Rocky Enterprise Linux 9.2 Manual Pages on command 'OSSL_DECODER_from_data.3ossl'

$ man OSSL_DECODER_from_data.3ossl

OSSL_DECODER_FROM_BIO(3ossl) OpenSSL OSSL_DECODER_FROM_BIO(3ossl)

NAME

OSSL_DECODER_from_data, OSSL_DECODER_from_bio, OSSL_DECODER_from_fp -

Routines to perform a decoding

SYNOPSIS

#include <openssl/decoder.h>

int OSSL_DECODER_from_bio(OSSL_DECODER_CTX *ctx, BIO *in);

int OSSL_DECODER_from_fp(OSSL_DECODER_CTX *ctx, FILE *fp);

int OSSL_DECODER_from_data(OSSL_DECODER_CTX *ctx, const unsigned char **pdata,

size_t *pdata_len);

Feature availability macros:

OSSL_DECODER_from_fp() is only available when OPENSSL_NO_STDIO is

undefined.

DESCRIPTION

OSSL_DECODER_from_data() runs the decoding process for the context ctx,
with input coming from *pdata, *pdata_len bytes long. Both *pdata and
*pdata_len must be non-NULL. When OSSL_DECODER_from_data() returns,
*pdata is updated to point at the location after what has been decoded,
and *pdata_len to have the number of remaining bytes.

OSSL_DECODER_from_bio() runs the decoding process for the context ctx,
with the input coming from the BIO in. Should it make a difference,
it's recommended to have the BIO set in binary mode rather than text
mode.

OSSL_DECODER_from_fp() does the same thing as OSSL_DECODER_from_bio(),
except that the input is coming from the FILE fp.

RETURN VALUES

OSSL_DECODER_from_bio() and OSSL_DECODER_from_fp() return 1 on success,
or 0 on failure.

EXAMPLES

To decode an RSA key encoded with PEM from a bio:

```
OSSL_DECODER_CTX *dctx;
EVP_PKEY *pkey = NULL;
const char *format = "PEM";   /* NULL for any format */
const char *structure = NULL; /* any structure */
const char *keytype = "RSA";  /* NULL for any key */
const unsigned char *pass = "my password";

dctx = OSSL_DECODER_CTX_new_for_pkey(&pkey, format, structure,
                                     keytype,
                                     OSSL_KEYMGMT_SELECT_KEYPAIR,
```

```
                                NULL, NULL);
if (dctx == NULL) {
    /* error: no suitable potential decoders found */
}
if (pass != NULL)
    OSSL_DECODER_CTX_set_passphrase(dctx, pass, strlen(pass));
if (OSSL_DECODER_from_bio(dctx, bio)) {
    /* pkey is created with the decoded data from the bio */
} else {
    /* decoding failure */
}
OSSL_DECODER_CTX_free(dctx);
```

To decode an EC key encoded with DER from a buffer:

```
OSSL_DECODER_CTX *dctx;
EVP_PKEY *pkey = NULL;
const char *format = "DER";   /* NULL for any format */
const char *structure = NULL; /* any structure */
const char *keytype = "EC";   /* NULL for any key */
const unsigned char *pass = NULL
const unsigned char *data = buffer;
size_t datalen = sizeof(buffer);

dctx = OSSL_DECODER_CTX_new_for_pkey(&pkey, format, structure,
                                keytype,
                                OSSL_KEYMGMT_SELECT_KEYPAIR
                                | OSSL_KEYMGMT_SELECT_DOMAIN_PARAMETERS,
                                NULL, NULL);
if (dctx == NULL) {
    /* error: no suitable potential decoders found */
}
if (pass != NULL)
```

```
    OSSL_DECODER_CTX_set_passphrase(dctx, pass, strlen(pass));
if (OSSL_DECODER_from_data(dctx, &data, &datalen)) {
    /* pkey is created with the decoded data from the buffer */
} else {
    /* decoding failure */
}
OSSL_DECODER_CTX_free(dctx);
```

SEE ALSO

provider(7), OSSL_DECODER_CTX(3)

HISTORY

The functions described here were added in OpenSSL 3.0.

COPYRIGHT