NAME

BIO_f_md, BIO_set_md, BIO_get_md, BIO_get_md_ctx - message digest BIO filter

SYNOPSIS

```
#include <openssl/bio.h>
#include <openssl/evp.h>

BIO_METHOD * BIO_f_md(void);
int BIO_set_md(BIO *b,EVP_MD *md);
int BIO_get_md(BIO *b,EVP_MD **mdp);
int BIO_get_md_ctx(BIO *b,EVP_MD_CTX **mdcp);
```

DESCRIPTION

BIO_f_md() returns the message digest BIO method. This is a filter BIO that digests any data passed through it, it is a BIO wrapper for the digest routines *EVP_DigestInit()*, *EVP_DigestUpdate()* and *EVP_DigestFinal()*.

Any data written or read through a digest BIO using *BIO_read()* and *BIO_write()* is digested.

BIO_gets(), if its **size** parameter is large enough finishes the digest calculation and returns the digest value. *BIO_puts()* is not supported.

BIO_reset() reinitialises a digest BIO.

BIO_set_md() sets the message digest of BIO **b** to **md**: this must be called to initialize a digest BIO before any data is passed through it. It is a *BIO_ctrl()* macro.

BIO_get_md() places the a pointer to the digest BIOs digest method in **mdp**, it is a *BIO_ctrl()* macro.

BIO_get_md_ctx() returns the digest BIOs context into **mdcp**.

NOTES

The context returned by *BIO_get_md_ctx()* can be used in calls to *EVP_DigestFinal()* and also the signature routines *EVP_SignFinal()* and *EVP_VerifyFinal()*.

The context returned by *BIO_get_md_ctx()* is an internal context structure. Changes made to this context will affect the digest BIO itself and the context pointer will become invalid when the digest BIO is freed.

After the digest has been retrieved from a digest BIO it must be reinitialized by calling *BIO_reset()*, or *BIO_set_md()* before any more data is passed through it.

If an application needs to call *BIO_gets()* or *BIO_puts()* through a chain containing digest BIOs then this can be done by prepending a buffering BIO.

Before OpenSSL 1.0.0 the call to *BIO_get_md_ctx()* would only work if the BIO had been initialized for example by calling *BIO_set_md()*). In OpenSSL 1.0.0 and later the context is always returned and the BIO is state is set to initialized. This allows applications to initialize the context externally if the standard calls such as *BIO_set_md()* are not sufficiently flexible.

RETURN VALUES

BIO_f_md() returns the digest BIO method.

BIO_set_md(), *BIO_get_md()* and *BIO_md_ctx()* return 1 for success and 0 for failure.

EXAMPLES

The following example creates a BIO chain containing an SHA1 and MD5 digest BIO and passes the string ‘‘Hello World’’ through it. Error checking has been omitted for clarity.

```
BIO *bio, *mdtmp;
char message[] = "Hello World";
bio = BIO_new(BIO_s_null());
mdtmp = BIO_new(BIO_f_md());
BIO_set_md(mdtmp, EVP_sha1());
/* For BIO_push() we want to append the sink BIO and keep a note of
* the start of the chain.
*/
bio = BIO_push(mdtmp, bio);
mdtmp = BIO_new(BIO_f_md());
BIO_set_md(mdtmp, EVP_md5());
bio = BIO_push(mdtmp, bio);
/* Note: mdtmp can now be discarded */
BIO_write(bio, message, strlen(message));
```

The next example digests data by reading through a chain instead:

```
BIO *bio, *mdtmp;
char buf[1024];
int rdlen;
bio = BIO_new_file(file, "rb");
mdtmp = BIO_new(BIO_f_md());
BIO_set_md(mdtmp, EVP_sha1());
bio = BIO_push(mdtmp, bio);
mdtmp = BIO_new(BIO_f_md());
BIO_set_md(mdtmp, EVP_md5());
bio = BIO_push(mdtmp, bio);
do {
rdlen = BIO_read(bio, buf, sizeof(buf));
/* Might want to do something with the data here */
} while(rdlen > 0);
```

This next example retrieves the message digests from a BIO chain and outputs them. This could be used with the examples above.

```
BIO *mdtmp;
unsigned char mdbuf[EVP_MAX_MD_SIZE];
int mdlen;
int i;
mdtmp = bio; /* Assume bio has previously been set up */
do {
EVP_MD *md;
mdtmp = BIO_find_type(mdtmp, BIO_TYPE_MD);
if(!mdtmp) break;
BIO_get_md(mdtmp, &md);
printf("%s digest", OBJ_nid2sn(EVP_MD_type(md)));
mdlen = BIO_gets(mdtmp, mdbuf, EVP_MAX_MD_SIZE);
for(i = 0; i < mdlen; i++) printf(":%02X", mdbuf[i]);
printf("\n");
mdtmp = BIO_next(mdtmp);
} while(mdtmp);

BIO_free_all(bio);
```

BUGS

The lack of support for *BIO_puts()* and the non standard behaviour of *BIO_gets()* could be regarded as anomalous. It could be argued that *BIO_gets()* and *BIO_puts()* should be passed to the next BIO in the chain

and digest the data passed through and that digests should be retrieved using a separate *BIO_ctrl()* call.

SEE ALSO

TBA