| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/X509Certificate.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/security/cert/TrustAnchor.html)   [**NEXT CLASS**](http://docs.google.com/java/security/cert/X509CertSelector.html) | [**FRAMES**](http://docs.google.com/index.html?java/security/cert/X509Certificate.html)    [**NO FRAMES**](http://docs.google.com/X509Certificate.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | FIELD | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#26in1rg) |

## **java.security.cert**

Class X509Certificate

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.security.cert.Certificate](http://docs.google.com/java/security/cert/Certificate.html)  
 **java.security.cert.X509Certificate**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [X509Extension](http://docs.google.com/java/security/cert/X509Extension.html)

public abstract class **X509Certificate**extends [Certificate](http://docs.google.com/java/security/cert/Certificate.html)implements [X509Extension](http://docs.google.com/java/security/cert/X509Extension.html)

Abstract class for X.509 certificates. This provides a standard way to access all the attributes of an X.509 certificate.

In June of 1996, the basic X.509 v3 format was completed by ISO/IEC and ANSI X9, which is described below in ASN.1:

Certificate ::= SEQUENCE {  
 tbsCertificate TBSCertificate,  
 signatureAlgorithm AlgorithmIdentifier,  
 signature BIT STRING }

These certificates are widely used to support authentication and other functionality in Internet security systems. Common applications include Privacy Enhanced Mail (PEM), Transport Layer Security (SSL), code signing for trusted software distribution, and Secure Electronic Transactions (SET).

These certificates are managed and vouched for by *Certificate Authorities* (CAs). CAs are services which create certificates by placing data in the X.509 standard format and then digitally signing that data. CAs act as trusted third parties, making introductions between principals who have no direct knowledge of each other. CA certificates are either signed by themselves, or by some other CA such as a "root" CA.

More information can be found in RFC 2459, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile" at <http://www.ietf.org/rfc/rfc2459.txt>.

The ASN.1 definition of tbsCertificate is:

TBSCertificate ::= SEQUENCE {  
 version [0] EXPLICIT Version DEFAULT v1,  
 serialNumber CertificateSerialNumber,  
 signature AlgorithmIdentifier,  
 issuer Name,  
 validity Validity,  
 subject Name,  
 subjectPublicKeyInfo SubjectPublicKeyInfo,  
 issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,  
 -- If present, version must be v2 or v3  
 subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,  
 -- If present, version must be v2 or v3  
 extensions [3] EXPLICIT Extensions OPTIONAL  
 -- If present, version must be v3  
 }

Certificates are instantiated using a certificate factory. The following is an example of how to instantiate an X.509 certificate:

InputStream inStream = new FileInputStream("fileName-of-cert");  
 CertificateFactory cf = CertificateFactory.getInstance("X.509");  
 X509Certificate cert = (X509Certificate)cf.generateCertificate(inStream);  
 inStream.close();

**See Also:**[Certificate](http://docs.google.com/java/security/cert/Certificate.html), [CertificateFactory](http://docs.google.com/java/security/cert/CertificateFactory.html), [X509Extension](http://docs.google.com/java/security/cert/X509Extension.html), [Serialized Form](http://docs.google.com/serialized-form.html#java.security.cert.X509Certificate)

| **Nested Class Summary** | |
| --- | --- |

| **Nested classes/interfaces inherited from class java.security.cert.**[**Certificate**](http://docs.google.com/java/security/cert/Certificate.html) |
| --- |
| [Certificate.CertificateRep](http://docs.google.com/java/security/cert/Certificate.CertificateRep.html) |

| **Constructor Summary** | |
| --- | --- |
| protected | [**X509Certificate**](http://docs.google.com/java/security/cert/X509Certificate.html#X509Certificate())()            Constructor for X.509 certificates. |

| **Method Summary** | |
| --- | --- |
| abstract  void | [**checkValidity**](http://docs.google.com/java/security/cert/X509Certificate.html#checkValidity())()            Checks that the certificate is currently valid. |
| abstract  void | [**checkValidity**](http://docs.google.com/java/security/cert/X509Certificate.html#checkValidity(java.util.Date))([Date](http://docs.google.com/java/util/Date.html) date)            Checks that the given date is within the certificate's validity period. |
| abstract  int | [**getBasicConstraints**](http://docs.google.com/java/security/cert/X509Certificate.html#getBasicConstraints())()            Gets the certificate constraints path length from the critical BasicConstraints extension, (OID = 2.5.29.19). |
| [List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)> | [**getExtendedKeyUsage**](http://docs.google.com/java/security/cert/X509Certificate.html#getExtendedKeyUsage())()            Gets an unmodifiable list of Strings representing the OBJECT IDENTIFIERs of the ExtKeyUsageSyntax field of the extended key usage extension, (OID = 2.5.29.37). |
| [Collection](http://docs.google.com/java/util/Collection.html)<[List](http://docs.google.com/java/util/List.html)<?>> | [**getIssuerAlternativeNames**](http://docs.google.com/java/security/cert/X509Certificate.html#getIssuerAlternativeNames())()            Gets an immutable collection of issuer alternative names from the IssuerAltName extension, (OID = 2.5.29.18). |
| abstract  [Principal](http://docs.google.com/java/security/Principal.html) | [**getIssuerDN**](http://docs.google.com/java/security/cert/X509Certificate.html#getIssuerDN())()  **Denigrated**, replaced by [getIssuerX500Principal()](http://docs.google.com/java/security/cert/X509Certificate.html#getIssuerX500Principal()). |
| abstract  boolean[] | [**getIssuerUniqueID**](http://docs.google.com/java/security/cert/X509Certificate.html#getIssuerUniqueID())()            Gets the issuerUniqueID value from the certificate. |
| [X500Principal](http://docs.google.com/javax/security/auth/x500/X500Principal.html) | [**getIssuerX500Principal**](http://docs.google.com/java/security/cert/X509Certificate.html#getIssuerX500Principal())()            Returns the issuer (issuer distinguished name) value from the certificate as an X500Principal. |
| abstract  boolean[] | [**getKeyUsage**](http://docs.google.com/java/security/cert/X509Certificate.html#getKeyUsage())()            Gets a boolean array representing bits of the KeyUsage extension, (OID = 2.5.29.15). |
| abstract  [Date](http://docs.google.com/java/util/Date.html) | [**getNotAfter**](http://docs.google.com/java/security/cert/X509Certificate.html#getNotAfter())()            Gets the notAfter date from the validity period of the certificate. |
| abstract  [Date](http://docs.google.com/java/util/Date.html) | [**getNotBefore**](http://docs.google.com/java/security/cert/X509Certificate.html#getNotBefore())()            Gets the notBefore date from the validity period of the certificate. |
| abstract  [BigInteger](http://docs.google.com/java/math/BigInteger.html) | [**getSerialNumber**](http://docs.google.com/java/security/cert/X509Certificate.html#getSerialNumber())()            Gets the serialNumber value from the certificate. |
| abstract  [String](http://docs.google.com/java/lang/String.html) | [**getSigAlgName**](http://docs.google.com/java/security/cert/X509Certificate.html#getSigAlgName())()            Gets the signature algorithm name for the certificate signature algorithm. |
| abstract  [String](http://docs.google.com/java/lang/String.html) | [**getSigAlgOID**](http://docs.google.com/java/security/cert/X509Certificate.html#getSigAlgOID())()            Gets the signature algorithm OID string from the certificate. |
| abstract  byte[] | [**getSigAlgParams**](http://docs.google.com/java/security/cert/X509Certificate.html#getSigAlgParams())()            Gets the DER-encoded signature algorithm parameters from this certificate's signature algorithm. |
| abstract  byte[] | [**getSignature**](http://docs.google.com/java/security/cert/X509Certificate.html#getSignature())()            Gets the signature value (the raw signature bits) from the certificate. |
| [Collection](http://docs.google.com/java/util/Collection.html)<[List](http://docs.google.com/java/util/List.html)<?>> | [**getSubjectAlternativeNames**](http://docs.google.com/java/security/cert/X509Certificate.html#getSubjectAlternativeNames())()            Gets an immutable collection of subject alternative names from the SubjectAltName extension, (OID = 2.5.29.17). |
| abstract  [Principal](http://docs.google.com/java/security/Principal.html) | [**getSubjectDN**](http://docs.google.com/java/security/cert/X509Certificate.html#getSubjectDN())()  **Denigrated**, replaced by [getSubjectX500Principal()](http://docs.google.com/java/security/cert/X509Certificate.html#getSubjectX500Principal()). |
| abstract  boolean[] | [**getSubjectUniqueID**](http://docs.google.com/java/security/cert/X509Certificate.html#getSubjectUniqueID())()            Gets the subjectUniqueID value from the certificate. |
| [X500Principal](http://docs.google.com/javax/security/auth/x500/X500Principal.html) | [**getSubjectX500Principal**](http://docs.google.com/java/security/cert/X509Certificate.html#getSubjectX500Principal())()            Returns the subject (subject distinguished name) value from the certificate as an X500Principal. |
| abstract  byte[] | [**getTBSCertificate**](http://docs.google.com/java/security/cert/X509Certificate.html#getTBSCertificate())()            Gets the DER-encoded certificate information, the tbsCertificate from this certificate. |
| abstract  int | [**getVersion**](http://docs.google.com/java/security/cert/X509Certificate.html#getVersion())()            Gets the version (version number) value from the certificate. |

| **Methods inherited from class java.security.cert.**[**Certificate**](http://docs.google.com/java/security/cert/Certificate.html) |
| --- |
| [equals](http://docs.google.com/java/security/cert/Certificate.html#equals(java.lang.Object)), [getEncoded](http://docs.google.com/java/security/cert/Certificate.html#getEncoded()), [getPublicKey](http://docs.google.com/java/security/cert/Certificate.html#getPublicKey()), [getType](http://docs.google.com/java/security/cert/Certificate.html#getType()), [hashCode](http://docs.google.com/java/security/cert/Certificate.html#hashCode()), [toString](http://docs.google.com/java/security/cert/Certificate.html#toString()), [verify](http://docs.google.com/java/security/cert/Certificate.html#verify(java.security.PublicKey)), [verify](http://docs.google.com/java/security/cert/Certificate.html#verify(java.security.PublicKey,%20java.lang.String)), [writeReplace](http://docs.google.com/java/security/cert/Certificate.html#writeReplace()) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Methods inherited from interface java.security.cert.**[**X509Extension**](http://docs.google.com/java/security/cert/X509Extension.html) |
| --- |
| [getCriticalExtensionOIDs](http://docs.google.com/java/security/cert/X509Extension.html#getCriticalExtensionOIDs()), [getExtensionValue](http://docs.google.com/java/security/cert/X509Extension.html#getExtensionValue(java.lang.String)), [getNonCriticalExtensionOIDs](http://docs.google.com/java/security/cert/X509Extension.html#getNonCriticalExtensionOIDs()), [hasUnsupportedCriticalExtension](http://docs.google.com/java/security/cert/X509Extension.html#hasUnsupportedCriticalExtension()) |

| **Constructor Detail** |
| --- |

### X509Certificate

protected **X509Certificate**()

Constructor for X.509 certificates.

| **Method Detail** |
| --- |

### checkValidity

public abstract void **checkValidity**()  
 throws [CertificateExpiredException](http://docs.google.com/java/security/cert/CertificateExpiredException.html),  
 [CertificateNotYetValidException](http://docs.google.com/java/security/cert/CertificateNotYetValidException.html)

Checks that the certificate is currently valid. It is if the current date and time are within the validity period given in the certificate.

The validity period consists of two date/time values: the first and last dates (and times) on which the certificate is valid. It is defined in ASN.1 as:

validity Validity

Validity ::= SEQUENCE {  
 notBefore CertificateValidityDate,  
 notAfter CertificateValidityDate }

CertificateValidityDate ::= CHOICE {  
 utcTime UTCTime,  
 generalTime GeneralizedTime }

**Throws:** [CertificateExpiredException](http://docs.google.com/java/security/cert/CertificateExpiredException.html) - if the certificate has expired. [CertificateNotYetValidException](http://docs.google.com/java/security/cert/CertificateNotYetValidException.html) - if the certificate is not yet valid.

### checkValidity

public abstract void **checkValidity**([Date](http://docs.google.com/java/util/Date.html) date)  
 throws [CertificateExpiredException](http://docs.google.com/java/security/cert/CertificateExpiredException.html),  
 [CertificateNotYetValidException](http://docs.google.com/java/security/cert/CertificateNotYetValidException.html)

Checks that the given date is within the certificate's validity period. In other words, this determines whether the certificate would be valid at the given date/time.

**Parameters:**date - the Date to check against to see if this certificate is valid at that date/time. **Throws:** [CertificateExpiredException](http://docs.google.com/java/security/cert/CertificateExpiredException.html) - if the certificate has expired with respect to the date supplied. [CertificateNotYetValidException](http://docs.google.com/java/security/cert/CertificateNotYetValidException.html) - if the certificate is not yet valid with respect to the date supplied.**See Also:**[checkValidity()](http://docs.google.com/java/security/cert/X509Certificate.html#checkValidity())

### getVersion

public abstract int **getVersion**()

Gets the version (version number) value from the certificate. The ASN.1 definition for this is:

version [0] EXPLICIT Version DEFAULT v1

Version ::= INTEGER { v1(0), v2(1), v3(2) }

**Returns:**the version number, i.e. 1, 2 or 3.

### getSerialNumber

public abstract [BigInteger](http://docs.google.com/java/math/BigInteger.html) **getSerialNumber**()

Gets the serialNumber value from the certificate. The serial number is an integer assigned by the certification authority to each certificate. It must be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate). The ASN.1 definition for this is:

serialNumber CertificateSerialNumber

CertificateSerialNumber ::= INTEGER

**Returns:**the serial number.

### getIssuerDN

public abstract [Principal](http://docs.google.com/java/security/Principal.html) **getIssuerDN**()

**Denigrated**, replaced by [getIssuerX500Principal()](http://docs.google.com/java/security/cert/X509Certificate.html#getIssuerX500Principal()). This method returns the issuer as an implementation specific Principal object, which should not be relied upon by portable code.

Gets the issuer (issuer distinguished name) value from the certificate. The issuer name identifies the entity that signed (and issued) the certificate.

The issuer name field contains an X.500 distinguished name (DN). The ASN.1 definition for this is:

issuer Name

Name ::= CHOICE { RDNSequence }  
 RDNSequence ::= SEQUENCE OF RelativeDistinguishedName  
 RelativeDistinguishedName ::=  
 SET OF AttributeValueAssertion  
  
 AttributeValueAssertion ::= SEQUENCE {  
 AttributeType,  
 AttributeValue }  
 AttributeType ::= OBJECT IDENTIFIER  
 AttributeValue ::= ANY

The Name describes a hierarchical name composed of attributes, such as country name, and corresponding values, such as US. The type of the AttributeValue component is determined by the AttributeType; in general it will be a directoryString. A directoryString is usually one of PrintableString, TeletexString or UniversalString.

**Returns:**a Principal whose name is the issuer distinguished name.

### getIssuerX500Principal

public [X500Principal](http://docs.google.com/javax/security/auth/x500/X500Principal.html) **getIssuerX500Principal**()

Returns the issuer (issuer distinguished name) value from the certificate as an X500Principal.

It is recommended that subclasses override this method.

**Returns:**an X500Principal representing the issuer distinguished name**Since:** 1.4

### getSubjectDN

public abstract [Principal](http://docs.google.com/java/security/Principal.html) **getSubjectDN**()

**Denigrated**, replaced by [getSubjectX500Principal()](http://docs.google.com/java/security/cert/X509Certificate.html#getSubjectX500Principal()). This method returns the subject as an implementation specific Principal object, which should not be relied upon by portable code.

Gets the subject (subject distinguished name) value from the certificate. If the subject value is empty, then the getName() method of the returned Principal object returns an empty string ("").

The ASN.1 definition for this is:

subject Name

See [getIssuerDN](http://docs.google.com/java/security/cert/X509Certificate.html#getIssuerDN()) for Name and other relevant definitions.

**Returns:**a Principal whose name is the subject name.

### getSubjectX500Principal

public [X500Principal](http://docs.google.com/javax/security/auth/x500/X500Principal.html) **getSubjectX500Principal**()

Returns the subject (subject distinguished name) value from the certificate as an X500Principal. If the subject value is empty, then the getName() method of the returned X500Principal object returns an empty string ("").

It is recommended that subclasses override this method.

**Returns:**an X500Principal representing the subject distinguished name**Since:** 1.4

### getNotBefore

public abstract [Date](http://docs.google.com/java/util/Date.html) **getNotBefore**()

Gets the notBefore date from the validity period of the certificate. The relevant ASN.1 definitions are:

validity Validity

Validity ::= SEQUENCE {  
 notBefore CertificateValidityDate,  
 notAfter CertificateValidityDate }

CertificateValidityDate ::= CHOICE {  
 utcTime UTCTime,  
 generalTime GeneralizedTime }

**Returns:**the start date of the validity period.**See Also:**[checkValidity()](http://docs.google.com/java/security/cert/X509Certificate.html#checkValidity())

### getNotAfter

public abstract [Date](http://docs.google.com/java/util/Date.html) **getNotAfter**()

Gets the notAfter date from the validity period of the certificate. See [getNotBefore](http://docs.google.com/java/security/cert/X509Certificate.html#getNotBefore()) for relevant ASN.1 definitions.

**Returns:**the end date of the validity period.**See Also:**[checkValidity()](http://docs.google.com/java/security/cert/X509Certificate.html#checkValidity())

### getTBSCertificate

public abstract byte[] **getTBSCertificate**()  
 throws [CertificateEncodingException](http://docs.google.com/java/security/cert/CertificateEncodingException.html)

Gets the DER-encoded certificate information, the tbsCertificate from this certificate. This can be used to verify the signature independently.

**Returns:**the DER-encoded certificate information. **Throws:** [CertificateEncodingException](http://docs.google.com/java/security/cert/CertificateEncodingException.html) - if an encoding error occurs.

### getSignature

public abstract byte[] **getSignature**()

Gets the signature value (the raw signature bits) from the certificate. The ASN.1 definition for this is:

signature BIT STRING

**Returns:**the signature.

### getSigAlgName

public abstract [String](http://docs.google.com/java/lang/String.html) **getSigAlgName**()

Gets the signature algorithm name for the certificate signature algorithm. An example is the string "SHA-1/DSA". The ASN.1 definition for this is:

signatureAlgorithm AlgorithmIdentifier

AlgorithmIdentifier ::= SEQUENCE {  
 algorithm OBJECT IDENTIFIER,  
 parameters ANY DEFINED BY algorithm OPTIONAL }  
 -- contains a value of the type  
 -- registered for use with the  
 -- algorithm object identifier value

The algorithm name is determined from the algorithm OID string.

**Returns:**the signature algorithm name.

### getSigAlgOID

public abstract [String](http://docs.google.com/java/lang/String.html) **getSigAlgOID**()

Gets the signature algorithm OID string from the certificate. An OID is represented by a set of nonnegative whole numbers separated by periods. For example, the string "1.2.840.10040.4.3" identifies the SHA-1 with DSA signature algorithm, as per RFC 2459.

See [getSigAlgName](http://docs.google.com/java/security/cert/X509Certificate.html#getSigAlgName()) for relevant ASN.1 definitions.

**Returns:**the signature algorithm OID string.

### getSigAlgParams

public abstract byte[] **getSigAlgParams**()

Gets the DER-encoded signature algorithm parameters from this certificate's signature algorithm. In most cases, the signature algorithm parameters are null; the parameters are usually supplied with the certificate's public key. If access to individual parameter values is needed then use [AlgorithmParameters](http://docs.google.com/java/security/AlgorithmParameters.html) and instantiate with the name returned by [getSigAlgName](http://docs.google.com/java/security/cert/X509Certificate.html#getSigAlgName()).

See [getSigAlgName](http://docs.google.com/java/security/cert/X509Certificate.html#getSigAlgName()) for relevant ASN.1 definitions.

**Returns:**the DER-encoded signature algorithm parameters, or null if no parameters are present.

### getIssuerUniqueID

public abstract boolean[] **getIssuerUniqueID**()

Gets the issuerUniqueID value from the certificate. The issuer unique identifier is present in the certificate to handle the possibility of reuse of issuer names over time. RFC 2459 recommends that names not be reused and that conforming certificates not make use of unique identifiers. Applications conforming to that profile should be capable of parsing unique identifiers and making comparisons.

The ASN.1 definition for this is:

issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL

UniqueIdentifier ::= BIT STRING

**Returns:**the issuer unique identifier or null if it is not present in the certificate.

### getSubjectUniqueID

public abstract boolean[] **getSubjectUniqueID**()

Gets the subjectUniqueID value from the certificate.

The ASN.1 definition for this is:

subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL

UniqueIdentifier ::= BIT STRING

**Returns:**the subject unique identifier or null if it is not present in the certificate.

### getKeyUsage

public abstract boolean[] **getKeyUsage**()

Gets a boolean array representing bits of the KeyUsage extension, (OID = 2.5.29.15). The key usage extension defines the purpose (e.g., encipherment, signature, certificate signing) of the key contained in the certificate. The ASN.1 definition for this is:

KeyUsage ::= BIT STRING {  
 digitalSignature (0),  
 nonRepudiation (1),  
 keyEncipherment (2),  
 dataEncipherment (3),  
 keyAgreement (4),  
 keyCertSign (5),  
 cRLSign (6),  
 encipherOnly (7),  
 decipherOnly (8) }

RFC 2459 recommends that when used, this be marked as a critical extension.

**Returns:**the KeyUsage extension of this certificate, represented as an array of booleans. The order of KeyUsage values in the array is the same as in the above ASN.1 definition. The array will contain a value for each KeyUsage defined above. If the KeyUsage list encoded in the certificate is longer than the above list, it will not be truncated. Returns null if this certificate does not contain a KeyUsage extension.

### getExtendedKeyUsage

public [List](http://docs.google.com/java/util/List.html)<[String](http://docs.google.com/java/lang/String.html)> **getExtendedKeyUsage**()  
 throws [CertificateParsingException](http://docs.google.com/java/security/cert/CertificateParsingException.html)

Gets an unmodifiable list of Strings representing the OBJECT IDENTIFIERs of the ExtKeyUsageSyntax field of the extended key usage extension, (OID = 2.5.29.37). It indicates one or more purposes for which the certified public key may be used, in addition to or in place of the basic purposes indicated in the key usage extension field. The ASN.1 definition for this is:

ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId

KeyPurposeId ::= OBJECT IDENTIFIER

Key purposes may be defined by any organization with a need. Object identifiers used to identify key purposes shall be assigned in accordance with IANA or ITU-T Rec. X.660 | ISO/IEC/ITU 9834-1.

This method was added to version 1.4 of the Java 2 Platform Standard Edition. In order to maintain backwards compatibility with existing service providers, this method is not abstract and it provides a default implementation. Subclasses should override this method with a correct implementation.

**Returns:**the ExtendedKeyUsage extension of this certificate, as an unmodifiable list of object identifiers represented as Strings. Returns null if this certificate does not contain an ExtendedKeyUsage extension. **Throws:** [CertificateParsingException](http://docs.google.com/java/security/cert/CertificateParsingException.html) - if the extension cannot be decoded**Since:** 1.4

### getBasicConstraints

public abstract int **getBasicConstraints**()

Gets the certificate constraints path length from the critical BasicConstraints extension, (OID = 2.5.29.19).

The basic constraints extension identifies whether the subject of the certificate is a Certificate Authority (CA) and how deep a certification path may exist through that CA. The pathLenConstraint field (see below) is meaningful only if cA is set to TRUE. In this case, it gives the maximum number of CA certificates that may follow this certificate in a certification path. A value of zero indicates that only an end-entity certificate may follow in the path.

Note that for RFC 2459 this extension is always marked critical if cA is TRUE, meaning this certificate belongs to a Certificate Authority.

The ASN.1 definition for this is:

BasicConstraints ::= SEQUENCE {  
 cA BOOLEAN DEFAULT FALSE,  
 pathLenConstraint INTEGER (0..MAX) OPTIONAL }

**Returns:**the value of pathLenConstraint if the BasicConstraints extension is present in the certificate and the subject of the certificate is a CA, otherwise -1. If the subject of the certificate is a CA and pathLenConstraint does not appear, Integer.MAX\_VALUE is returned to indicate that there is no limit to the allowed length of the certification path.

### getSubjectAlternativeNames

public [Collection](http://docs.google.com/java/util/Collection.html)<[List](http://docs.google.com/java/util/List.html)<?>> **getSubjectAlternativeNames**()  
 throws [CertificateParsingException](http://docs.google.com/java/security/cert/CertificateParsingException.html)

Gets an immutable collection of subject alternative names from the SubjectAltName extension, (OID = 2.5.29.17).

The ASN.1 definition of the SubjectAltName extension is:

SubjectAltName ::= GeneralNames  
  
 GeneralNames :: = SEQUENCE SIZE (1..MAX) OF GeneralName  
  
 GeneralName ::= CHOICE {  
 otherName [0] OtherName,  
 rfc822Name [1] IA5String,  
 dNSName [2] IA5String,  
 x400Address [3] ORAddress,  
 directoryName [4] Name,  
 ediPartyName [5] EDIPartyName,  
 uniformResourceIdentifier [6] IA5String,  
 iPAddress [7] OCTET STRING,  
 registeredID [8] OBJECT IDENTIFIER}

If this certificate does not contain a SubjectAltName extension, null is returned. Otherwise, a Collection is returned with an entry representing each GeneralName included in the extension. Each entry is a List whose first entry is an Integer (the name type, 0-8) and whose second entry is a String or a byte array (the name, in string or ASN.1 DER encoded form, respectively).

RFC 822, DNS, and URI names are returned as Strings, using the well-established string formats for those types (subject to the restrictions included in RFC 2459). IPv4 address names are returned using dotted quad notation. IPv6 address names are returned in the form "a1:a2:...:a8", where a1-a8 are hexadecimal values representing the eight 16-bit pieces of the address. OID names are returned as Strings represented as a series of nonnegative integers separated by periods. And directory names (distinguished names) are returned in RFC 2253 string format. No standard string format is defined for otherNames, X.400 names, EDI party names, or any other type of names. They are returned as byte arrays containing the ASN.1 DER encoded form of the name.

Note that the Collection returned may contain more than one name of the same type. Also, note that the returned Collection is immutable and any entries containing byte arrays are cloned to protect against subsequent modifications.

This method was added to version 1.4 of the Java 2 Platform Standard Edition. In order to maintain backwards compatibility with existing service providers, this method is not abstract and it provides a default implementation. Subclasses should override this method with a correct implementation.

**Returns:**an immutable Collection of subject alternative names (or null) **Throws:** [CertificateParsingException](http://docs.google.com/java/security/cert/CertificateParsingException.html) - if the extension cannot be decoded**Since:** 1.4

### getIssuerAlternativeNames

public [Collection](http://docs.google.com/java/util/Collection.html)<[List](http://docs.google.com/java/util/List.html)<?>> **getIssuerAlternativeNames**()  
 throws [CertificateParsingException](http://docs.google.com/java/security/cert/CertificateParsingException.html)

Gets an immutable collection of issuer alternative names from the IssuerAltName extension, (OID = 2.5.29.18).

The ASN.1 definition of the IssuerAltName extension is:

IssuerAltName ::= GeneralNames

The ASN.1 definition of GeneralNames is defined in [getSubjectAlternativeNames](http://docs.google.com/java/security/cert/X509Certificate.html#getSubjectAlternativeNames()).

If this certificate does not contain an IssuerAltName extension, null is returned. Otherwise, a Collection is returned with an entry representing each GeneralName included in the extension. Each entry is a List whose first entry is an Integer (the name type, 0-8) and whose second entry is a String or a byte array (the name, in string or ASN.1 DER encoded form, respectively). For more details about the formats used for each name type, see the getSubjectAlternativeNames method.

Note that the Collection returned may contain more than one name of the same type. Also, note that the returned Collection is immutable and any entries containing byte arrays are cloned to protect against subsequent modifications.

This method was added to version 1.4 of the Java 2 Platform Standard Edition. In order to maintain backwards compatibility with existing service providers, this method is not abstract and it provides a default implementation. Subclasses should override this method with a correct implementation.

**Returns:**an immutable Collection of issuer alternative names (or null) **Throws:** [CertificateParsingException](http://docs.google.com/java/security/cert/CertificateParsingException.html) - if the extension cannot be decoded**Since:** 1.4

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/X509Certificate.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/security/cert/TrustAnchor.html)   [**NEXT CLASS**](http://docs.google.com/java/security/cert/X509CertSelector.html) | [**FRAMES**](http://docs.google.com/index.html?java/security/cert/X509Certificate.html)    [**NO FRAMES**](http://docs.google.com/X509Certificate.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | FIELD | [CONSTR](#tyjcwt) | [METHOD](#3dy6vkm) | DETAIL: FIELD | [CONSTR](#17dp8vu) | [METHOD](#26in1rg) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).