



*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 'ECDSA\_SIG\_set0.3oss1' command**

**\$ man ECDSA\_SIG\_set0.3oss1**

ECDSA\_SIG\_NEW(3oss1)          OpenSSL          ECDSA\_SIG\_NEW(3oss1)

### NAME

ECDSA\_SIG\_get0, ECDSA\_SIG\_get0\_r, ECDSA\_SIG\_get0\_s, ECDSA\_SIG\_set0,  
ECDSA\_SIG\_new, ECDSA\_SIG\_free, ECDSA\_size, ECDSA\_sign, ECDSA\_do\_sign,  
ECDSA\_verify, ECDSA\_do\_verify, ECDSA\_sign\_setup, ECDSA\_sign\_ex,  
ECDSA\_do\_sign\_ex - low-level elliptic curve digital signature algorithm  
(ECDSA) functions

### 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);
```

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):

```
int ECDSA_size(const EC_KEY *eckey);

int ECDSA_sign(int type, const unsigned char *dgst, int dgstlen,
               unsigned char *sig, unsigned int *siglen, EC_KEY *eckey);

ECDSA_SIG *ECDSA_do_sign(const unsigned char *dgst, int dgst_len,
                          EC_KEY *eckey);
```

```

int ECDSA_verify(int type, const unsigned char *dgst, int dgstlen,
                 const unsigned char *sig, int siglen, EC_KEY *eckey);
int ECDSA_do_verify(const unsigned char *dgst, int dgst_len,
                   const ECDSA_SIG *sig, EC_KEY* eckey);
ECDSA_SIG *ECDSA_do_sign_ex(const unsigned char *dgst, int dgstlen,
                             const BIGNUM *kinv, const BIGNUM *rp,
                             EC_KEY *eckey);
int ECDSA_sign_setup(EC_KEY *eckey, BN_CTX *ctx, BIGNUM **kinv, BIGNUM **rp);
int ECDSA_sign_ex(int type, const unsigned char *dgst, int dgstlen,
                  unsigned char *sig, unsigned int *siglen,
                  const BIGNUM *kinv, const BIGNUM *rp, EC_KEY *eckey);

```

## DESCRIPTION

ECDSA\_SIG is an opaque structure consisting of two BIGNUMs for the r and s value of an ECDSA signature (see X9.62 or FIPS186-2).

ECDSA\_SIG\_new() allocates an empty ECDSA\_SIG structure. Note: before OpenSSL 1.1.0 the: 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.

All of the functions described below are deprecated. Applications should use the higher level EVP interface such as EVP\_DigestSignInit(3) or EVP\_DigestVerifyInit(3) instead.

ECDSA\_size() returns the maximum length of a DER encoded ECDSA signature created with the private EC key eckey. To obtain the actual

signature size use `EVP_PKEY_sign(3)` with a `NULL` sig parameter.

`ECDSA_sign()` computes a digital signature of the `dgstlen` bytes hash value `dgst` using the private EC key `eckey`. The DER encoded signatures is stored in `sig` and its length is returned in `sig_len`. Note: `sig` must point to `ECDSA_size(eckey)` bytes of memory. The parameter type is currently ignored. `ECDSA_sign()` is wrapper function for `ECDSA_sign_ex()` with `kinv` and `rp` set to `NULL`.

`ECDSA_do_sign()` is similar to `ECDSA_sign()` except the signature is returned as a newly allocated `ECDSA_SIG` structure (or `NULL` on error).

`ECDSA_do_sign()` is a wrapper function for `ECDSA_do_sign_ex()` with `kinv` and `rp` set to `NULL`.

`ECDSA_verify()` verifies that the signature in `sig` of size `siglen` is a valid ECDSA signature of the hash value `dgst` of size `dgstlen` using the public key `eckey`. The parameter type is ignored.

`ECDSA_do_verify()` is similar to `ECDSA_verify()` except the signature is presented in the form of a pointer to an `ECDSA_SIG` structure.

The remaining functions utilise the internal `kinv` and `r` values used during signature computation. Most applications will never need to call these and some external ECDSA ENGINE implementations may not support them at all if either `kinv` or `r` is not `NULL`.

`ECDSA_sign_setup()` may be used to precompute parts of the signing operation. `eckey` is the private EC key and `ctx` is a pointer to `BN_CTX` structure (or `NULL`). The precomputed values are returned in `kinv` and `rp` and can be used in a later call to `ECDSA_sign_ex()` or `ECDSA_do_sign_ex()`.

`ECDSA_sign_ex()` computes a digital signature of the `dgstlen` bytes hash value `dgst` using the private EC key `eckey` and the optional pre-computed values `kinv` and `rp`. The DER encoded signature is stored in `sig` and its length is returned in `sig_len`. Note: `sig` must point to

`ECDSA_size(eckey)` bytes of memory. The parameter type is ignored.

`ECDSA_do_sign_ex()` is similar to `ECDSA_sign_ex()` except the signature is returned as a newly allocated `ECDSA_SIG` structure (or `NULL` on error).

## 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.

ECDSA\_size() returns the maximum length signature or 0 on error.

ECDSA\_sign(), ECDSA\_sign\_ex() and ECDSA\_sign\_setup() return 1 if successful or 0 on error.

ECDSA\_do\_sign() and ECDSA\_do\_sign\_ex() return a pointer to an allocated ECDSA\_SIG structure or NULL on error.

ECDSA\_verify() and ECDSA\_do\_verify() return 1 for a valid signature, 0 for an invalid signature and -1 on error. The error codes can be obtained by ERR\_get\_error(3).

## EXAMPLES

Creating an ECDSA signature of a given SHA-256 hash value using the named curve prime256v1 (aka P-256).

First step: create an EC\_KEY object (note: this part is not ECDSA specific)

```
int ret;
ECDSA_SIG *sig;
EC_KEY *eckey;
eckey = EC_KEY_new_by_curve_name(NID_X9_62_prime256v1);
if (eckey == NULL)
    /* error */
if (EC_KEY_generate_key(eckey) == 0)
    /* error */
```

Second step: compute the ECDSA signature of a SHA-256 hash value using

ECDSA\_do\_sign():

```
sig = ECDSA_do_sign(digest, 32, eckey);
if (sig == NULL)
    /* error */
```

or using ECDSA\_sign():

```
unsigned char *buffer, *pp;
```

```

int buf_len;

buf_len = ECDSA_size(eckey);

buffer = OPENSSL_malloc(buf_len);

pp = buffer;

if (ECDSA_sign(0, dgst, dgstlen, pp, &buf_len, eckey) == 0)
    /* error */

```

Third step: verify the created ECDSA signature using ECDSA\_do\_verify():

```

ret = ECDSA_do_verify(digest, 32, sig, eckey);

or using ECDSA_verify():

ret = ECDSA_verify(0, digest, 32, buffer, buf_len, eckey);

```

and finally evaluate the return value:

```

if (ret == 1)
    /* signature ok */

else if (ret == 0)
    /* incorrect signature */

else
    /* error */

```

## CONFORMING TO

ANSI X9.62, US Federal Information Processing Standard FIPS186-2  
(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)

## HISTORY

The ECDSA\_size(), ECDSA\_sign(), ECDSA\_do\_sign(), ECDSA\_verify(),  
ECDSA\_do\_verify(), ECDSA\_sign\_setup(), ECDSA\_sign\_ex() and  
ECDSA\_do\_sign\_ex() functions were deprecated in OpenSSL 3.0.

## COPYRIGHT

Copyright 2004-2022 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>>.

