



*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 'SSL\_CIPHER\_get\_kx\_nid.3ossl' command**

**\$ man SSL\_CIPHER\_get\_kx\_nid.3ossl**

SSL\_CIPHER\_GET\_NAME(3ossl)      OpenSSL      SSL\_CIPHER\_GET\_NAME(3ossl)

### NAME

SSL\_CIPHER\_get\_name, SSL\_CIPHER\_standard\_name, OPENSSL\_cipher\_name,  
SSL\_CIPHER\_get\_bits, SSL\_CIPHER\_get\_version, SSL\_CIPHER\_description,  
SSL\_CIPHER\_get\_cipher\_nid, SSL\_CIPHER\_get\_digest\_nid,  
SSL\_CIPHER\_get\_handshake\_digest, SSL\_CIPHER\_get\_kx\_nid,  
SSL\_CIPHER\_get\_auth\_nid, SSL\_CIPHER\_is\_aead, SSL\_CIPHER\_find,  
SSL\_CIPHER\_get\_id, SSL\_CIPHER\_get\_protocol\_id - get SSL\_CIPHER  
properties

### SYNOPSIS

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

const char *SSL_CIPHER_get_name(const SSL_CIPHER *cipher);
const char *SSL_CIPHER_standard_name(const SSL_CIPHER *cipher);
const char *OPENSSL_cipher_name(const char *stdname);
int SSL_CIPHER_get_bits(const SSL_CIPHER *cipher, int *alg_bits);
const char *SSL_CIPHER_get_version(const SSL_CIPHER *cipher);
char *SSL_CIPHER_description(const SSL_CIPHER *cipher, char *buf, int size);
int SSL_CIPHER_get_cipher_nid(const SSL_CIPHER *c);
int SSL_CIPHER_get_digest_nid(const SSL_CIPHER *c);
const EVP_MD *SSL_CIPHER_get_handshake_digest(const SSL_CIPHER *c);
int SSL_CIPHER_get_kx_nid(const SSL_CIPHER *c);
int SSL_CIPHER_get_auth_nid(const SSL_CIPHER *c);
int SSL_CIPHER_is_aead(const SSL_CIPHER *c);
```

```
const SSL_CIPHER *SSL_CIPHER_find(SSL *ssl, const unsigned char *ptr);
uint32_t SSL_CIPHER_get_id(const SSL_CIPHER *c);
uint32_t SSL_CIPHER_get_protocol_id(const SSL_CIPHER *c);
```

## DESCRIPTION

`SSL_CIPHER_get_name()` returns a pointer to the name of cipher. If the cipher is NULL, it returns "(NONE)".

`SSL_CIPHER_standard_name()` returns a pointer to the standard RFC name of cipher. If the cipher is NULL, it returns "(NONE)". If the cipher has no standard name, it returns NULL. If cipher was defined in both SSLv3 and TLS, it returns the TLS name.

`OPENSSL_cipher_name()` returns a pointer to the OpenSSL name of stdname. If the stdname is NULL, or stdname has no corresponding OpenSSL name, it returns "(NONE)". Where both exist, stdname should be the TLS name rather than the SSLv3 name.

`SSL_CIPHER_get_bits()` returns the number of secret bits used for cipher. If cipher is NULL, 0 is returned.

`SSL_CIPHER_get_version()` returns string which indicates the SSL/TLS protocol version that first defined the cipher. It returns "(NONE)" if cipher is NULL.

`SSL_CIPHER_get_cipher_nid()` returns the cipher NID corresponding to c. If there is no cipher (e.g. for cipher suites with no encryption) then `NID_undef` is returned.

`SSL_CIPHER_get_digest_nid()` returns the digest NID corresponding to the MAC used by c during record encryption/decryption. If there is no digest (e.g. for AEAD cipher suites) then `NID_undef` is returned.

`SSL_CIPHER_get_handshake_digest()` returns an `EVP_MD` for the digest used during the SSL/TLS handshake when using the `SSL_CIPHER` c. Note that this may be different to the digest used to calculate the MAC for encrypted records.

`SSL_CIPHER_get_kx_nid()` returns the key exchange NID corresponding to the method used by c. If there is no key exchange, then `NID_undef` is returned. If any appropriate key exchange algorithm can be used (as in the case of TLS 1.3 cipher suites) `NID_kx_any` is returned. Examples

(not comprehensive):

NID\_kx\_rsa

NID\_kx\_ecdhe

NID\_kx\_dhe

NID\_kx\_psk

SSL\_CIPHER\_get\_auth\_nid() returns the authentication NID corresponding to the method used by c. If there is no authentication, then NID\_undef is returned. If any appropriate authentication algorithm can be used (as in the case of TLS 1.3 cipher suites) NID\_auth\_any is returned.

Examples (not comprehensive):

NID\_auth\_rsa

NID\_auth\_ecdsa

NID\_auth\_psk

SSL\_CIPHER\_is\_aead() returns 1 if the cipher c is AEAD (e.g. GCM or ChaCha20/Poly1305), and 0 if it is not AEAD.

SSL\_CIPHER\_find() returns a SSL\_CIPHER structure which has the cipher ID stored in ptr. The ptr parameter is a two element array of char, which stores the two-byte TLS cipher ID (as allocated by IANA) in network byte order. This parameter is usually retrieved from a TLS packet by using functions like SSL\_client\_hello\_get0\_ciphers(3).

SSL\_CIPHER\_find() returns NULL if an error occurs or the indicated cipher is not found.

SSL\_CIPHER\_get\_id() returns the OpenSSL-specific ID of the given cipher c. That ID is not the same as the IANA-specific ID.

SSL\_CIPHER\_get\_protocol\_id() returns the two-byte ID used in the TLS protocol of the given cipher c.

SSL\_CIPHER\_description() returns a textual description of the cipher used into the buffer buf of length len provided. If buf is provided, it must be at least 128 bytes, otherwise a buffer will be allocated using OPENSSL\_malloc(). If the provided buffer is too small, or the allocation fails, NULL is returned.

The string returned by SSL\_CIPHER\_description() consists of several fields separated by whitespace:

<ciphername>

Textual representation of the cipher name.

<protocol version>

The minimum protocol version that the ciphersuite supports, such as TLSv1.2. Note that this is not always the same as the protocol version in which the ciphersuite was first defined because some ciphersuites are backwards compatible with earlier protocol versions.

Kx=<key exchange>

Key exchange method such as RSA, ECDHE, etc.

Au=<authentication>

Authentication method such as RSA, None, etc.. None is the representation of anonymous ciphers.

Enc=<symmetric encryption method>

Encryption method, with number of secret bits, such as AESGCM(128).

Mac=<message authentication code>

Message digest, such as SHA256.

Some examples for the output of SSL\_CIPHER\_description():

ECDHE-RSA-AES256-GCM-SHA256 TLSv1.2 Kx=ECDH Au=RSA Enc=AESGCM(256) Mac=AEAD

RSA-PSK-AES256-CBC-SHA384 TLSv1.0 Kx=RSAPSK Au=RSA Enc=AES(256) Mac=SHA384

## RETURN VALUES

SSL\_CIPHER\_get\_name(), SSL\_CIPHER\_standard\_name(), OPENSSL\_cipher\_name(), SSL\_CIPHER\_get\_version() and SSL\_CIPHER\_description() return the corresponding value in a NUL-terminated string for a specific cipher or "(NONE)" if the cipher is not found.

SSL\_CIPHER\_get\_bits() returns a positive integer representing the number of secret bits or 0 if an error occurred.

SSL\_CIPHER\_get\_cipher\_nid(), SSL\_CIPHER\_get\_digest\_nid(), SSL\_CIPHER\_get\_kx\_nid() and SSL\_CIPHER\_get\_auth\_nid() return the NID value or NID\_undef if an error occurred.

SSL\_CIPHER\_get\_handshake\_digest() returns a valid EVP\_MD structure or NULL if an error occurred.

SSL\_CIPHER\_is\_aead() returns 1 if the cipher is AEAD or 0 otherwise.

SSL\_CIPHER\_find() returns a valid SSL\_CIPHER structure or NULL if an error occurred.

SSL\_CIPHER\_get\_id() returns a 4-byte integer representing the OpenSSL-specific ID.

SSL\_CIPHER\_get\_protocol\_id() returns a 2-byte integer representing the TLS protocol-specific ID.

## SEE ALSO

ssl(7), SSL\_get\_current\_cipher(3), SSL\_get\_ciphers(3),  
openssl-ciphers(1)

## HISTORY

The SSL\_CIPHER\_get\_version() function was updated to always return the correct protocol string in OpenSSL 1.1.0.

The SSL\_CIPHER\_description() function was changed to return NULL on error, rather than a fixed string, in OpenSSL 1.1.0.

The SSL\_CIPHER\_get\_handshake\_digest() function was added in OpenSSL 1.1.1.

The SSL\_CIPHER\_standard\_name() function was globally available in OpenSSL 1.1.1.

Before OpenSSL 1.1.1, tracing (enable-ssl-trace argument to Configure) was required to enable this function.

The OPENSSL\_cipher\_name() function was added in OpenSSL 1.1.1.

## 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>>.

3.0.7                    2023-07-13      SSL\_CIPHER\_GET\_NAME(3ossl)