(const TYPE *a, unsigned char **ppout);
int i2d_TYPEPrivateKey(TYPE *a, unsigned char **ppout);
int i2d_TYPEPrivateKey_fp(FILE *fp, const TYPE *a);
int i2d_TYPEPrivateKey_fp(FILE *fp, TYPE *a);
int i2d_TYPEPrivateKey_bio(BIO *bp, const TYPE *a);
int i2d_TYPEPrivateKey_bio(BIO *bp, TYPE *a);
int i2d_TYPEPublicKey(const TYPE *a, unsigned char **ppout);
```

```
int i2d_TYPEPublicKey(TYPE *a, unsigned char **ppout);
int i2d_TYPEPublicKey_fp(FILE *fp, const TYPE *a);
int i2d_TYPEPublicKey_fp(FILE *fp, TYPE *a);
int i2d_TYPEPublicKey_bio(BIO *bp, const TYPE *a);
int i2d_TYPEPublicKey_bio(BIO *bp, TYPE *a);
int i2d_TYPEparams(const TYPE *a, unsigned char **ppout);
int i2d_TYPEparams(TYPE *a, unsigned char **ppout);
int i2d_TYPEparams_fp(FILE *fp, const TYPE *a);
int i2d_TYPEparams_fp(FILE *fp, TYPE *a);
int i2d_TYPEparams_bio(BIO *bp, const TYPE *a);
int i2d_TYPEparams_bio(BIO *bp, TYPE *a);
int i2d_TYPE_PUBKEY(const TYPE *a, unsigned char **ppout);
int i2d_TYPE_PUBKEY(TYPE *a, unsigned char **ppout);
int i2d_TYPE_PUBKEY_fp(FILE *fp, const TYPE *a);
int i2d_TYPE_PUBKEY_fp(FILE *fp, TYPE *a);
int i2d_TYPE_PUBKEY_bio(BIO *bp, const TYPE *a);
int i2d_TYPE_PUBKEY_bio(BIO *bp, TYPE *a);
```

DESCRIPTION

All functions described here are deprecated. Please use **OSSL_DECODER**(3) instead of the **d2i** functions and **OSSL_ENCODER**(3) instead of the **i2d** functions. See "Migration" below.

In the description here, ***TYPE*** is used a placeholder for any of the OpenSSL datatypes, such as **RSA**. The function parameters *ppin* and *ppout* are generally either both named *pp* in the headers, or *in* and *out*.

All the functions here behave the way that's described in **d2i_X509**(3).

Please note that not all functions in the synopsis are available for all key types. For example, there are no **d2i_RSAparams()** or **i2d_RSAparams()**, because the PKCS#1 **RSA** structure doesn't include any key parameters.

d2i_*TYPE*PrivateKey() and derivates thereof decode DER encoded ***TYPE*** private key data organized in a type specific structure.

d2i_*TYPE*PublicKey() and derivates thereof decode DER encoded ***TYPE*** public key data organized in a type specific structure.

d2i_*TYPE*params() and derivates thereof decode DER encoded ***TYPE*** key parameters organized in a type specific structure.

d2i_*TYPE*_PUBKEY() and derivates thereof decode DER encoded ***TYPE*** public key data organized in a **SubjectPublicKeyInfo** structure.

i2d_*TYPE*PrivateKey() and derivates thereof encode the private key ***TYPE*** data into a type specific DER encoded structure.

i2d_*TYPE*PublicKey() and derivates thereof encode the public key ***TYPE*** data into a type specific DER encoded structure.

i2d_*TYPE*params() and derivates thereof encode the ***TYPE*** key parameters data into a type specific DER encoded structure.

i2d_*TYPE*_PUBKEY() and derivates thereof encode the public key ***TYPE*** data into a DER encoded **SubjectPublicKeyInfo** structure.

For example, **d2i_RSAPrivateKey()** and **d2i_RSAPublicKey()** expects the structure defined by PKCS#1. Similarly, **i2d_RSAPrivateKey()** and **i2d_RSAPublicKey()** produce DER encoded string organized according to PKCS#1.

Migration

Migration from the diverse ***TYPE***s requires using corresponding new OpenSSL types. For all ***TYPE***s described here, the corresponding new type is **EVP_PKEY**. The rest of this section assumes that this has been done, exactly how to do that is described elsewhere.

There are two migration paths:

- Replace b<d2i_*TYPE***PrivateKey()**> with **d2i_PrivateKey**(3), b<d2i_*TYPE***PublicKey()**> with **d2i_PublicKey**(3), b<d2i_*TYPE***params()**> with **d2i_KeyParams**(3), b<d2i_*TYPE*_**PUBKEY()**> with **d2i_PUBKEY**(3), b<i2d_*TYPE***PrivateKey()**> with **i2d_PrivateKey**(3), b<i2d_*TYPE***PublicKey()**> with **i2d_PublicKey**(3), b<i2d_*TYPE***params()**> with **i2d_KeyParams**(3), b<i2d_*TYPE*_**PUBKEY()**> with **i2d_PUBKEY**(3). A caveat is that **i2d_PrivateKey**(3) may output a DER encoded PKCS#8 outermost structure instead of the type specific structure, and that **d2i_PrivateKey**(3) recognises and unpacks a PKCS#8 structures.

- Use **OSSL_DECODER**(3) and **OSSL_ENCODER**(3). How to migrate is described below. All those descriptions assume that the key to be encoded is in the variable *pkey*.

*Migrating **i2d** functions to **OSSL_ENCODER***

The exact **OSSL_ENCODER**(3) output is driven by arguments rather than by function names. The

sample code to get DER encoded output in a type specific structure is uniform, the only things that vary are the selection of what part of the **EVP_PKEY** should be output, and the structure. The **i2d** functions names can therefore be translated into two variables, *selection* and *structure* as follows:

i2d_*TYPE*PrivateKey() translates into:

```
int selection = EVP_PKEY_KEYPAIR;
const char *structure = "type-specific";
```

i2d_*TYPE*PublicKey() translates into:

```
int selection = EVP_PKEY_PUBLIC_KEY;
const char *structure = "type-specific";
```

i2d_*TYPE*params() translates into:

```
int selection = EVP_PKEY_PARAMETERS;
const char *structure = "type-specific";
```

i2d_*TYPE*_PUBKEY() translates into:

```
int selection = EVP_PKEY_PUBLIC_KEY;
const char *structure = "SubjectPublicKeyInfo";
```

The following sample code does the rest of the work:

```
unsigned char *p = buffer;     /* |buffer| is supplied by the caller */
size_t len = buffer_size;      /* assumed be the size of |buffer| */
OSSL_ENCODER_CTX *ctx =
   OSSL_ENCODER_CTX_new_for_pkey(pkey, selection, "DER", structure,
                                 NULL, NULL);
if (ctx == NULL) {
   /* fatal error handling */
}
if (OSSL_ENCODER_CTX_get_num_encoders(ctx) == 0) {
   OSSL_ENCODER_CTX_free(ctx);
   /* non-fatal error handling */
}
if (!OSSL_ENCODER_to_data(ctx, &p, &len)) {
   OSSL_ENCODER_CTX_free(ctx);
   /* error handling */
}
OSSL_ENCODER_CTX_free(ctx);
```

NOTES

The letters **i** and **d** in **i2d_*TYPE***() stand for "internal" (that is, an internal C structure) and "DER" respectively. So **i2d_*TYPE***() converts from internal to DER.

The functions can also understand **BER** forms.

The actual TYPE structure passed to **i2d_*TYPE***() must be a valid populated ***TYPE*** structure -- it **cannot** simply be fed with an empty structure such as that returned by **TYPE_new()**.

The encoded data is in binary form and may contain embedded zeros. Therefore, any FILE pointers or BIOs should be opened in binary mode. Functions such as **strlen()** will **not** return the correct length of the encoded structure.

The ways that **ppin* and **ppout* are incremented after the operation can trap the unwary. See the **WARNINGS** section in **d2i_X509**(3) for some common errors. The reason for this-auto increment behaviour is to reflect a typical usage of ASN1 functions: after one structure is encoded or decoded another will be processed after it.

The following points about the data types might be useful:

DSA_PUBKEY

Represents a DSA public key using a **SubjectPublicKeyInfo** structure.

DSAPublicKey, **DSAPrivateKey**

Use a non-standard OpenSSL format and should be avoided; use **DSA_PUBKEY**, **PEM_write_PrivateKey**(3), or similar instead.

RETURN VALUES

d2i_*TYPE*(), **d2i_*TYPE*_bio**() and **d2i_*TYPE*_fp**() return a valid ***TYPE*** structure or NULL if an error occurs. If the "reuse" capability has been used with a valid structure being passed in via *a*, then the object is freed in the event of error and **a* is set to NULL.

i2d_*TYPE*() returns the number of bytes successfully encoded or a negative value if an error occurs.

i2d_*TYPE*_bio() and **i2d_*TYPE*_fp**() return 1 for success and 0 if an error occurs.

SEE ALSO

OSSL_ENCODER(3), **OSSL_DECODER**(3), **d2i_PrivateKey**(3), **d2i_PublicKey**(3), **d2i_KeyParams**(3), **d2i_PUBKEY**(3), **i2d_PrivateKey**(3), **i2d_PublicKey**(3), **i2d_KeyParams**(3), **i2d_PUBKEY**(3)

COPYRIGHT