



## ***Red Hat Enterprise Linux Release 9.2 Manual Pages on 'i2d\_RSA\_PUBKEY\_bio.3ossl' command***

***\$ man i2d\_RSA\_PUBKEY\_bio.3ossl***

D2I\_RSAPRIVATEKEY(3ossl)      OpenSSL      D2I\_RSAPRIVATEKEY(3ossl)

### NAME

d2i\_DSAPrivateKey, d2i\_DSAPrivateKey\_bio, d2i\_DSAPrivateKey\_fp,  
d2i\_DSAPublicKey, d2i\_DSA\_PUBKEY, d2i\_DSA\_PUBKEY\_bio,  
d2i\_DSA\_PUBKEY\_fp, d2i\_DSAPrivateKey, 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\_ECPKParameters, d2i\_ECPParameters,  
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\_DSAPrivateKey,  
i2d\_ECPKParameters, i2d\_ECPParameters, i2d\_ECPrivateKey,  
i2d\_ECPrivateKey\_bio, i2d\_ECPrivateKey\_fp, i2d\_EC\_PUBKEY,  
i2d\_EC\_PUBKEY\_bio, i2d\_EC\_PUBKEY\_fp - DEPRECATED

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_TYPEPrivateKey()` and derivatives thereof decode DER encoded `TYPE` private key data organized in a type specific structure.

`d2i_TYPEPublicKey()` and derivatives thereof decode DER encoded `TYPE` public key data organized in a type specific structure.

`d2i_TYPEparams()` and derivatives thereof decode DER encoded `TYPE` key parameters organized in a type specific structure.

d2i\_TYPE\_PUBKEY() and derivatives thereof decode DER encoded TYPE public key data organized in a SubjectPublicKeyInfo structure.

i2d\_TYPEPrivateKey() and derivatives thereof encode the private key TYPE data into a type specific DER encoded structure.

i2d\_TYPEPublicKey() and derivatives thereof encode the public key TYPE data into a type specific DER encoded structure.

i2d\_TYPEparams() and derivatives thereof encode the TYPE key parameters data into a type specific DER encoded structure.

i2d\_TYPE\_PUBKEY() and derivatives 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 TYPEs requires using corresponding new OpenSSL types. For all TYPEs 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\_TYPEPrivateKey()> with d2i\_PrivateKey(3),  
b<d2i\_TYPEPublicKey()> with d2i\_PublicKey(3), b<d2i\_TYPEparams()>  
with d2i\_KeyParams(3), b<d2i\_TYPE\_PUBKEY()> with d2i\_PUBKEY(3),  
b<i2d\_TYPEPrivateKey()> with i2d\_PrivateKey(3),

b<i2d\_TYPEPublicKey()> with i2d\_PublicKey(3), b<i2d\_TYPEparams()> 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\_TYPEPrivateKey() translates into:

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

i2d\_TYPEPublicKey() translates into:

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

i2d\_TYPEparams() 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.

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

Copyright 2020-2021 The OpenSSL Project Authors. All Rights Reserved.

Licensed under the Apache License 2.0 (the "License"). You may not use  
this file except in compliance with the License. You can obtain a copy  
in the file LICENSE in the source distribution or at  
<<https://www.openssl.org/source/license.html>>.

3.0.7                    2023-07-13            D2I\_RSAPRIVATEKEY(3ossl)