NAME

ECDSA_SIG_new, ECDSA_SIG_free, ECDSA_SIG_get0, ECDSA_SIG_get0_r, ECDSA_SIG_get0_s, ECDSA_SIG_set0 - Functions for creating, destroying and manipulating ECDSA_SIG objects

SYNOPSIS

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

ECDSA_SIG *ECDSA_SIG_new(void);
void ECDSA_SIG_free(ECDSA_SIG *sig);
void ECDSA_SIG_get0(const ECDSA_SIG *sig, const BIGNUM **pr, const BIGNUM **ps);
const BIGNUM *ECDSA_SIG_get0_r(const ECDSA_SIG *sig);
const BIGNUM *ECDSA_SIG_get0_s(const ECDSA_SIG *sig);
int ECDSA_SIG_set0(ECDSA_SIG *sig, BIGNUM *r, BIGNUM *s);
```

DESCRIPTION

ECDSA_SIG is an opaque structure consisting of two BIGNUMs for the *r* and *s* value of an Elliptic Curve Digital Signature Algorithm (ECDSA) signature (see FIPS186-4 or X9.62). The **ECDSA_SIG** object was mainly used by the deprecated low level functions described in **ECDSA_sign**(3), it is still required in order to be able to set or get the values of *r* and *s* into or from a signature. This is mainly used for testing purposes as shown in the "EXAMPLES".

ECDSA_SIG_new() allocates an empty **ECDSA_SIG** structure. Note: before OpenSSL 1.1.0, the *r* and *s* components were initialised.

ECDSA_SIG_free() frees the **ECDSA_SIG** structure *sig*.

ECDSA_SIG_get0() returns internal pointers the *r* and *s* values contained in *sig* and stores them in **pr* and **ps*, respectively. The pointer *pr* or *ps* can be NULL, in which case the corresponding value is not returned.

The values *r*, *s* can also be retrieved separately by the corresponding function **ECDSA_SIG_get0_r()** and **ECDSA_SIG_get0_s()**, respectively.

Non-NULL *r* and *s* values can be set on the *sig* by calling **ECDSA_SIG_set0()**. Calling this function transfers the memory management of the values to the **ECDSA_SIG** object, and therefore the values that have been passed in should not be freed by the caller.

See **i2d_ECDSA_SIG**(3) and **d2i_ECDSA_SIG**(3) for information about encoding and decoding ECDSA signatures to/from DER.

RETURN VALUES

ECDSA_SIG_new() returns NULL if the allocation fails.

ECDSA_SIG_set0() returns 1 on success or 0 on failure.

ECDSA_SIG_get0_r() and **ECDSA_SIG_get0_s()** return the corresponding value, or NULL if it is unset.

EXAMPLES

Extract signature *r* and *s* values from a ECDSA *signature* of size *signaturelen*:

```
ECDSA_SIG *obj;
const BIGNUM *r, *s;

/* Load a signature into the ECDSA_SIG object */
obj = d2i_ECDSA_SIG(NULL, &signature, signaturelen);
if (obj == NULL)
    /* error */

r = ECDSA_SIG_get0_r(obj);
s = ECDSA_SIG_get0_s(obj);
if (r == NULL || s == NULL)
    /* error */

/* Use BN_bn2binpad() here to convert to r and s into byte arrays */

/*
 * Do not try to access I<r> or I<s> after calling ECDSA_SIG_free(),
 * as they are both freed by this call.
 */
ECDSA_SIG_free(obj);
```

Convert *r* and *s* byte arrays into an ECDSA_SIG *signature* of size *signaturelen*:

```
ECDSA_SIG *obj = NULL;
unsigned char *signature = NULL;
size_t signaturelen;
BIGNUM *rbn = NULL, *sbn = NULL;

obj = ECDSA_SIG_new();
```

```
if (obj == NULL)
    /* error */
rbn = BN_bin2bn(r, rlen, NULL);
sbn = BN_bin2bn(s, slen, NULL);
if (rbn == NULL || sbn == NULL)
    /* error */

if (!ECDSA_SIG_set0(obj, rbn, sbn))
    /* error */
/* Set these to NULL since they are now owned by obj */
rbn = sbn = NULL;

signaturelen = i2d_ECDSA_SIG(obj, &signature);
if (signaturelen <= 0)
    /* error */

/*
 * This signature could now be passed to L<EVP_DigestVerify(3)>
 * or L<EVP_DigestVerifyFinal(3)>
 */

BN_free(rbn);
BN_free(sbn);
OPENSSL_free(signature);
ECDSA_SIG_free(obj);
```

CONFORMING TO

ANSI X9.62, US Federal Information Processing Standard FIPS186-4 (Digital Signature Standard, DSS)

SEE ALSO

EC_KEY_new(3), **EVP_DigestSignInit**(3), **EVP_DigestVerifyInit**(3), **EVP_PKEY_sign**(3) **i2d_ECDSA_SIG**(3), **d2i_ECDSA_SIG**(3), **ECDSA_sign**(3)

COPYRIGHT