NAME

EVP_EC_gen, EC_KEY_get_method, EC_KEY_set_method, EC_KEY_new_ex, EC_KEY_new, EC_KEY_get_flags, EC_KEY_set_flags, EC_KEY_clear_flags, EC_KEY_new_by_curve_name_ex, EC_KEY_new_by_curve_name, EC_KEY_free, EC_KEY_copy, EC_KEY_dup, EC_KEY_up_ref, EC_KEY_get0_engine, EC_KEY_get0_group, EC_KEY_set_group, EC_KEY_get0_private_key, EC_KEY_set_private_key, EC_KEY_get0_public_key, EC_KEY_set_public_key, EC_KEY_get_conv_form, EC_KEY_set_conv_form, EC_KEY_set_asn1_flag, EC_KEY_decoded_from_explicit_params, EC_KEY_precompute_mult, EC_KEY_generate_key, EC_KEY_check_key, EC_KEY_set_public_key_affine_coordinates, EC_KEY_oct2key, EC_KEY_key2buf, EC_KEY_oct2priv, EC_KEY_priv2oct, EC_KEY_priv2buf - Functions for creating, destroying and manipulating EC_KEY objects

SYNOPSIS

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

EVP_PKEY *EVP_EC_gen(const char *curve);
```

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):

```
EC_KEY *EC_KEY_new_ex(OSSL_LIB_CTX *ctx, const char *propq);
EC_KEY *EC_KEY_new(void);
int EC_KEY_get_flags(const EC_KEY *key);
void EC_KEY_set_flags(EC_KEY *key, int flags);
void EC_KEY_clear_flags(EC_KEY *key, int flags);
EC_KEY *EC_KEY_new_by_curve_name_ex(OSSL_LIB_CTX *ctx, const char *propq,
                                    int nid);
EC_KEY *EC_KEY_new_by_curve_name(int nid);
void EC_KEY_free(EC_KEY *key);
EC_KEY *EC_KEY_copy(EC_KEY *dst, const EC_KEY *src);
EC_KEY *EC_KEY_dup(const EC_KEY *src);
int EC_KEY_up_ref(EC_KEY *key);
ENGINE *EC_KEY_get0_engine(const EC_KEY *eckey);
const EC_GROUP *EC_KEY_get0_group(const EC_KEY *key);
int EC_KEY_set_group(EC_KEY *key, const EC_GROUP *group);
const BIGNUM *EC_KEY_get0_private_key(const EC_KEY *key);
int EC_KEY_set_private_key(EC_KEY *key, const BIGNUM *priv_key);
const EC_POINT *EC_KEY_get0_public_key(const EC_KEY *key);
int EC_KEY_set_public_key(EC_KEY *key, const EC_POINT *pub);
point_conversion_form_t EC_KEY_get_conv_form(const EC_KEY *key);
```

```
void EC_KEY_set_conv_form(EC_KEY *eckey, point_conversion_form_t cform);
void EC_KEY_set_asn1_flag(EC_KEY *eckey, int asn1_flag);
int EC_KEY_decoded_from_explicit_params(const EC_KEY *key);
int EC_KEY_generate_key(EC_KEY *key);
int EC_KEY_check_key(const EC_KEY *key);
int EC_KEY_set_public_key_affine_coordinates(EC_KEY *key, BIGNUM *x, BIGNUM *y);
const EC_KEY_METHOD *EC_KEY_get_method(const EC_KEY *key);
int EC_KEY_set_method(EC_KEY *key, const EC_KEY_METHOD *meth);

int EC_KEY_oct2key(EC_KEY *eckey, const unsigned char *buf, size_t len, BN_CTX *ctx);
size_t EC_KEY_key2buf(const EC_KEY *eckey, point_conversion_form_t form,
                      unsigned char **pbuf, BN_CTX *ctx);

int EC_KEY_oct2priv(EC_KEY *eckey, const unsigned char *buf, size_t len);
size_t EC_KEY_priv2oct(const EC_KEY *eckey, unsigned char *buf, size_t len);

size_t EC_KEY_priv2buf(const EC_KEY *eckey, unsigned char **pbuf);
int EC_KEY_precompute_mult(EC_KEY *key, BN_CTX *ctx);
```

DESCRIPTION

EVP_EC_gen() generates a new EC key pair on the given *curve*.

All of the functions described below are deprecated. Applications should instead use **EVP_EC_gen()**, **EVP_PKEY_Q_keygen**(3), or **EVP_PKEY_keygen_init**(3) and **EVP_PKEY_keygen**(3).

An EC_KEY represents a public key and, optionally, the associated private key. A new EC_KEY with no associated curve can be constructed by calling **EC_KEY_new_ex()** and specifying the associated library context in *ctx* (see **OSSL_LIB_CTX**(3)) and property query string *propq*. The *ctx* parameter may be NULL in which case the default library context is used. The reference count for the newly created EC_KEY is initially set to 1. A curve can be associated with the EC_KEY by calling **EC_KEY_set_group()**.

EC_KEY_new() is the same as **EC_KEY_new_ex()** except that the default library context is always used.

Alternatively a new EC_KEY can be constructed by calling **EC_KEY_new_by_curve_name_ex()** and supplying the nid of the associated curve, the library context to be used *ctx* (see **OSSL_LIB_CTX**(3)) and any property query string *propq*. The *ctx* parameter may be NULL in which case the default library context is used. The *propq* value may also be NULL. See **EC_GROUP_new**(3) for a description of curve names. This function simply wraps calls to **EC_KEY_new_ex()** and

EC_GROUP_new_by_curve_name_ex().

EC_KEY_new_by_curve_name() is the same as **EC_KEY_new_by_curve_name_ex()** except that the default library context is always used and a NULL property query string.

Calling **EC_KEY_free()** decrements the reference count for the EC_KEY object, and if it has dropped to zero then frees the memory associated with it. If *key* is NULL nothing is done.

EC_KEY_copy() copies the contents of the EC_KEY in *src* into *dest*.

EC_KEY_dup() creates a new EC_KEY object and copies *ec_key* into it.

EC_KEY_up_ref() increments the reference count associated with the EC_KEY object.

EC_KEY_get0_engine() returns a handle to the ENGINE that has been set for this EC_KEY object.

EC_KEY_generate_key() generates a new public and private key for the supplied *eckey* object. *eckey* must have an EC_GROUP object associated with it before calling this function. The private key is a random integer (0 < priv_key < order, where *order* is the order of the EC_GROUP object). The public key is an EC_POINT on the curve calculated by multiplying the generator for the curve by the private key.

EC_KEY_check_key() performs various sanity checks on the EC_KEY object to confirm that it is valid.

EC_KEY_set_public_key_affine_coordinates() sets the public key for *key* based on its affine coordinates; i.e., it constructs an EC_POINT object based on the supplied *x* and *y* values and sets the public key to be this EC_POINT. It also performs certain sanity checks on the key to confirm that it is valid.

The functions **EC_KEY_get0_group()**, **EC_KEY_set_group()**, **EC_KEY_get0_private_key()**, **EC_KEY_set_private_key()**, **EC_KEY_get0_public_key()**, and **EC_KEY_set_public_key()** get and set the EC_GROUP object, the private key, and the EC_POINT public key for the **key** respectively. The function **EC_KEY_set_private_key()** accepts NULL as the priv_key argument to securely clear the private key component from the EC_KEY.

The functions **EC_KEY_get_conv_form()** and **EC_KEY_set_conv_form()** get and set the point_conversion_form for the *key*. For a description of point_conversion_forms please see **EC_POINT_new**(3).

EC_KEY_set_flags() sets the flags in the *flags* parameter on the EC_KEY object. Any flags that are already set are left set. The flags currently defined are EC_FLAG_NON_FIPS_ALLOW and EC_FLAG_FIPS_CHECKED. In addition there is the flag EC_FLAG_COFACTOR_ECDH which is specific to ECDH. **EC_KEY_get_flags()** returns the current flags that are set for this EC_KEY. **EC_KEY_clear_flags()** clears the flags indicated by the *flags* parameter; all other flags are left in their existing state.

EC_KEY_set_asn1_flag() sets the asn1_flag on the underlying EC_GROUP object (if set). Refer to **EC_GROUP_copy**(3) for further information on the asn1_flag.

EC_KEY_decoded_from_explicit_params() returns 1 if the group of the *key* was decoded from data with explicitly encoded group parameters, -1 if the *key* is NULL or the group parameters are missing, and 0 otherwise.

EC_KEY_precompute_mult() stores multiples of the underlying EC_GROUP generator for faster point multiplication. See also **EC_POINT_add**(3). Modern versions should instead switch to named curves which OpenSSL has hardcoded lookup tables for.

EC_KEY_oct2key() and **EC_KEY_key2buf()** are identical to the functions **EC_POINT_oct2point()** and **EC_POINT_point2buf()** except they use the public key EC_POINT in *eckey*.

EC_KEY_oct2priv() and **EC_KEY_priv2oct()** convert between the private key component of *eckey* and octet form. The octet form consists of the content octets of the *privateKey* OCTET STRING in an *ECPrivateKey* ASN.1 structure.

The function **EC_KEY_priv2oct()** must be supplied with a buffer long enough to store the octet form. The return value provides the number of octets stored. Calling the function with a NULL buffer will not perform the conversion but will just return the required buffer length.

The function **EC_KEY_priv2buf()** allocates a buffer of suitable length and writes an EC_KEY to it in octet format. The allocated buffer is written to **pbuf* and its length is returned. The caller must free up the allocated buffer with a call to **OPENSSL_free()**. Since the allocated buffer value is written to **pbuf* the *pbuf* parameter **MUST NOT** be **NULL**.

EC_KEY_priv2buf() converts an EC_KEY private key into an allocated buffer.

RETURN VALUES

EC_KEY_new_ex(), **EC_KEY_new()**, **EC_KEY_new_by_curve_name_ex()**, **EC_KEY_new_by_curve_name()** and **EC_KEY_dup()** return a pointer to the newly created EC_KEY object, or NULL on error.

EC_KEY_get_flags() returns the flags associated with the EC_KEY object as an integer.

EC_KEY_copy() returns a pointer to the destination key, or NULL on error.

EC_KEY_get0_engine() returns a pointer to an ENGINE, or NULL if it wasn't set.

EC_KEY_up_ref(), **EC_KEY_set_group()**, **EC_KEY_set_public_key()**, **EC_KEY_precompute_mult()**, **EC_KEY_generate_key()**, **EC_KEY_check_key()**, **EC_KEY_set_public_key_affine_coordinates()**, **EC_KEY_oct2key()** and **EC_KEY_oct2priv()** return 1 on success or 0 on error.

EC_KEY_set_private_key() returns 1 on success or 0 on error except when the priv_key argument is NULL, in that case it returns 0, for legacy compatibility, and should not be treated as an error.

EC_KEY_get0_group() returns the EC_GROUP associated with the EC_KEY.

EC_KEY_get0_private_key() returns the private key associated with the EC_KEY.

EC_KEY_get_conv_form() return the point_conversion_form for the EC_KEY.

EC_KEY_key2buf(), **EC_KEY_priv2oct()** and **EC_KEY_priv2buf()** return the length of the buffer or 0 on error.

SEE ALSO

EVP_PKEY_Q_keygen(3) **crypto**(7), **EC_GROUP_new**(3), **EC_GROUP_copy**(3), **EC_POINT_new**(3), **EC_POINT_add**(3), **EC_GFp_simple_method**(3), **d2i_ECPKParameters**(3), **OSSL_LIB_CTX**(3)

HISTORY

EVP_EC_gen() was added in OpenSSL 3.0. All other functions described here were deprecated in OpenSSL 3.0. For replacement see **EVP_PKEY-EC**(7).

COPYRIGHT

Copyright 2013-2023 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>.