



*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 'HMAC\_CTX\_copy.3oss1' command***

***\$ man HMAC\_CTX\_copy.3oss1***

HMAC(3oss1)                      OpenSSL                      HMAC(3oss1)

### NAME

HMAC, HMAC\_CTX\_new, HMAC\_CTX\_reset, HMAC\_CTX\_free, HMAC\_Init, HMAC\_Init\_ex, HMAC\_Update, HMAC\_Final, HMAC\_CTX\_copy, HMAC\_CTX\_set\_flags, HMAC\_CTX\_get\_md, HMAC\_size - HMAC message authentication code

### SYNOPSIS

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

```
unsigned char *HMAC(const EVP_MD *evp_md, const void *key, int key_len,  
                    const unsigned char *data, size_t data_len,  
                    unsigned char *md, unsigned int *md_len);
```

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):

```
HMAC_CTX *HMAC_CTX_new(void);  
int HMAC_CTX_reset(HMAC_CTX *ctx);
```

```
int HMAC_Init_ex(HMAC_CTX *ctx, const void *key, int key_len,
```

```

    const EVP_MD *md, ENGINE *impl);
int HMAC_Update(HMAC_CTX *ctx, const unsigned char *data, size_t len);
int HMAC_Final(HMAC_CTX *ctx, unsigned char *md, unsigned int *len);

void HMAC_CTX_free(HMAC_CTX *ctx);

int HMAC_CTX_copy(HMAC_CTX *dctx, HMAC_CTX *sctx);
void HMAC_CTX_set_flags(HMAC_CTX *ctx, unsigned long flags);
const EVP_MD *HMAC_CTX_get_md(const HMAC_CTX *ctx);

size_t HMAC_size(const HMAC_CTX *e);

```

The following function has been deprecated since OpenSSL 1.1.0, and can be hidden entirely by defining `OPENSSL_API_COMPAT` with a suitable version value, see `openssl_user_macros(7)`:

```

int HMAC_Init(HMAC_CTX *ctx, const void *key, int key_len,
             const EVP_MD *md);

```

## DESCRIPTION

HMAC is a MAC (message authentication code), i.e. a keyed hash function used for message authentication, which is based on a hash function.

HMAC() computes the message authentication code of the `data_len` bytes at `data` using the hash function `evp_md` and the key `key` which is `key_len` bytes long. The key may also be NULL with `key_len` being 0.

It places the result in `md` (which must have space for the output of the hash function, which is no more than `EVP_MAX_MD_SIZE` bytes). If `md` is NULL, the digest is placed in a static array. The size of the output is placed in `md_len`, unless it is NULL. Note: passing a NULL value for `md` to use the static array is not thread safe.

evp\_md is a message digest such as EVP\_sha1(), EVP\_ripemd160() etc.

HMAC does not support variable output length digests such as

EVP\_shake128() and EVP\_shake256().

All of the functions described below are deprecated. Applications should instead use EVP\_MAC\_CTX\_new(3), EVP\_MAC\_CTX\_free(3), EVP\_MAC\_init(3), EVP\_MAC\_update(3) and EVP\_MAC\_final(3) or the 'quick' single-shot MAC function EVP\_Q\_mac(3).

HMAC\_CTX\_new() creates a new HMAC\_CTX in heap memory.

HMAC\_CTX\_reset() clears an existing HMAC\_CTX and associated resources, making it suitable for new computations as if it was newly created with HMAC\_CTX\_new().

HMAC\_CTX\_free() erases the key and other data from the HMAC\_CTX, releases any associated resources and finally frees the HMAC\_CTX itself.

The following functions may be used if the message is not completely stored in memory:

HMAC\_Init\_ex() initializes or reuses a HMAC\_CTX structure to use the hash function evp\_md and key key. If both are NULL, or if key is NULL and evp\_md is the same as the previous call, then the existing key is reused. ctx must have been created with HMAC\_CTX\_new() before the first use of an HMAC\_CTX in this function.

If HMAC\_Init\_ex() is called with key NULL and evp\_md is not the same as the previous digest used by ctx then an error is returned because reuse of an existing key with a different digest is not supported.

HMAC\_Init() initializes a HMAC\_CTX structure to use the hash function

evp\_md and the key key which is key\_len bytes long.

HMAC\_Update() can be called repeatedly with chunks of the message to be authenticated (len bytes at data).

HMAC\_Final() places the message authentication code in md, which must have space for the hash function output.

HMAC\_CTX\_copy() copies all of the internal state from sctx into dctx.

HMAC\_CTX\_set\_flags() applies the specified flags to the internal EVP\_MD\_CTXs. These flags have the same meaning as for EVP\_MD\_CTX\_set\_flags(3).

HMAC\_CTX\_get\_md() returns the EVP\_MD that has previously been set for the supplied HMAC\_CTX.

HMAC\_size() returns the length in bytes of the underlying hash function output.

## RETURN VALUES

HMAC() returns a pointer to the message authentication code or NULL if an error occurred.

HMAC\_CTX\_new() returns a pointer to a new HMAC\_CTX on success or NULL if an error occurred.

HMAC\_CTX\_reset(), HMAC\_Init\_ex(), HMAC\_Update(), HMAC\_Final() and HMAC\_CTX\_copy() return 1 for success or 0 if an error occurred.

HMAC\_CTX\_get\_md() return the EVP\_MD previously set for the supplied HMAC\_CTX or NULL if no EVP\_MD has been set.

HMAC\_size() returns the length in bytes of the underlying hash function output or zero on error.

## CONFORMING TO

RFC 2104

## SEE ALSO

SHA1(3), EVP\_Q\_mac(3), evp(7)

## HISTORY

All functions except for HMAC() were deprecated in OpenSSL 3.0.

HMAC\_CTX\_init() was replaced with HMAC\_CTX\_reset() in OpenSSL 1.1.0.

HMAC\_CTX\_cleanup() existed in OpenSSL before version 1.1.0.

HMAC\_CTX\_new(), HMAC\_CTX\_free() and HMAC\_CTX\_get\_md() are new in OpenSSL 1.1.0.

HMAC\_Init\_ex(), HMAC\_Update() and HMAC\_Final() did not return values in OpenSSL before version 1.0.0.

## COPYRIGHT

Copyright 2000-2021 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>.