NAME

OSSL_ENCODER_to_data, OSSL_ENCODER_to_bio, OSSL_ENCODER_to_fp - Routines to perform an encoding

SYNOPSIS

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

int OSSL_ENCODER_to_data(OSSL_ENCODER_CTX *ctx, unsigned char **pdata,
                         size_t *pdata_len);
int OSSL_ENCODER_to_bio(OSSL_ENCODER_CTX *ctx, BIO *out);
int OSSL_ENCODER_to_fp(OSSL_ENCODER_CTX *ctx, FILE *fp);
```

Feature availability macros:

OSSL_ENCODER_to_fp() is only available when **OPENSSL_NO_STDIO** is undefined.

DESCRIPTION

OSSL_ENCODER_to_data() runs the encoding process for the context *ctx*, with the output going to the **pdata* and **pdata_len*. If **pdata* is NULL when **OSSL_ENCODER_to_data()** is called, a buffer will be allocated using **OPENSSL_zalloc**(3), and **pdata* will be set to point at the start of that buffer, and **pdata_len* will be assigned its length when **OSSL_ENCODER_to_data()** returns. If **pdata* is non-NULL when **OSSL_ENCODER_to_data()** is called, **pdata_len* is assumed to have its size. In this case, **pdata* will be set to point after the encoded bytes, and **pdata_len* will be assigned the number of remaining bytes.

OSSL_ENCODER_to_bio() runs the encoding process for the context *ctx*, with the output going to the **BIO** *out*.

OSSL_ENCODER_to_fp() does the same thing as **OSSL_ENCODER_to_bio()**, except that the output is going to the **FILE** *fp*.

For **OSSL_ENCODER_to_bio()** and **OSSL_ENCODER_to_fp()**, the application is required to set up the **BIO** or **FILE** properly, for example to have it in text or binary mode as is appropriate for the encoder output type.

RETURN VALUES

OSSL_ENCODER_to_bio(), **OSSL_ENCODER_to_fp()** and **OSSL_ENCODER_to_data()** return 1 on success, or 0 on failure.

EXAMPLES

To encode a pkey as PKCS#8 with PEM format into a bio:

```
OSSL_ENCODER_CTX *ectx;
const char *format = "PEM";
const char *structure = "PrivateKeyInfo"; /* PKCS#8 structure */
const unsigned char *pass = "my password";

ectx = OSSL_ENCODER_CTX_new_for_pkey(pkey,
                                     OSSL_KEYMGMT_SELECT_KEYPAIR
                                     | OSSL_KEYMGMT_SELECT_DOMAIN_PARAMETERS,
                                     format, structure,
                                     NULL);
if (ectx == NULL) {
    /* error: no suitable potential encoders found */
}
if (pass != NULL)
    OSSL_ENCODER_CTX_set_passphrase(ectx, pass, strlen(pass));
if (OSSL_ENCODER_to_bio(ectx, bio)) {
    /* pkey was successfully encoded into the bio */
} else {
    /* encoding failure */
}
OSSL_ENCODER_CTX_free(ectx);
```

To encode a pkey as PKCS#8 with DER format encrypted with AES-256-CBC into a buffer:

```
OSSL_ENCODER_CTX *ectx;
const char *format = "DER";
const char *structure = "PrivateKeyInfo"; /* PKCS#8 structure */
const unsigned char *pass = "my password";
unsigned char *data = NULL;
size_t datalen;

ectx = OSSL_ENCODER_CTX_new_for_pkey(pkey,
                                     OSSL_KEYMGMT_SELECT_KEYPAIR
                                     | OSSL_KEYMGMT_SELECT_DOMAIN_PARAMETERS,
                                     format, structure,
                                     NULL);
if (ectx == NULL) {
    /* error: no suitable potential encoders found */
```

```
}
if (pass != NULL) {
    OSSL_ENCODER_CTX_set_passphrase(ectx, pass, strlen(pass));
    OSSL_ENCODER_CTX_set_cipher(ctx, "AES-256-CBC", NULL);
}
if (OSSL_ENCODER_to_data(ectx, &data, &datalen)) {
    /*
     * pkey was successfully encoded into a newly allocated
     * data buffer
     */
} else {
    /* encoding failure */
}
OSSL_ENCODER_CTX_free(ectx);
```

SEE ALSO

provider(7), **OSSL_ENCODER_CTX**(3)

HISTORY

The functions described here were added in OpenSSL 3.0.

COPYRIGHT