



*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 'X509V3\_add1\_i2d.3ossl' command**

**\$ man X509V3\_add1\_i2d.3ossl**

X509V3\_GET\_D2I(3ossl)      OpenSSL      X509V3\_GET\_D2I(3ossl)

### NAME

X509V3\_get\_d2i, X509V3\_add1\_i2d, X509V3\_EXT\_d2i, X509V3\_EXT\_i2d,  
X509\_get\_ext\_d2i, X509\_add1\_ext\_i2d, X509\_CRL\_get\_ext\_d2i,  
X509\_CRL\_add1\_ext\_i2d, X509\_REVOKED\_get\_ext\_d2i,  
X509\_REVOKED\_add1\_ext\_i2d, X509\_get0\_extensions,  
X509\_CRL\_get0\_extensions, X509\_REVOKED\_get0\_extensions - X509 extension  
decode and encode functions

### SYNOPSIS

```
#include <openssl/x509v3.h>

void *X509V3_get_d2i(const STACK_OF(X509_EXTENSION) *x, int nid, int *crit,
    int *idx);

int X509V3_add1_i2d(STACK_OF(X509_EXTENSION) **x, int nid, void *value,
    int crit, unsigned long flags);

void *X509V3_EXT_d2i(X509_EXTENSION *ext);

X509_EXTENSION *X509V3_EXT_i2d(int ext_nid, int crit, void *ext_struct);

void *X509_get_ext_d2i(const X509 *x, int nid, int *crit, int *idx);

int X509_add1_ext_i2d(X509 *x, int nid, void *value, int crit,
```

unsigned long flags);

```
void *X509_CRL_get_ext_d2i(const X509_CRL *crl, int nid, int *crit, int *idx);
```

```
int X509_CRL_add1_ext_i2d(X509_CRL *crl, int nid, void *value, int crit,  
    unsigned long flags);
```

```
void *X509_REVOKED_get_ext_d2i(const X509_REVOKED *r, int nid, int *crit, int *idx);
```

```
int X509_REVOKED_add1_ext_i2d(X509_REVOKED *r, int nid, void *value, int crit,  
    unsigned long flags);
```

```
const STACK_OF(X509_EXTENSION) *X509_get0_extensions(const X509 *x);
```

```
const STACK_OF(X509_EXTENSION) *X509_CRL_get0_extensions(const X509_CRL *crl);
```

```
const STACK_OF(X509_EXTENSION) *X509_REVOKED_get0_extensions(const X509_REVOKED *r);
```

## DESCRIPTION

X509V3\_get\_d2i() looks for an extension with OID nid in the extensions x and, if found, decodes it. If idx is NULL then only one occurrence of an extension is permissible, otherwise the first extension after index \*idx is returned and \*idx updated to the location of the extension. If crit is not NULL then \*crit is set to a status value: -2 if the extension occurs multiple times (this is only returned if idx is NULL), -1 if the extension could not be found, 0 if the extension is found and is not critical and 1 if critical. A pointer to an extension specific structure or NULL is returned.

X509V3\_add1\_i2d() adds extension value to STACK \*x (allocating a new STACK if necessary) using OID nid and criticality crit according to flags.

X509V3\_EXT\_d2i() attempts to decode the ASN.1 data contained in extension ext and returns a pointer to an extension specific structure or NULL if the extension could not be decoded (invalid syntax or not supported).

X509V3\_EXT\_i2d() encodes the extension specific structure ext\_struct with OID ext\_nid and criticality crit.

X509\_get\_ext\_d2i() and X509\_add1\_ext\_i2d() operate on the extensions of certificate x. They are otherwise identical to X509V3\_get\_d2i() and X509V3\_add1\_i2d().

X509\_CRL\_get\_ext\_d2i() and X509\_CRL\_add1\_ext\_i2d() operate on the extensions of CRL crl. They are otherwise identical to X509V3\_get\_d2i() and X509V3\_add1\_i2d().

X509\_REVOKED\_get\_ext\_d2i() and X509\_REVOKED\_add1\_ext\_i2d() operate on the extensions of X509\_REVOKED structure r (i.e for CRL entry extensions). They are otherwise identical to X509V3\_get\_d2i() and X509V3\_add1\_i2d().

X509\_get0\_extensions(), X509\_CRL\_get0\_extensions() and X509\_REVOKED\_get0\_extensions() return a STACK of all the extensions of a certificate, a CRL or a CRL entry respectively.

## NOTES

In almost all cases an extension can occur at most once and multiple occurrences is an error. Therefore, the idx parameter is usually NULL.

The flags parameter may be one of the following values.

X509V3\_ADD\_DEFAULT appends a new extension only if the extension does not exist. An error is returned if the extension exists.

X509V3\_ADD\_APPEND appends a new extension, ignoring whether the extension exists.

X509V3\_ADD\_REPLACE replaces an existing extension. If the extension does not exist, appends a new extension.

X509V3\_ADD\_REPLACE\_EXISTING replaces an existing extension. If the extension does not exist, returns an error.

X509V3\_ADD\_KEEP\_EXISTING appends a new extension only if the extension does not exist. An error is not returned if the extension exists.

X509V3\_ADD\_DELETE deletes and frees an existing extension. If the extension does not exist, returns an error. No new extension is added.

If X509V3\_ADD\_SILENT is bitwise ORed with flags: any error returned will not be added to the error queue.

The function X509V3\_get\_d2i() and its variants will return NULL if the extension is not found, occurs multiple times or cannot be decoded. It is possible to determine the precise reason by checking the value of \*crit.

The function X509V3\_add1\_i2d() and its variants allocate X509\_EXTENSION objects on STACK \*x depending on flags. The X509\_EXTENSION objects must be explicitly freed using X509\_EXTENSION\_free().

## SUPPORTED EXTENSIONS

The following sections contain a list of all supported extensions including their name and NID.

### PKIX Certificate Extensions

The following certificate extensions are defined in PKIX standards such as RFC5280.

Key Usage	NID_key_usage
Extended Key Usage	NID_ext_key_usage
Subject Key Identifier	NID_subject_key_identifier
Authority Key Identifier	NID_authority_key_identifier
Private Key Usage Period	NID_private_key_usage_period
Subject Alternative Name	NID_subject_alt_name
Issuer Alternative Name	NID_issuer_alt_name
Authority Information Access	NID_info_access
Subject Information Access	NID_sinfo_access
Name Constraints	NID_name_constraints
Certificate Policies	NID_certificate_policies
Policy Mappings	NID_policy_mappings
Policy Constraints	NID_policy_constraints
Inhibit Any Policy	NID_inhibit_any_policy
TLS Feature	NID_tlsfeature

#### Netscape Certificate Extensions

The following are (largely obsolete) Netscape certificate extensions.

Netscape Cert Type	NID_netscape_cert_type
Netscape Base Url	NID_netscape_base_url
Netscape Revocation Url	NID_netscape_revocation_url
Netscape CA Revocation Url	NID_netscape_ca_revocation_url
Netscape Renewal Url	NID_netscape_renewal_url
Netscape CA Policy Url	NID_netscape_ca_policy_url
Netscape SSL Server Name	NID_netscape_ssl_server_name

Netscape Comment            NID\_netscape\_comment

#### Miscellaneous Certificate Extensions

Strong Extranet ID            NID\_sxnet

Proxy Certificate Information    NID\_proxyCertInfo

#### PKIX CRL Extensions

The following are CRL extensions from PKIX standards such as RFC5280.

CRL Number                    NID\_crl\_number

CRL Distribution Points        NID\_crl\_distribution\_points

Delta CRL Indicator          NID\_delta\_crl

Freshest CRL                  NID\_freshest\_crl

Invalidity Date                NID\_invalid\_date

Issuing Distribution Point     NID\_issuing\_distribution\_point

The following are CRL entry extensions from PKIX standards such as RFC5280.

CRL Reason Code                NID\_crl\_reason

Certificate Issuer              NID\_certificate\_issuer

#### OCSP Extensions

OCSP Nonce                    NID\_id\_pkix\_OCSP\_Nonce

OCSP CRL ID                    NID\_id\_pkix\_OCSP\_CrIID

Acceptable OCSP Responses     NID\_id\_pkix\_OCSP\_acceptableResponses

OCSP No Check                 NID\_id\_pkix\_OCSP\_noCheck

OCSP Archive Cutoff            NID\_id\_pkix\_OCSP\_archiveCutoff

OCSP Service Locator          NID\_id\_pkix\_OCSP\_serviceLocator

Hold Instruction Code          NID\_hold\_instruction\_code

#### Certificate Transparency Extensions

The following extensions are used by certificate transparency, RFC6962

CT Precertificate SCTs          NID\_ct\_precert\_scts

CT Certificate SCTs            NID\_ct\_cert\_scts

## RETURN VALUES

X509V3\_get\_d2i(), its variants, and X509V3\_EXT\_d2i() return a pointer to an extension specific structure or NULL if an error occurs.

X509V3\_add1\_i2d() and its variants return 1 if the operation is successful and 0 if it fails due to a non-fatal error (extension not found, already exists, cannot be encoded) or -1 due to a fatal error such as a memory allocation failure.

X509V3\_EXT\_i2d() returns a pointer to an X509\_EXTENSION structure or NULL if an error occurs.

X509\_get0\_extensions(), X509\_CRL\_get0\_extensions() and X509\_REVOKED\_get0\_extensions() return a stack of extensions. They return NULL if no extensions are present.

## SEE ALSO

d2i\_X509(3), ERR\_get\_error(3), X509\_CRL\_get0\_by\_serial(3),  
X509\_get0\_signature(3), X509\_get\_ext\_d2i(3),  
X509\_get\_extension\_flags(3), X509\_get\_pubkey(3),  
X509\_get\_subject\_name(3), X509\_get\_version(3),  
X509\_NAME\_add\_entry\_by\_txt(3), X509\_NAME\_ENTRY\_get\_object(3),  
X509\_NAME\_get\_index\_by\_NID(3), X509\_NAME\_print\_ex(3), X509\_new(3),  
X509\_sign(3), X509\_verify\_cert(3)

## COPYRIGHT

Copyright 2015-2022 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            X509V3\_GET\_D2I(3oss)