NAME

EVP_PKEY, EVP_PKEY_new, EVP_PKEY_up_ref, EVP_PKEY_dup, EVP_PKEY_free, EVP_PKEY_new_raw_private_key_ex, EVP_PKEY_new_raw_private_key, EVP_PKEY_new_raw_public_key_ex, EVP_PKEY_new_raw_public_key, EVP_PKEY_new_CMAC_key, EVP_PKEY_new_mac_key, EVP_PKEY_get_raw_private_key, EVP_PKEY_get_raw_public_key - public/private key allocation and raw key handling functions

SYNOPSIS

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

typedef evp_pkey_st EVP_PKEY;

EVP_PKEY *EVP_PKEY_new(void);
int EVP_PKEY_up_ref(EVP_PKEY *key);
EVP_PKEY *EVP_PKEY_dup(EVP_PKEY *key);
void EVP_PKEY_free(EVP_PKEY *key);

EVP_PKEY *EVP_PKEY_new_raw_private_key_ex(OSSL_LIB_CTX *libctx,
                                          const char *keytype,
                                          const char *propq,
                                          const unsigned char *key,
                                          size_t keylen);
EVP_PKEY *EVP_PKEY_new_raw_private_key(int type, ENGINE *e,
                                       const unsigned char *key, size_t keylen);
EVP_PKEY *EVP_PKEY_new_raw_public_key_ex(OSSL_LIB_CTX *libctx,
                                         const char *keytype,
                                         const char *propq,
                                         const unsigned char *key,
                                         size_t keylen);
EVP_PKEY *EVP_PKEY_new_raw_public_key(int type, ENGINE *e,
                                      const unsigned char *key, size_t keylen);
EVP_PKEY *EVP_PKEY_new_mac_key(int type, ENGINE *e, const unsigned char *key,
                               int keylen);

int EVP_PKEY_get_raw_private_key(const EVP_PKEY *pkey, unsigned char *priv,
                                 size_t *len);
int EVP_PKEY_get_raw_public_key(const EVP_PKEY *pkey, unsigned char *pub,
                                size_t *len);
```

The following function has 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):

```
EVP_PKEY *EVP_PKEY_new_CMAC_key(ENGINE *e, const unsigned char *priv,
                                size_t len, const EVP_CIPHER *cipher);
```

DESCRIPTION

EVP_PKEY is a generic structure to hold diverse types of asymmetric keys (also known as "key pairs"), and can be used for diverse operations, like signing, verifying signatures, key derivation, etc. The asymmetric keys themselves are often referred to as the "internal key", and are handled by backends, such as providers (through **EVP_KEYMGMT**(3)) or **ENGINE**s.

Conceptually, an **EVP_PKEY** internal key may hold a private key, a public key, or both (a keypair), and along with those, key parameters if the key type requires them. The presence of these components determine what operations can be made; for example, signing normally requires the presence of a private key, and verifying normally requires the presence of a public key.

EVP_PKEY has also been used for MAC algorithm that were conceived as producing signatures, although not being public key algorithms; "POLY1305", "SIPHASH", "HMAC", "CMAC". This usage is considered legacy and is discouraged in favor of the **EVP_MAC**(3) API.

The **EVP_PKEY_new()** function allocates an empty **EVP_PKEY** structure which is used by OpenSSL to store public and private keys. The reference count is set to **1**.

EVP_PKEY_up_ref() increments the reference count of *key*.

EVP_PKEY_dup() duplicates the *key*. The *key* must not be ENGINE based or a raw key, otherwise the duplication will fail.

EVP_PKEY_free() decrements the reference count of *key* and, if the reference count is zero, frees it up. If *key* is NULL, nothing is done.

EVP_PKEY_new_raw_private_key_ex() allocates a new **EVP_PKEY**. Unless an engine should be used for the key type, a provider for the key is found using the library context *libctx* and the property query string *propq*. The *keytype* argument indicates what kind of key this is. The value should be a string for a public key algorithm that supports raw private keys, i.e one of "X25519", "ED25519", "X448" or "ED448". *key* points to the raw private key data for this **EVP_PKEY** which should be of length *keylen*. The length should be appropriate for the type of the key. The public key data will be automatically derived from the given private key data (if appropriate for the algorithm type).

EVP_PKEY_new_raw_private_key() does the same as **EVP_PKEY_new_raw_private_key_ex()**

except that the default library context and default property query are used instead. If *e* is non-NULL then the new **EVP_PKEY** structure is associated with the engine *e*. The *type* argument indicates what kind of key this is. The value should be a NID for a public key algorithm that supports raw private keys, i.e. one of **EVP_PKEY_X25519**, **EVP_PKEY_ED25519**, **EVP_PKEY_X448** or **EVP_PKEY_ED448**.

EVP_PKEY_new_raw_private_key_ex() and **EVP_PKEY_new_raw_private_key()** may also be used with most MACs implemented as public key algorithms, so key types such as "HMAC", "POLY1305", "SIPHASH", or their NID form **EVP_PKEY_POLY1305**, **EVP_PKEY_SIPHASH**, **EVP_PKEY_HMAC** are also accepted. This usage is, as mentioned above, discouraged in favor of the **EVP_MAC**(3) API.

EVP_PKEY_new_raw_public_key_ex() works in the same way as **EVP_PKEY_new_raw_private_key_ex()** except that *key* points to the raw public key data. The **EVP_PKEY** structure will be initialised without any private key information. Algorithm types that support raw public keys are "X25519", "ED25519", "X448" or "ED448".

EVP_PKEY_new_raw_public_key() works in the same way as **EVP_PKEY_new_raw_private_key()** except that *key* points to the raw public key data. The **EVP_PKEY** structure will be initialised without any private key information. Algorithm types that support raw public keys are **EVP_PKEY_X25519**, **EVP_PKEY_ED25519**, **EVP_PKEY_X448** or **EVP_PKEY_ED448**.

EVP_PKEY_new_mac_key() works in the same way as **EVP_PKEY_new_raw_private_key()**. New applications should use **EVP_PKEY_new_raw_private_key()** instead.

EVP_PKEY_get_raw_private_key() fills the buffer provided by *priv* with raw private key data. The size of the *priv* buffer should be in **len* on entry to the function, and on exit **len* is updated with the number of bytes actually written. If the buffer *priv* is NULL then **len* is populated with the number of bytes required to hold the key. The calling application is responsible for ensuring that the buffer is large enough to receive the private key data. This function only works for algorithms that support raw private keys. Currently this is: **EVP_PKEY_HMAC**, **EVP_PKEY_POLY1305**, **EVP_PKEY_SIPHASH**, **EVP_PKEY_X25519**, **EVP_PKEY_ED25519**, **EVP_PKEY_X448** or **EVP_PKEY_ED448**.

EVP_PKEY_get_raw_public_key() fills the buffer provided by *pub* with raw public key data. The size of the *pub* buffer should be in **len* on entry to the function, and on exit **len* is updated with the number of bytes actually written. If the buffer *pub* is NULL then **len* is populated with the number of bytes required to hold the key. The calling application is responsible for ensuring that the buffer is large enough to receive the public key data. This function only works for algorithms that support raw public keys. Currently this is: **EVP_PKEY_X25519**, **EVP_PKEY_ED25519**, **EVP_PKEY_X448** or

EVP_PKEY_ED448.

EVP_PKEY_new_CMAC_key() works in the same way as **EVP_PKEY_new_raw_private_key()** except it is only for the **EVP_PKEY_CMAC** algorithm type. In addition to the raw private key data, it also takes a cipher algorithm to be used during creation of a CMAC in the **cipher** argument. The cipher should be a standard encryption-only cipher. For example AEAD and XTS ciphers should not be used.

Applications should use the **EVP_MAC**(3) API instead and set the **OSSL_MAC_PARAM_CIPHER** parameter on the **EVP_MAC_CTX** object with the name of the cipher being used.

NOTES

The **EVP_PKEY** structure is used by various OpenSSL functions which require a general private key without reference to any particular algorithm.

The structure returned by **EVP_PKEY_new()** is empty. To add a private or public key to this empty structure use the appropriate functions described in **EVP_PKEY_set1_RSA**(3), **EVP_PKEY_set1_DSA**(3), **EVP_PKEY_set1_DH**(3) or **EVP_PKEY_set1_EC_KEY**(3).

RETURN VALUES

EVP_PKEY_new(), **EVP_PKEY_new_raw_private_key()**, **EVP_PKEY_new_raw_public_key()**, **EVP_PKEY_new_CMAC_key()** and **EVP_PKEY_new_mac_key()** return either the newly allocated **EVP_PKEY** structure or NULL if an error occurred.

EVP_PKEY_dup() returns the key duplicate or NULL if an error occurred.

EVP_PKEY_up_ref(), **EVP_PKEY_get_raw_private_key()** and **EVP_PKEY_get_raw_public_key()** return 1 for success and 0 for failure.

SEE ALSO

EVP_PKEY_set1_RSA(3), **EVP_PKEY_set1_DSA**(3), **EVP_PKEY_set1_DH**(3) or **EVP_PKEY_set1_EC_KEY**(3)

HISTORY

The **EVP_PKEY_new()** and **EVP_PKEY_free()** functions exist in all versions of OpenSSL.

The **EVP_PKEY_up_ref()** function was added in OpenSSL 1.1.0.

The **EVP_PKEY_new_raw_private_key()**, **EVP_PKEY_new_raw_public_key()**, **EVP_PKEY_new_CMAC_key()**, **EVP_PKEY_new_raw_private_key()** and **EVP_PKEY_get_raw_public_key()** functions were added in OpenSSL 1.1.1.

The **EVP_PKEY_dup()**, **EVP_PKEY_new_raw_private_key_ex()**, and **EVP_PKEY_new_raw_public_key_ex()** functions were added in OpenSSL 3.0.

The **EVP_PKEY_new_CMAC_key()** was deprecated in OpenSSL 3.0.

The documentation of **EVP_PKEY** was amended in OpenSSL 3.0 to allow there to be the private part of the keypair without the public part, where this was previously implied to be disallowed.

COPYRIGHT

Copyright 2002-2023 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>.