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 'BIO_f_base64.3ossl' command

$ man BIO_f_base64.3ossl

BIO_F_BASE64(3ossl) OpenSSL BIO_F_BASE64(3ossl)

NAME

BIO_f_base64 - base64 BIO filter

SYNOPSIS

#include <openssl/bio.h>

#include <openssl/evp.h>

const BIO_METHOD *BIO_f_base64(void);

DESCRIPTION

BIO_f_base64() returns the base64 BIO method. This is a filter BIO that base64 encodes any data written through it and decodes any data read through it.

Base64 BIOs do not support BIO_gets() or BIO_puts().

For writing, output is by default divided to lines of length 64 characters and there is always a newline at the end of output.

For reading, first line should be at most 1024 characters long. If it is longer then it is ignored completely. Other input lines can be of

any length. There must be a newline at the end of input.

This behavior can be changed with BIO_FLAGS_BASE64_NO_NL flag.

BIO_flush() on a base64 BIO that is being written through is used to signal that no more data is to be encoded: this is used to flush the final block through the BIO.

The flag BIO_FLAGS_BASE64_NO_NL can be set with BIO_set_flags(). For writing, it causes all data to be written on one line without newline at the end. For reading, it expects the data to be all on one line (with or without a trailing newline).

NOTES

Because of the format of base64 encoding the end of the encoded block cannot always be reliably determined.

RETURN VALUES

BIO_f_base64() returns the base64 BIO method.

EXAMPLES

Base64 encode the string "Hello World\n" and write the result to standard output:

```
BIO *bio, *b64;
char message[] = "Hello World \n";

b64 = BIO_new(BIO_f_base64());
bio = BIO_new_fp(stdout, BIO_NOCLOSE);
BIO_push(b64, bio);
BIO_write(b64, message, strlen(message));
BIO_flush(b64);
```

```
BIO_free_all(b64);
```

Read Base64 encoded data from standard input and write the decoded data to standard output:

```
BIO *bio, *b64, *bio_out;
char inbuf[512];
int inlen;

b64 = BIO_new(BIO_f_base64());
bio = BIO_new_fp(stdin, BIO_NOCLOSE);
bio_out = BIO_new_fp(stdout, BIO_NOCLOSE);
BIO_push(b64, bio);
while ((inlen = BIO_read(b64, inbuf, 512)) > 0)
    BIO_write(bio_out, inbuf, inlen);

BIO_flush(bio_out);
BIO_free_all(b64);
```

BUGS

The ambiguity of EOF in base64 encoded data can cause additional data following the base64 encoded block to be misinterpreted.

There should be some way of specifying a test that the BIO can perform to reliably determine EOF (for example a MIME boundary).

COPYRIGHT