

**NAME**

EVP\_MD\_meth\_new, EVP\_MD\_meth\_dup, EVP\_MD\_meth\_free,  
 EVP\_MD\_meth\_set\_input\_blocksize, EVP\_MD\_meth\_set\_result\_size,  
 EVP\_MD\_meth\_set\_app\_datasize, EVP\_MD\_meth\_set\_flags, EVP\_MD\_meth\_set\_init,  
 EVP\_MD\_meth\_set\_update, EVP\_MD\_meth\_set\_final, EVP\_MD\_meth\_set\_copy,  
 EVP\_MD\_meth\_set\_cleanup, EVP\_MD\_meth\_set\_ctrl, EVP\_MD\_meth\_get\_input\_blocksize,  
 EVP\_MD\_meth\_get\_result\_size, EVP\_MD\_meth\_get\_app\_datasize, EVP\_MD\_meth\_get\_flags,  
 EVP\_MD\_meth\_get\_init, EVP\_MD\_meth\_get\_update, EVP\_MD\_meth\_get\_final,  
 EVP\_MD\_meth\_get\_copy, EVP\_MD\_meth\_get\_cleanup, EVP\_MD\_meth\_get\_ctrl - Routines to build  
 up legacy EVP\_MD methods

**SYNOPSIS**

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

The following functions have been deprecated since OpenSSL 3.0, and can be hidden entirely by defining **OPENSSL\_API\_COMPAT** with a suitable version value, see **openssl\_user\_macros(7)**:

```
EVP_MD *EVP_MD_meth_new(int md_type, int pkey_type);
void EVP_MD_meth_free(EVP_MD *md);
EVP_MD *EVP_MD_meth_dup(const EVP_MD *md);

int EVP_MD_meth_set_input_blocksize(EVP_MD *md, int blocksize);
int EVP_MD_meth_set_result_size(EVP_MD *md, int resultsize);
int EVP_MD_meth_set_app_datasize(EVP_MD *md, int datasize);
int EVP_MD_meth_set_flags(EVP_MD *md, unsigned long flags);
int EVP_MD_meth_set_init(EVP_MD *md, int (*init)(EVP_MD_CTX *ctx));
int EVP_MD_meth_set_update(EVP_MD *md, int (*update)(EVP_MD_CTX *ctx,
              const void *data,
              size_t count));
int EVP_MD_meth_set_final(EVP_MD *md, int (*final)(EVP_MD_CTX *ctx,
              unsigned char *md));
int EVP_MD_meth_set_copy(EVP_MD *md, int (*copy)(EVP_MD_CTX *to,
              const EVP_MD_CTX *from));
int EVP_MD_meth_set_cleanup(EVP_MD *md, int (*cleanup)(EVP_MD_CTX *ctx));
int EVP_MD_meth_set_ctrl(EVP_MD *md, int (*ctrl)(EVP_MD_CTX *ctx, int cmd,
              int p1, void *p2));

int EVP_MD_meth_get_input_blocksize(const EVP_MD *md);
int EVP_MD_meth_get_result_size(const EVP_MD *md);
int EVP_MD_meth_get_app_datasize(const EVP_MD *md);
```

```
unsigned long EVP_MD_meth_get_flags(const EVP_MD *md);
int (*EVP_MD_meth_get_init(const EVP_MD *md))(EVP_MD_CTX *ctx);
int (*EVP_MD_meth_get_update(const EVP_MD *md))(EVP_MD_CTX *ctx,
        const void *data,
        size_t count);
int (*EVP_MD_meth_get_final(const EVP_MD *md))(EVP_MD_CTX *ctx,
        unsigned char *md);
int (*EVP_MD_meth_get_copy(const EVP_MD *md))(EVP_MD_CTX *to,
        const EVP_MD_CTX *from);
int (*EVP_MD_meth_get_cleanup(const EVP_MD *md))(EVP_MD_CTX *ctx);
int (*EVP_MD_meth_get_ctrl(const EVP_MD *md))(EVP_MD_CTX *ctx, int cmd,
        int p1, void *p2);
```

## DESCRIPTION

All of the functions described on this page are deprecated. Applications should instead use the OSSL\_PROVIDER APIs.

The **EVP\_MD** type is a structure for digest method implementation. It can also have associated public/private key signing and verifying routines.

**EVP\_MD\_meth\_new()** creates a new **EVP\_MD** structure. These **EVP\_MD** structures are reference counted.

**EVP\_MD\_meth\_dup()** creates a copy of **md**.

**EVP\_MD\_meth\_free()** decrements the reference count for the **EVP\_MD** structure. If the reference count drops to 0 then the structure is freed.

**EVP\_MD\_meth\_set\_input\_blocksize()** sets the internal input block size for the method **md** to **blocksize** bytes.

**EVP\_MD\_meth\_set\_result\_size()** sets the size of the result that the digest method in **md** is expected to produce to **resultsize** bytes.

The digest method may have its own private data, which OpenSSL will allocate for it.

**EVP\_MD\_meth\_set\_app\_datasize()** should be used to set the size for it to **datasize**.

**EVP\_MD\_meth\_set\_flags()** sets the flags to describe optional behaviours in the particular **md**. Several flags can be or'd together. The available flags are:

**EVP\_MD\_FLAG\_ONESHOT**

This digest method can only handle one block of input.

**EVP\_MD\_FLAG\_XOF**

This digest method is an extensible-output function (XOF) and supports the **EVP\_MD\_CTRL\_XOF\_LEN** control.

**EVP\_MD\_FLAG\_DIGESTALGID\_NULL**

When setting up a DigestAlgorithmIdentifier, this flag will have the parameter set to NULL by default. Use this for PKCS#1. *Note: if combined with **EVP\_MD\_FLAG\_DIGESTALGID\_ABSENT**, the latter will override.*

**EVP\_MD\_FLAG\_DIGESTALGID\_ABSENT**

When setting up a DigestAlgorithmIdentifier, this flag will have the parameter be left absent by default. *Note: if combined with **EVP\_MD\_FLAG\_DIGESTALGID\_NULL**, the latter will be overridden.*

**EVP\_MD\_FLAG\_DIGESTALGID\_CUSTOM**

Custom DigestAlgorithmIdentifier handling via ctrl, with **EVP\_MD\_FLAG\_DIGESTALGID\_ABSENT** as default. *Note: if combined with **EVP\_MD\_FLAG\_DIGESTALGID\_NULL**, the latter will be overridden.* Currently unused.

**EVP\_MD\_FLAG\_FIPS**

This digest method is suitable for use in FIPS mode. Currently unused.

**EVP\_MD\_meth\_set\_init()** sets the digest init function for **md**. The digest init function is called by **EVP\_Digest()**, **EVP\_DigestInit()**, **EVP\_DigestInit\_ex()**, **EVP\_SignInit**, **EVP\_SignInit\_ex()**, **EVP\_VerifyInit()** and **EVP\_VerifyInit\_ex()**.

**EVP\_MD\_meth\_set\_update()** sets the digest update function for **md**. The digest update function is called by **EVP\_Digest()**, **EVP\_DigestUpdate()** and **EVP\_SignUpdate()**.

**EVP\_MD\_meth\_set\_final()** sets the digest final function for **md**. The digest final function is called by **EVP\_Digest()**, **EVP\_DigestFinal()**, **EVP\_DigestFinal\_ex()**, **EVP\_SignFinal()** and **EVP\_VerifyFinal()**.

**EVP\_MD\_meth\_set\_copy()** sets the function for **md** to do extra computations after the method's private data structure has been copied from one **EVP\_MD\_CTX** to another. If all that's needed is to copy the data, there is no need for this copy function. Note that the copy function is passed two **EVP\_MD\_CTX \***, the private data structure is then available with **EVP\_MD\_CTX\_get0\_md\_data()**. This copy function is called by **EVP\_MD\_CTX\_copy()** and **EVP\_MD\_CTX\_copy\_ex()**.

**EVP\_MD\_meth\_set\_cleanup()** sets the function for **md** to do extra cleanup before the method's private data structure is cleaned out and freed. Note that the cleanup function is passed a **EVP\_MD\_CTX \***, the private data structure is then available with **EVP\_MD\_CTX\_get0\_md\_data()**. This cleanup function is called by **EVP\_MD\_CTX\_reset()** and **EVP\_MD\_CTX\_free()**.

**EVP\_MD\_meth\_set\_ctrl()** sets the control function for **md**. See **EVP\_MD\_CTX\_ctrl(3)** for the available controls.

**EVP\_MD\_meth\_get\_input\_blocksize()**, **EVP\_MD\_meth\_get\_result\_size()**, **EVP\_MD\_meth\_get\_app\_datasize()**, **EVP\_MD\_meth\_get\_flags()**, **EVP\_MD\_meth\_get\_init()**, **EVP\_MD\_meth\_get\_update()**, **EVP\_MD\_meth\_get\_final()**, **EVP\_MD\_meth\_get\_copy()**, **EVP\_MD\_meth\_get\_cleanup()** and **EVP\_MD\_meth\_get\_ctrl()** are all used to retrieve the method data given with the **EVP\_MD\_meth\_set\_\***() functions above.

## RETURN VALUES

**EVP\_MD\_meth\_new()** and **EVP\_MD\_meth\_dup()** return a pointer to a newly created **EVP\_MD**, or NULL on failure. All **EVP\_MD\_meth\_set\_\***() functions return 1. **EVP\_MD\_get\_input\_blocksize()**, **EVP\_MD\_meth\_get\_result\_size()**, **EVP\_MD\_meth\_get\_app\_datasize()** and **EVP\_MD\_meth\_get\_flags()** return the indicated sizes or flags. All other **EVP\_CIPHER\_meth\_get\_\***() functions return pointers to their respective **md** function.

## SEE ALSO

**EVP\_DigestInit(3)**, **EVP\_SignInit(3)**, **EVP\_VerifyInit(3)**

## HISTORY

All of these functions were deprecated in OpenSSL 3.0.

The **EVP\_MD** structure was openly available in OpenSSL before version 1.1. The functions described here were added in OpenSSL 1.1. The **EVP\_MD** structure created with these functions became reference counted in OpenSSL 3.0.

## COPYRIGHT

Copyright 2015-2021 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>.