NAME

PEM_bytes_read_bio, PEM_bytes_read_bio_secmem - read a PEM-encoded data structure from a BIO

SYNOPSIS

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

 int PEM_bytes_read_bio(unsigned char **pdata, long *plen, char **pnm,
                        const char *name, BIO *bp, pem_password_cb *cb,
                        void *u);
 int PEM_bytes_read_bio_secmem(unsigned char **pdata, long *plen, char **pnm,
                               const char *name, BIO *bp, pem_password_cb *cb,
                               void *u);
```

DESCRIPTION

PEM_bytes_read_bio() reads PEM-formatted (IETF RFC 1421 and IETF RFC 7468) data from the BIO *bp* for the data type given in *name* (RSA PRIVATE KEY, CERTIFICATE, etc.). If multiple PEM-encoded data structures are present in the same stream, **PEM_bytes_read_bio()** will skip non-matching data types and continue reading. Non-PEM data present in the stream may cause an error.

The PEM header may indicate that the following data is encrypted; if so, the data will be decrypted, waiting on user input to supply a passphrase if needed. The password callback *cb* and rock *u* are used to obtain the decryption passphrase, if applicable.

Some data types have compatibility aliases, such as a file containing X509 CERTIFICATE matching a request for the deprecated type CERTIFICATE. The actual type indicated by the file is returned in **pnm* if *pnm* is non-NULL. The caller must free the storage pointed to by **pnm*.

The returned data is the DER-encoded form of the requested type, in **pdata* with length **plen*. The caller must free the storage pointed to by **pdata*.

PEM_bytes_read_bio_secmem() is similar to **PEM_bytes_read_bio()**, but uses memory from the secure heap for its temporary buffers and the storage returned in **pdata* and **pnm*. Accordingly, the caller must use **OPENSSL_secure_free()** to free that storage.

NOTES

PEM_bytes_read_bio_secmem() only enforces that the secure heap is used for storage allocated within the PEM processing stack. The BIO stack from which input is read may also use temporary buffers, which are not necessarily allocated from the secure heap. In cases where it is desirable to ensure that the contents of the PEM file only appears in memory from the secure heap, care is needed in generating the BIO passed as *bp*. In particular, the use of **BIO_s_file()** indicates the use of the operating system stdio functionality, which includes buffering as a feature; **BIO_s_fd()** is likely to be more appropriate in such cases.

These functions make no assumption regarding the pass phrase received from the password callback. It will simply be treated as a byte sequence.

RETURN VALUES

PEM_bytes_read_bio() and **PEM_bytes_read_bio_secmem()** return 1 for success or 0 for failure.

SEE ALSO

PEM_read_bio_ex(3), **passphrase-encoding**(7)

HISTORY

PEM_bytes_read_bio_secmem() was introduced in OpenSSL 1.1.1

COPYRIGHT

Copyright 2017-2018 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>.