Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'EC_GROUP_get_curve_name.3ossl' command

$ man EC_GROUP_get_curve_name.3ossl

EC_GROUP_COPY(3ossl) OpenSSL EC_GROUP_COPY(3ossl)

NAME

EC_GROUP_get0_order, EC_GROUP_order_bits, EC_GROUP_get0_cofactor,

EC_GROUP_copy, EC_GROUP_dup, EC_GROUP_method_of,

EC_GROUP_set_generator, EC_GROUP_get0_generator, EC_GROUP_get_order,

EC_GROUP_get_cofactor, EC_GROUP_set_curve_name,

EC_GROUP_get_curve_name, EC_GROUP_set_asn1_flag,

EC_GROUP_get_asn1_flag, EC_GROUP_set_point_conversion_form,

EC_GROUP_get_point_conversion_form, EC_GROUP_get0_seed,

EC_GROUP_get_seed_len, EC_GROUP_set_seed, EC_GROUP_get_degree,

EC_GROUP_check, EC_GROUP_check_named_curve,

EC_GROUP_check_discriminant, EC_GROUP_cmp, EC_GROUP_get_basis_type,

EC_GROUP_get_trinomial_basis, EC_GROUP_get_pentanomial_basis,

EC_GROUP_get0_field, EC_GROUP_get_field_type - Functions for

manipulating EC_GROUP objects

SYNOPSIS

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

int EC_GROUP_copy(EC_GROUP *dst, const EC_GROUP *src);

EC_GROUP *EC_GROUP_dup(const EC_GROUP *src);

int EC_GROUP_set_generator(EC_GROUP *group, const EC_POINT *generator,
                           const BIGNUM *order, const BIGNUM *cofactor);

const EC_POINT *EC_GROUP_get0_generator(const EC_GROUP *group);

int EC_GROUP_get_order(const EC_GROUP *group, BIGNUM *order, BN_CTX *ctx);
```

```
const BIGNUM *EC_GROUP_get0_order(const EC_GROUP *group);
int EC_GROUP_order_bits(const EC_GROUP *group);
int EC_GROUP_get_cofactor(const EC_GROUP *group, BIGNUM *cofactor, BN_CTX *ctx);
const BIGNUM *EC_GROUP_get0_cofactor(const EC_GROUP *group);
const BIGNUM *EC_GROUP_get0_field(const EC_GROUP *group);
void EC_GROUP_set_curve_name(EC_GROUP *group, int nid);
int EC_GROUP_get_curve_name(const EC_GROUP *group);
void EC_GROUP_set_asn1_flag(EC_GROUP *group, int flag);
int EC_GROUP_get_asn1_flag(const EC_GROUP *group);
void EC_GROUP_set_point_conversion_form(EC_GROUP *group, point_conversion_form_t form);
point_conversion_form_t EC_GROUP_get_point_conversion_form(const EC_GROUP *group);
unsigned char *EC_GROUP_get0_seed(const EC_GROUP *group);
size_t EC_GROUP_get_seed_len(const EC_GROUP *group);
size_t EC_GROUP_set_seed(EC_GROUP *group, const unsigned char *, size_t len);
int EC_GROUP_get_degree(const EC_GROUP *group);
int EC_GROUP_check(const EC_GROUP *group, BN_CTX *ctx);
int EC_GROUP_check_named_curve(const EC_GROUP *group, int nist_only,
                               BN_CTX *ctx);
int EC_GROUP_check_discriminant(const EC_GROUP *group, BN_CTX *ctx);
int EC_GROUP_cmp(const EC_GROUP *a, const EC_GROUP *b, BN_CTX *ctx);
int EC_GROUP_get_basis_type(const EC_GROUP *group);
int EC_GROUP_get_trinomial_basis(const EC_GROUP *group, unsigned int *k);
int EC_GROUP_get_pentanomial_basis(const EC_GROUP *group, unsigned int *k1,
                                   unsigned int *k2, unsigned int *k3);
int EC_GROUP_get_field_type(const EC_GROUP *group);
```

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):

```
const EC_METHOD *EC_GROUP_method_of(const EC_GROUP *group);
```

DESCRIPTION

EC_GROUP_copy() copies the curve src into dst. Both src and dst must use the same EC_METHOD.

EC_GROUP_dup() creates a new EC_GROUP object and copies the content from src to the newly created EC_GROUP object.

EC_GROUP_method_of() obtains the EC_METHOD of group. This function was deprecated in OpenSSL 3.0, since EC_METHOD is no longer a public concept.

EC_GROUP_set_generator() sets curve parameters that must be agreed by all participants using the curve. These parameters include the generator, the order and the cofactor. The generator is a well defined point on the curve chosen for cryptographic operations. Integers used for point multiplications will be between 0 and n-1 where n is the order. The order multiplied by the cofactor gives the number of points on the curve.

EC_GROUP_get0_generator() returns the generator for the identified group.

EC_GROUP_get_order() retrieves the order of group and copies its value into order. It fails in case group is not fully initialized (i.e., its order is not set or set to zero).

EC_GROUP_get_cofactor() retrieves the cofactor of group and copies its value into cofactor. It fails in case group is not fully initialized or if the cofactor is not set (or set to zero).

The functions EC_GROUP_set_curve_name() and EC_GROUP_get_curve_name(), set and get the NID for the curve respectively (see EC_GROUP_new(3)). If a curve does not have a NID associated with it, then EC_GROUP_get_curve_name will return NID_undef.

The asn1_flag value is used to determine whether the curve encoding uses explicit parameters or a named curve using an ASN1 OID: many applications only support the latter form. If asn1_flag is OPENSSL_EC_NAMED_CURVE then the named curve form is used and the parameters must have a corresponding named curve NID set. If asn1_flags is OPENSSL_EC_EXPLICIT_CURVE the parameters are explicitly encoded. The functions EC_GROUP_get_asn1_flag() and EC_GROUP_set_asn1_flag() get and set the status of the asn1_flag for the curve. Note: OPENSSL_EC_EXPLICIT_CURVE was added in OpenSSL 1.1.0, for previous versions of OpenSSL the value 0 must be used instead. Before OpenSSL

1.1.0 the default form was to use explicit parameters (meaning that applications would have to explicitly set the named curve form) in OpenSSL 1.1.0 and later the named curve form is the default.

The point_conversion_form for a curve controls how EC_POINT data is encoded as ASN1 as defined in X9.62 (ECDSA). point_conversion_form_t is an enum defined as follows:

```
 typedef enum {
        /** the point is encoded as z||x, where the octet z specifies
         *   which solution of the quadratic equation y is  */
        POINT_CONVERSION_COMPRESSED = 2,
        /** the point is encoded as z||x||y, where z is the octet 0x04  */
        POINT_CONVERSION_UNCOMPRESSED = 4,
        /** the point is encoded as z||x||y, where the octet z specifies
         *  which solution of the quadratic equation y is  */
        POINT_CONVERSION_HYBRID = 6
 } point_conversion_form_t;
```

For POINT_CONVERSION_UNCOMPRESSED the point is encoded as an octet signifying the UNCOMPRESSED form has been used followed by the octets for x, followed by the octets for y.

For any given x co-ordinate for a point on a curve it is possible to derive two possible y values. For POINT_CONVERSION_COMPRESSED the point is encoded as an octet signifying that the COMPRESSED form has been used AND which of the two possible solutions for y has been used, followed by the octets for x.

For POINT_CONVERSION_HYBRID the point is encoded as an octet signifying the HYBRID form has been used AND which of the two possible solutions for y has been used, followed by the octets for x, followed by the octets for y.

The functions EC_GROUP_set_point_conversion_form() and EC_GROUP_get_point_conversion_form(), set and get the point_conversion_form for the curve respectively.

ANSI X9.62 (ECDSA standard) defines a method of generating the curve parameter b from a random number. This provides advantages in that a parameter obtained in this way is highly unlikely to be susceptible to special purpose attacks, or have any trapdoors in it. If the seed is present for a curve then the b parameter was generated in a verifiable fashion using that seed. The OpenSSL EC library does not use this seed value but does enable you to inspect it using EC_GROUP_get0_seed(). This returns a pointer to a memory block containing the seed that was used. The length of the memory block can be obtained using EC_GROUP_get_seed_len(). A number of the built-in curves within the library provide seed values that can be obtained. It is also possible to set a custom seed using EC_GROUP_set_seed() and passing a pointer to a memory block, along with the length of the seed. Again, the EC library will not use this seed value, although it will be preserved in any ASN1 based communications.

EC_GROUP_get_degree() gets the degree of the field. For Fp fields this will be the number of bits in p. For F2^m fields this will be the value m.

EC_GROUP_get_field_type() identifies what type of field the EC_GROUP structure supports, which will be either F2^m or Fp.

The function EC_GROUP_check_discriminant() calculates the discriminant for the curve and verifies that it is valid. For a curve defined over Fp the discriminant is given by the formula 4*a^3 + 27*b^2 whilst for F2^m curves the discriminant is simply b. In either case for the curve to be valid the discriminant must be non zero.

The function EC_GROUP_check() behaves in the following way: For the OpenSSL default provider it performs a number of checks on a curve to verify that it is valid. Checks performed include verifying that the discriminant is non zero; that a generator has been defined; that the generator is on the curve and has the correct order. For the OpenSSL FIPS provider it uses EC_GROUP_check_named_curve() to conform to SP800-56Ar3.

The function EC_GROUP_check_named_curve() determines if the group's domain parameters match one of the built-in curves supported by the library. The curve name is returned as a NID if it matches. If the group's domain parameters have been modified then no match will be found. If the curve name of the given group is NID_undef (e.g. it has been created by using explicit parameters with no curve name), then this method can be used to lookup the name of the curve that matches the group domain parameters. The built-in curves contain aliases, so that multiple NID's can map to the same domain parameters. For such curves it is unspecified which of the aliases will be returned if the curve name of the given group is NID_undef. If nist_only is 1 it will only look for NIST approved curves, otherwise it searches all built-in curves. This function may be passed a BN_CTX object in the ctx parameter. The ctx parameter may be NULL.

EC_GROUP_cmp() compares a and b to determine whether they represent the same curve or not.

The functions EC_GROUP_get_basis_type(), EC_GROUP_get_trinomial_basis() and EC_GROUP_get_pentanomial_basis() should only be called for curves defined over an F2^m field. Addition and multiplication operations within an F2^m field are performed using an irreducible polynomial function f(x). This function is either a trinomial of the form:

f(x) = x^m + x^k + 1 with m > k >= 1

or a pentanomial of the form:

f(x) = x^m + x^k3 + x^k2 + x^k1 + 1 with m > k3 > k2 > k1 >= 1

The function EC_GROUP_get_basis_type() returns a NID identifying whether a trinomial or pentanomial is in use for the field. The function EC_GROUP_get_trinomial_basis() must only be called where f(x) is of the trinomial form, and returns the value of k. Similarly the function EC_GROUP_get_pentanomial_basis() must only be called where f(x) is of the pentanomial form, and returns the values of k1, k2 and k3 respectively.

RETURN VALUES

The following functions return 1 on success or 0 on error:

EC_GROUP_copy(), EC_GROUP_set_generator(), EC_GROUP_check(), EC_GROUP_check_discriminant(), EC_GROUP_get_trinomial_basis() and EC_GROUP_get_pentanomial_basis().

EC_GROUP_dup() returns a pointer to the duplicated curve, or NULL on error.

EC_GROUP_method_of() returns the EC_METHOD implementation in use for the given curve or NULL on error.

EC_GROUP_get0_generator() returns the generator for the given curve or NULL on error.

EC_GROUP_get_order() returns 0 if the order is not set (or set to zero) for group or if copying into order fails, 1 otherwise.

EC_GROUP_get_cofactor() returns 0 if the cofactor is not set (or is set to zero) for group or if copying into cofactor fails, 1 otherwise.

EC_GROUP_get_curve_name() returns the curve name (NID) for group or will return NID_undef if no curve name is associated.

EC_GROUP_get_asn1_flag() returns the ASN1 flag for the specified group

.

EC_GROUP_get_point_conversion_form() returns the point_conversion_form for group.

EC_GROUP_get_degree() returns the degree for group or 0 if the operation is not supported by the underlying group implementation.

EC_GROUP_get_field_type() returns either NID_X9_62_prime_field for prime curves or NID_X9_62_characteristic_two_field for binary curves; these values are defined in the <openssl/obj_mac.h> header file.

EC_GROUP_check_named_curve() returns the nid of the matching named curve, otherwise it returns 0 for no match, or -1 on error.

EC_GROUP_get0_order() returns an internal pointer to the group order.

EC_GROUP_order_bits() returns the number of bits in the group order.

EC_GROUP_get0_cofactor() returns an internal pointer to the group cofactor. EC_GROUP_get0_field() returns an internal pointer to the group field. For curves over GF(p), this is the modulus; for curves over GF(2^m), this is the irreducible polynomial defining the field.

EC_GROUP_get0_seed() returns a pointer to the seed that was used to generate the parameter b, or NULL if the seed is not specified.

EC_GROUP_get_seed_len() returns the length of the seed or 0 if the seed is not specified.

EC_GROUP_set_seed() returns the length of the seed that has been set.

If the supplied seed is NULL, or the supplied seed length is 0, the

return value will be 1. On error 0 is returned.

EC_GROUP_cmp() returns 0 if the curves are equal, 1 if they are not

equal, or -1 on error.

EC_GROUP_get_basis_type() returns the values NID_X9_62_tpBasis or

NID_X9_62_ppBasis (as defined in <openssl/obj_mac.h>) for a trinomial

or pentanomial respectively. Alternatively in the event of an error a 0

is returned.

SEE ALSO

crypto(7), EC_GROUP_new(3), EC_POINT_new(3), EC_POINT_add(3),

EC_KEY_new(3), EC_GFp_simple_method(3), d2i_ECPKParameters(3)

HISTORY

EC_GROUP_method_of() was deprecated in OpenSSL 3.0.

EC_GROUP_check_named_curve() and EC_GROUP_get_field_type() were added

in OpenSSL 3.0.

COPYRIGHT