

NAME

X509_get0_subject_key_id, X509_get0_authority_key_id, X509_get0_authority_issuer, X509_get0_authority_serial, X509_get_pathlen, X509_get_extension_flags, X509_get_key_usage, X509_get_extended_key_usage, X509_set_proxy_flag, X509_set_proxy_pathlen, X509_get_proxy_pathlen - retrieve certificate extension data

SYNOPSIS

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

```
long X509_get_pathlen(X509 *x);
uint32_t X509_get_extension_flags(X509 *x);
uint32_t X509_get_key_usage(X509 *x);
uint32_t X509_get_extended_key_usage(X509 *x);
const ASN1_OCTET_STRING *X509_get0_subject_key_id(X509 *x);
const ASN1_OCTET_STRING *X509_get0_authority_key_id(X509 *x);
const GENERAL_NAMES *X509_get0_authority_issuer(X509 *x);
const ASN1_INTEGER *X509_get0_authority_serial(X509 *x);
void X509_set_proxy_flag(X509 *x);
void X509_set_proxy_pathlen(int l);
long X509_get_proxy_pathlen(X509 *x);
```

DESCRIPTION

These functions retrieve information related to commonly used certificate extensions.

X509_get_pathlen() retrieves the path length extension from a certificate. This extension is used to limit the length of a cert chain that may be issued from that CA.

X509_get_extension_flags() retrieves general information about a certificate, it will return one or more of the following flags ored together.

EXFLAG_V1

The certificate is an obsolete version 1 certificate.

EXFLAG_BCONS

The certificate contains a basic constraints extension.

EXFLAG_CA

The certificate contains basic constraints and asserts the CA flag.

EXFLAG_PROXY

The certificate is a valid proxy certificate.

EXFLAG_SI

The certificate is self issued (that is subject and issuer names match).

EXFLAG_SS

The subject and issuer names match and extension values imply it is self signed.

EXFLAG_FRESHEST

The freshest CRL extension is present in the certificate.

EXFLAG_CRITICAL

The certificate contains an unhandled critical extension.

EXFLAG_INVALID

Some certificate extension values are invalid or inconsistent. The certificate should be rejected. This bit may also be raised after an out-of-memory error while processing the X509 object, so it may not be related to the processed ASN1 object itself.

EXFLAG_NO_FINGERPRINT

Failed to compute the internal SHA1 hash value of the certificate or CRL. This may be due to malloc failure or because no SHA1 implementation was found.

EXFLAG_INVALID_POLICY

The NID_certificate_policies certificate extension is invalid or inconsistent. The certificate should be rejected. This bit may also be raised after an out-of-memory error while processing the X509 object, so it may not be related to the processed ASN1 object itself.

EXFLAG_KUSAGE

The certificate contains a key usage extension. The value can be retrieved using `X509_get_key_usage()`.

EXFLAG_XKUSAGE

The certificate contains an extended key usage extension. The value can be retrieved using `X509_get_extended_key_usage()`.

`X509_get_key_usage()` returns the value of the key usage extension. If key usage is present will return zero or more of the flags: **KU_DIGITAL_SIGNATURE**, **KU_NON_REPUDIATION**, **KU_KEY_ENCIPHERMENT**, **KU_DATA_ENCIPHERMENT**, **KU_KEY_AGREEMENT**, **KU_KEY_CERT_SIGN**, **KU_CRL_SIGN**, **KU_ENCIPHER_ONLY** or **KU_DECIPHER_ONLY**

corresponding to individual key usage bits. If key usage is absent then **UINT32_MAX** is returned.

X509_get_extended_key_usage() returns the value of the extended key usage extension. If extended key usage is present it will return zero or more of the flags: **XKU_SSL_SERVER**, **XKU_SSL_CLIENT**, **XKU_SMIME**, **XKU_CODE_SIGN**, **XKU_OCSP_SIGN**, **XKU_TIMESTAMP**, **XKU_DVCS** or **XKU_ANYEKU**. These correspond to the OIDs **id-kp-serverAuth**, **id-kp-clientAuth**, **id-kp-emailProtection**, **id-kp-codeSigning**, **id-kp-OCSPSigning**, **id-kp-timeStamping**, **id-kp-dvcs** and **anyExtendedKeyUsage** respectively. Additionally **XKU_SGC** is set if either Netscape or Microsoft SGC OIDs are present.

X509_get0_subject_key_id() returns an internal pointer to the subject key identifier of **x** as an **ASN1_OCTET_STRING** or **NULL** if the extension is not present or cannot be parsed.

X509_get0_authority_key_id() returns an internal pointer to the authority key identifier of **x** as an **ASN1_OCTET_STRING** or **NULL** if the extension is not present or cannot be parsed.

X509_get0_authority_issuer() returns an internal pointer to the authority certificate issuer of **x** as a stack of **GENERAL_NAME** structures or **NULL** if the extension is not present or cannot be parsed.

X509_get0_authority_serial() returns an internal pointer to the authority certificate serial number of **x** as an **ASN1_INTEGER** or **NULL** if the extension is not present or cannot be parsed.

X509_set_proxy_flag() marks the certificate with the **EXFLAG_PROXY** flag. This is for the users who need to mark non-RFC3820 proxy certificates as such, as OpenSSL only detects RFC3820 compliant ones.

X509_set_proxy_pathlen() sets the proxy certificate path length for the given certificate **x**. This is for the users who need to mark non-RFC3820 proxy certificates as such, as OpenSSL only detects RFC3820 compliant ones.

X509_get_proxy_pathlen() returns the proxy certificate path length for the given certificate **x** if it is a proxy certificate.

NOTES

The value of the flags correspond to extension values which are cached in the **X509** structure. If the flags returned do not provide sufficient information an application should examine extension values directly for example using **X509_get_ext_d2i()**.

If the key usage or extended key usage extension is absent then typically usage is unrestricted. For this reason **X509_get_key_usage()** and **X509_get_extended_key_usage()** return **UINT32_MAX** when the

corresponding extension is absent. Applications can additionally check the return value of **X509_get_extension_flags()** and take appropriate action if an extension is absent.

If **X509_get0_subject_key_id()** returns **NULL** then the extension may be absent or malformed. Applications can determine the precise reason using **X509_get_ext_d2i()**.

RETURN VALUES

X509_get_pathlen() returns the path length value, or -1 if the extension is not present.

X509_get_extension_flags(), **X509_get_key_usage()** and **X509_get_extended_key_usage()** return sets of flags corresponding to the certificate extension values.

X509_get0_subject_key_id() returns the subject key identifier as a pointer to an **ASN1_OCTET_STRING** structure or **NULL** if the extension is absent or an error occurred during parsing.

X509_get_proxy_pathlen() returns the path length value if the given certificate is a proxy one and has a path length set, and -1 otherwise.

SEE ALSO

X509_check_purpose(3)

HISTORY

X509_get_pathlen(), **X509_set_proxy_flag()**, **X509_set_proxy_pathlen()** and **X509_get_proxy_pathlen()** were added in OpenSSL 1.1.0.

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>>.