NAME

EC_GROUP_get_ecparameters, EC_GROUP_get_ecpkparameters, EC_GROUP_new_from_params, EC_GROUP_to_params, EC_GROUP_new_from_ecparameters, EC_GROUP_new_from_ecpkparameters, EC_GROUP_new, EC_GROUP_free, EC_GROUP_clear_free, EC_GROUP_new_curve_GFp, EC_GROUP_new_curve_GF2m, EC_GROUP_new_by_curve_name_ex, EC_GROUP_new_by_curve_name, EC_GROUP_set_curve, EC_GROUP_get_curve, EC_GROUP_set_curve_GFp, EC_GROUP_get_curve_GFp, EC_GROUP_set_curve_GF2m, EC_GROUP_get_curve_GF2m, EC_get_builtin_curves, OSSL_EC_curve_nid2name − Functions for creating and destroying EC_GROUP objects

SYNOPSIS

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

EC_GROUP *EC_GROUP_new_from_params(const OSSL_PARAM params[],
                                   OSSL_LIB_CTX *libctx, const char *propq);
OSSL_PARAM *EC_GROUP_to_params(const EC_GROUP *group, OSSL_LIB_CTX *libctx,
                               const char *propq, BN_CTX *bnctx);
EC_GROUP *EC_GROUP_new_from_ecparameters(const ECPARAMETERS *params);
EC_GROUP *EC_GROUP_new_from_ecpkparameters(const ECPKPARAMETERS *params);
void EC_GROUP_free(EC_GROUP *group);

EC_GROUP *EC_GROUP_new_curve_GFp(const BIGNUM *p, const BIGNUM *a,
                                 const BIGNUM *b, BN_CTX *ctx);
EC_GROUP *EC_GROUP_new_curve_GF2m(const BIGNUM *p, const BIGNUM *a,
                                  const BIGNUM *b, BN_CTX *ctx);
EC_GROUP *EC_GROUP_new_by_curve_name_ex(OSSL_LIB_CTX *libctx, const char *propq,
                                        int nid);
EC_GROUP *EC_GROUP_new_by_curve_name(int nid);

int EC_GROUP_set_curve(EC_GROUP *group, const BIGNUM *p, const BIGNUM *a,
                       const BIGNUM *b, BN_CTX *ctx);
int EC_GROUP_get_curve(const EC_GROUP *group, BIGNUM *p, BIGNUM *a, BIGNUM *b,
                       BN_CTX *ctx);

ECPARAMETERS *EC_GROUP_get_ecparameters(const EC_GROUP *group,
                                        ECPARAMETERS *params);
ECPKPARAMETERS *EC_GROUP_get_ecpkparameters(const EC_GROUP *group,
                                            ECPKPARAMETERS *params);

size_t EC_get_builtin_curves(EC_builtin_curve *r, size_t nitems);
const char *OSSL_EC_curve_nid2name(int nid);
```

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_GROUP *EC_GROUP_new(const EC_METHOD *meth);
void EC_GROUP_clear_free(EC_GROUP *group);

int EC_GROUP_set_curve_GFp(EC_GROUP *group, const BIGNUM *p,
                           const BIGNUM *a, const BIGNUM *b, BN_CTX *ctx);
int EC_GROUP_get_curve_GFp(const EC_GROUP *group, BIGNUM *p,
                           BIGNUM *a, BIGNUM *b, BN_CTX *ctx);
int EC_GROUP_set_curve_GF2m(EC_GROUP *group, const BIGNUM *p,
                            const BIGNUM *a, const BIGNUM *b, BN_CTX *ctx);
int EC_GROUP_get_curve_GF2m(const EC_GROUP *group, BIGNUM *p,
                            BIGNUM *a, BIGNUM *b, BN_CTX *ctx);
```

DESCRIPTION

Within the library there are two forms of elliptic curve that are of interest. The first form is those defined over the prime field Fp. The elements of Fp are the integers 0 to p−1, where p is a prime number. This gives us a revised elliptic curve equation as follows:

y^2 mod p = x^3 +ax + b mod p

The second form is those defined over a binary field F2^m where the elements of the field are integers of length at most m bits. For this form the elliptic curve equation is modified to:

y^2 + xy = x^3 + ax^2 + b (where b != 0)

Operations in a binary field are performed relative to an **irreducible polynomial**. All such curves with OpenSSL use a trinomial or a pentanomial for this parameter.

Although deprecated since OpenSSL 3.0 and should no longer be used, a new curve can be constructed by calling **EC_GROUP_new()**, using the implementation provided by *meth* (see **EC_GFp_simple_method**(3)) and associated with the library context *ctx* (see **OSSL_LIB_CTX**(3)). The *ctx* parameter may be NULL in which case the default library context is used. It is then necessary to call **EC_GROUP_set_curve()** to set the curve parameters. Applications should instead use one of the other EC_GROUP_new_* constructors.

EC_GROUP_new_from_params() creates a group with parameters specified by *params*. The library context *libctx* (see **OSSL_LIB_CTX**(3)) and property query string *propq* are used to fetch algorithms from providers. *params* may be either a list of explicit params or a named group, The values for *ctx* and *propq* may be NULL. The *params* that can be used are described in **EVP_PKEY−EC**(7).

EC_GROUP_to_params creates an OSSL_PARAM array with the corresponding parameters describing the given EC_GROUP. The resulting parameters may contain parameters describing a named or explicit curve depending on the EC_GROUP. The library context *libctx* (see **OSSL_LIB_CTX**(3)) and property query string *propq* are used to fetch algorithms from providers. *bnctx* is an optional preallocated BN_CTX (to save the overhead of allocating and freeing the structure in a loop). The values for *libctx*, *propq* and *bnctx* may be NULL. The caller is responsible for freeing the OSSL_PARAM pointer returned.

EC_GROUP_new_from_ecparameters() will create a group from the specified *params* and **EC_GROUP_new_from_ecpkparameters()** will create a group from the specific PK *params*.

EC_GROUP_set_curve() sets the curve parameters *p*, *a* and *b*. For a curve over Fp *p* is the prime for the field. For a curve over F2^m *p* represents the irreducible polynomial − each bit represents a term in the polynomial. Therefore, there will either be three or five bits set dependent on whether the polynomial is a trinomial or a pentanomial. In either case, *a* and *b* represents the coefficients a and b from the relevant equation introduced above.

EC_group_get_curve() obtains the previously set curve parameters.

EC_GROUP_set_curve_GFp() and **EC_GROUP_set_curve_GF2m()** are synonyms for **EC_GROUP_set_curve()**. They are defined for backwards compatibility only and should not be used.

EC_GROUP_get_curve_GFp() and **EC_GROUP_get_curve_GF2m()** are synonyms for **EC_GROUP_get_curve()**. They are defined for backwards compatibility only and should not be used.

The functions **EC_GROUP_new_curve_GFp()** and **EC_GROUP_new_curve_GF2m()** are shortcuts for calling **EC_GROUP_new()** and then the **EC_GROUP_set_curve()** function. An appropriate default implementation method will be used.

Whilst the library can be used to create any curve using the functions described above, there are also a number of predefined curves that are available. In order to obtain a list of all of the predefined curves, call the function **EC_get_builtin_curves()**. The parameter *r* should be an array of EC_builtin_curve structures of size *nitems*. The function will populate the *r* array with information about the built−in curves. If *nitems* is less than the total number of curves available, then the first *nitems* curves will be returned. Otherwise the total number of curves will be provided. The return value is the total number of curves available (whether that number has been populated in *r* or not). Passing a NULL *r*, or setting *nitems* to 0 will do nothing other than return the total number of curves available. The EC_builtin_curve structure is defined as follows:

```
typedef struct {
       int nid;
       const char *comment;
       } EC_builtin_curve;
```

Each EC_builtin_curve item has a unique integer id (*nid*), and a human readable comment string describing the curve.

In order to construct a built−in curve use the function **EC_GROUP_new_by_curve_name_ex()** and provide the *nid* of the curve to be constructed, the associated library context to be used in *ctx* (see **OSSL_LIB_CTX**(3)) and any property query string in *propq*. The *ctx* value may be NULL in which case the default library context is used. The *propq* value may also be NULL.

EC_GROUP_new_by_curve_name() is the same as **EC_GROUP_new_by_curve_name_ex()** except that the default library context is always used along with a NULL property query string.

EC_GROUP_free() frees the memory associated with the EC_GROUP. If*gr oup* is NULL nothing is done.

EC_GROUP_clear_free() is deprecated: it was meant to destroy any sensitive data held within the EC_GROUP and then free its memory, but since all the data stored in the EC_GROUP is public anyway, this function is unnecessary. Its use can be safely replaced with**EC_GR OUP_free()**. If*gr oup* is NULL nothing is done.

OSSL_EC_curve_nid2name() converts a curve *nid* into the corresponding name.

RETURN VALUES

All EC_GROUP_new* functions return a pointer to the newly constructed group, or NULL on error.

EC_get_builtin_curves() returns the number of built−in curves that are available.

EC_GROUP_set_curve_GFp(), **EC_GROUP_get_curve_GFp()**, **EC_GROUP_set_curve_GF2m()**, **EC_GROUP_get_curve_GF2m()** return 1 on success or 0 on error.

OSSL_EC_curve_nid2name() returns a character string constant, or NULL on error.

SEE ALSO

crypto(7), **EC_GROUP_copy**(3), **EC_POINT_new**(3), **EC_POINT_add**(3), **EC_KEY_new**(3), **EC_GFp_simple_method**(3), **d2i_ECPKParameters**(3), **OSSL_LIB_CTX**(3), **EVP_PKEY−EC**(7)

HISTORY

EC_GROUP_to_params() was added in OpenSSL 3.2.

- **EC_GROUP_new()** was deprecated in OpenSSL 3.0.

  EC_GROUP_new_by_curve_name_ex() and **EC_GROUP_new_from_params()** were added in OpenSSL 3.0.

- **EC_GROUP_clear_free()** was deprecated in OpenSSL 3.0; use **EC_GROUP_free()** instead.

-

```
EC_GROUP_set_curve_GFp(), EC_GROUP_get_curve_GFp(),
EC_GROUP_set_curve_GF2m() and EC_GROUP_get_curve_GF2m() were deprecated in
OpenSSL 3.0; use EC_GROUP_set_curve() and EC_GROUP_get_curve() instead.
```

COPYRIGHT