

**NAME**

EVP\_PKEY\_set1\_encoded\_public\_key, EVP\_PKEY\_get1\_encoded\_public\_key, EVP\_PKEY\_set1\_tls\_encodedpoint, EVP\_PKEY\_get1\_tls\_encodedpoint - functions to set and get public key data within an EVP\_PKEY

**SYNOPSIS**

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

```
int EVP_PKEY_set1_encoded_public_key(EVP_PKEY *pkey,
                                     const unsigned char *pub, size_t publen);
```

```
size_t EVP_PKEY_get1_encoded_public_key(EVP_PKEY *pkey, unsigned char **ppub);
```

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)**:

```
int EVP_PKEY_set1_tls_encodedpoint(EVP_PKEY *pkey,
                                    const unsigned char *pt, size_t ptlen);
```

```
size_t EVP_PKEY_get1_tls_encodedpoint(EVP_PKEY *pkey, unsigned char **ppt);
```

**DESCRIPTION**

**EVP\_PKEY\_set1\_encoded\_public\_key()** can be used to set the public key value within an existing EVP\_PKEY object. For the built-in OpenSSL algorithms this currently only works for those that support key exchange. Parameters are not set as part of this operation, so typically an application will create an EVP\_PKEY first, set the parameters on it, and then call this function. For example setting the parameters might be done using **EVP\_PKEY\_copy\_parameters(3)**.

The format for the encoded public key will depend on the algorithm in use. For DH it should be encoded as a positive integer in big-endian form. For EC it should be a point conforming to Sec. 2.3.4 of the SECG SEC 1 ("Elliptic Curve Cryptography") standard. For X25519 and X448 it should be encoded in a format as defined by RFC7748.

The key to be updated is supplied in **pkey**. The buffer containing the encoded key is pointed to be **pub**. The length of the buffer is supplied in **publen**.

**EVP\_PKEY\_get1\_encoded\_public\_key()** does the equivalent operation except that the encoded public key is returned to the application. The key containing the public key data is supplied in **pkey**. A buffer containing the encoded key will be allocated and stored in **\*ppub**. The length of the encoded public key is returned by the function. The application is responsible for freeing the allocated buffer.

The macro **EVP\_PKEY\_set1\_tls\_encodedpoint()** is deprecated and simply calls **EVP\_PKEY\_set1\_encoded\_public\_key()** with all the same arguments. New applications should use **EVP\_PKEY\_set1\_encoded\_public\_key()** instead.

The macro **EVP\_PKEY\_get1\_tls\_encodedpoint()** is deprecated and simply calls **EVP\_PKEY\_get1\_encoded\_public\_key()** with all the same arguments. New applications should use **EVP\_PKEY\_get1\_encoded\_public\_key()** instead.

## RETURN VALUES

**EVP\_PKEY\_set1\_encoded\_public\_key()** returns 1 for success and 0 or a negative value for failure.

**EVP\_PKEY\_get1\_encoded\_public\_key()** returns the length of the encoded key or 0 for failure.

## EXAMPLES

See **EVP\_PKEY\_derive\_init(3)** and **EVP\_PKEY\_derive(3)** for information about performing a key exchange operation.

### Set up a peer's EVP\_PKEY ready for a key exchange operation

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

int exchange(EVP_PKEY *ourkey, unsigned char *peer_pub, size_t peer_pub_len)
{
    EVP_PKEY *peerkey = EVP_PKEY_new();

    if (peerkey == NULL || EVP_PKEY_copy_parameters(peerkey, ourkey) <= 0)
        return 0;

    if (EVP_PKEY_set1_encoded_public_key(peerkey, peer_pub,
                                         peer_pub_len) <= 0)
        return 0;

    /* Do the key exchange here */

    EVP_PKEY_free(peerkey);

    return 1;
}
```

### Get an encoded public key to send to a peer

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

```
int get_encoded_pub_key(EVP_PKEY *ourkey)
{
    unsigned char *pubkey;
    size_t pubkey_len;

    pubkey_len = EVP_PKEY_get1_encoded_public_key(ourkey, &pubkey);
    if (pubkey_len == 0)
        return 0;

    /*
     * Send the encoded public key stored in the buffer at "pubkey" and of
     * length pubkey_len, to the peer.
     */

    OPENSSL_free(pubkey);
    return 1;
}
```

## SEE ALSO

**EVP\_PKEY\_new(3)**, **EVP\_PKEY\_copy\_parameters(3)**, **EVP\_PKEY\_derive\_init(3)**,  
**EVP\_PKEY\_derive(3)**, **EVP\_PKEY-DH(7)**, **EVP\_PKEY-EC(7)**, **EVP\_PKEY-X25519(7)**,  
**EVP\_PKEY-X448(7)**

## HISTORY

**EVP\_PKEY\_set1\_encoded\_public\_key()** and **EVP\_PKEY\_get1\_encoded\_public\_key()** were added in OpenSSL 3.0.

**EVP\_PKEY\_set1\_tls\_encodedpoint()** and **EVP\_PKEY\_get1\_tls\_encodedpoint()** were deprecated in OpenSSL 3.0.

## 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>>.