Java™ Certification Path API Specification

Public Review Draft Version 1.0

Contents

Certification Path API Specification

Description

This document is the API specification for the Java 2 Platform, Standard Edition, version 1.4.

Package Summary	
Java 2 Platform Packages	
java.security.cert[5]	Provides classes and interfaces for parsing and managing certificates, certificate revocation lists (CRLs), and certification paths.

Package
java.security.cert

Description

Provides classes and interfaces for parsing and managing certificates, certificate revocation lists (CRLs), and certification paths. It contains support for X.509 v3 certificates and X.509 v2 CRLs.

Package Specification

- RFC 2459: X.509 Certificate and CRL Profile

Related Documentation

For information about X.509 certificates and CRLs, please see:

- http://www.ietf.org/rfc/rfc2459.txt
- X.509 Certificates and CRLs

Since: 1.2

Class Summary

Interfaces

`CertPathBuilderResult` 49	A specification of the result of a certification path builder algorithm.
`CertPathParameters` 53	A specification of certification path algorithm parameters.
`CertPathValidatorResult` 63	A specification of the result of a certification path validator algorithm.
`CertSelector` 66	A selector that defines a set of criteria for selecting `Certificate`s.
`CertStoreParameters` 77	A specification of `CertStore` parameters.
`CRLSelector` 89	A selector that defines a set of criteria for selecting `CRL`s.
`PolicyNode` 125	A valid policy tree node resulting from the PKIX certification path validation algorithm.
`X509Extension` 178	Interface for an X.509 extension.

Classes

`Certificate` 8	Abstract class for managing a variety of identity certificates.
`Certificate.CertificateRep` 12	Alternate Certificate class for serialization.
`CertificateFactory` 20	This class defines the functionality of a certificate factory, which is used to generate certificate, certification path (`CertPath`) and certificate revocation list (CRL) objects from their encodings.
`CertificateFactorySpi` 27	This class defines the *Service Provider Interface* (**SPI**) for the `CertificateFactory` class.

Class Summary

CertPath 37	An immutable sequence of certificates (a certification path).
CertPathBuilder 41	A class for building certification paths (also known as certificate chains).
CertPathBuilderSpi 51	The *Service Provider Interface* (**SPI**) for the CertPathBuilder 41 class.
CertPathValidator 54	A class for validating certification paths (also known as certificate chains).
CertPathValidatorSpi 64	The *Service Provider Interface* (**SPI**) for the CertPathValidator 54 class.
CertStore 68	A class for retrieving Certificates and CRLs from a repository.
CertStoreSpi 78	The *Service Provider Interface* (**SPI**) for the CertStore 68 class.
CollectionCertStoreParameters 81	Parameters used as input for the Collection CertStore algorithm.
CRL 85	This class is an abstraction of certificate revocation lists (CRLs) that have different formats but important common uses.
LDAPCertStoreParameters 91	Parameters used as input for the LDAP CertStore algorithm.
PKIXBuilderParameters 95	Parameters used as input for the PKIX CertPathBuilder algorithm.
PKIXCertPathBuilderResult 100	This class represents the successful result of the PKIX certification path builder algorithm.
PKIXCertPathChecker 104	An abstract class that performs one or more checks on an X509Certificate.
PKIXCertPathValidatorResult 108	This class represents the successful result of the PKIX certification path validation algorithm.
PKIXParameters 112	Parameters used as input for the PKIX CertPathValidator algorithm.
PolicyQualifierInfo 128	A policy qualifier represented by the ASN.1 PolicyQualifierInfo structure.
X509Certificate 132	Abstract class for X.509 certificates.
X509CertSelector 141	A CertSelector that selects X509Certificates that match all specified criteria.
X509CRL 161	Abstract class for an X.509 Certificate Revocation List (CRL).
X509CRLEntry 168	Abstract class for a revoked certificate in a CRL (Certificate Revocation List).
X509CRLSelector 171	A CRLSelector that selects X509CRLs that match all specified criteria.
Exceptions	
CertificateEncodingException 14	Certificate Encoding Exception.
CertificateException 16	This exception indicates one of a variety of certificate problems.
CertificateExpiredException 18	Certificate Expired Exception.
CertificateNotYetValidException 33	Certificate is not yet valid exception.
CertificateParsingException 35	Certificate Parsing Exception.
CertPathBuilderException 45	An exception indicating one of a variety of problems encountered when building a certification path with a CertPathBuilder.

Class Summary	
CertPathValidatorException 58	An exception indicating one of a variety of problems encountered when validating a certification path.
CertStoreException 73	An exception indicating one of a variety of problems retrieving certificates and CRLs from a `CertStore`.
CRLException 87	CRL (Certificate Revocation List) Exception

java.security.cert
Certificate

Syntax
```
public abstract class Certificate implements java.io.Serializable
```

```
java.lang.Object
  |
  +--java.security.cert.Certificate
```

Direct Known Subclasses: X509Certificate[132]

All Implemented Interfaces: java.io.Serializable

Description
Abstract class for managing a variety of identity certificates. An identity certificate is a binding of a principal to a public key which is vouched for by another principal. (A principal represents an entity such as an individual user, a group, or a corporation.)

This class is an abstraction for certificates that have different formats but important common uses. For example, different types of certificates, such as X.509 and PGP, share general certificate functionality (like encoding and verifying) and some types of information (like a public key).

X.509, PGP, and SDSI certificates can all be implemented by subclassing the Certificate class, even though they contain different sets of information, and they store and retrieve the information in different ways.

See Also: X509Certificate[132], CertificateFactory[20]

Member Summary

Inner Classes

Certificate.CertificateRep[12]	Alternate Certificate class for serialization.

Constructors

protected	Certificate(String)[9] Creates a certificate of the specified type.

Methods

public boolean	equals(Object)[9] Compares this certificate for equality with the specified object.
public abstract byte	getEncoded()[10] Returns the encoded form of this certificate.
public abstract PublicKey	getPublicKey()[10] Gets the public key from this certificate.

Member Summary	
public final String	getType()[10] Returns the type of this certificate.
public int	hashCode()[10] Returns a hashcode value for this certificate from its encoded form.
public abstract String	toString()[10] Returns a string representation of this certificate.
public abstract void	verify(PublicKey)[11] Verifies that this certificate was signed using the private key that corresponds to the specified public key.
public abstract void	verify(PublicKey, String)[11] Verifies that this certificate was signed using the private key that corresponds to the specified public key.
protected Object	writeReplace()[11] Replace the Certificate to be serialized.

Inherited Member Summary
Methods inherited from class java.lang.Object
getClass, clone, notify, notifyAll, wait, wait, wait, finalize

Constructors

Certificate(String)

```
protected Certificate(java.lang.String type)
```

Creates a certificate of the specified type.

Parameters:

`type` - the standard name of the certificate type. See Appendix A in the Java Cryptography Architecture API Specification & Reference for information about standard certificate types.

Methods

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this certificate for equality with the specified object. If the `other` object is an `instanceof Certificate`, then its encoded form is retrieved and compared with the encoded form of this certificate.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:
`other` - the object to test for equality with this certificate.

Returns: true iff the encoded forms of the two certificates match, false otherwise.

---

getEncoded()

```
public abstract byte[] getEncoded()
```

Returns the encoded form of this certificate. It is assumed that each certificate type would have only a single form of encoding; for example, X.509 certificates would be encoded as ASN.1 DER.

Returns: the encoded form of this certificate

Throws:
`CertificateEncodingException`[14] - if an encoding error occurs.

---

getPublicKey()

```
public abstract java.security.PublicKey getPublicKey()
```

Gets the public key from this certificate.

Returns: the public key.

---

getType()

```
public final java.lang.String getType()
```

Returns the type of this certificate.

Returns: the type of this certificate.

---

hashCode()

```
public int hashCode()
```

Returns a hashcode value for this certificate from its encoded form.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: the hashcode value.

---

toString()

```
public abstract java.lang.String toString()
```

Returns a string representation of this certificate.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a string representation of this certificate.

verify(PublicKey)

```
public abstract void verify(java.security.PublicKey key)
```

Verifies that this certificate was signed using the private key that corresponds to the specified public key.

Parameters:

`key` - the PublicKey used to carry out the verification.

Throws:

`NoSuchAlgorithmException` - on unsupported signature algorithms.

`InvalidKeyException` - on incorrect key.

`NoSuchProviderException` - if there's no default provider.

`SignatureException` - on signature errors.

`CertificateException`[16] - on encoding errors.

verify(PublicKey, String)

```
public abstract void verify(java.security.PublicKey key, java.lang.String sigProvider)
```

Verifies that this certificate was signed using the private key that corresponds to the specified public key. This method uses the signature verification engine supplied by the specified provider.

Parameters:

`key` - the PublicKey used to carry out the verification.

`sigProvider` - the name of the signature provider.

Throws:

`NoSuchAlgorithmException` - on unsupported signature algorithms.

`InvalidKeyException` - on incorrect key.

`NoSuchProviderException` - on incorrect provider.

`SignatureException` - on signature errors.

`CertificateException`[16] - on encoding errors.

writeReplace()

```
protected java.lang.Object writeReplace()
```

Replace the Certificate to be serialized.

Returns: the alternate Certificate object to be serialized.

Throws:

`ObjectStreamException` - if a new object representing this certificate could not be created

`ObjectStreamException`

java.security.cert
Certificate.CertificateRep

Syntax
```
protected static class Certificate.CertificateRep implements java.io.Serializable
```

```
java.lang.Object
  |
  +--java.security.cert.Certificate.CertificateRep
```

All Implemented Interfaces: java.io.Serializable

Description
Alternate Certificate class for serialization.

Member Summary	
Constructors	
protected	Certificate.CertificateRep(String, byte[])[12] Construct the alternate Certificate class with the Certificate type and Certificate encoding bytes.
Methods	
protected Object	readResolve()[13] Resolve the Certificate Object.

Inherited Member Summary
Methods inherited from class java.lang.Object
getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize

Constructors

Certificate.CertificateRep(String, byte[])

```
protected Certificate.CertificateRep(java.lang.String type, byte[] data)
```

Construct the alternate Certificate class with the Certificate type and Certificate encoding bytes.

Parameters:

`type` - the standard name of the Certificate type.

`data` - the Certificate data.

Methods

---

readResolve()

```
protected java.lang.Object readResolve()
```

Resolve the Certificate Object.

Returns: the resolved Certificate Object.

Throws:

`ObjectStreamException` - if the Certificate could not be resolved.

`ObjectStreamException`

java.security.cert
CertificateEncodingException

Syntax
```
public class CertificateEncodingException extends CertificateException16

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--CertificateException16
                          |
                          +--java.security.cert.CertificateEncodingException
```

All Implemented Interfaces: java.io.Serializable

Description
Certificate Encoding Exception. This is thrown whenever an error occurs while attempting to encode a certificate.

Member Summary		
Constructors		
	public	CertificateEncodingException()15 Constructs a CertificateEncodingException with no detail message.
	public	CertificateEncodingException(String)15 Constructs a CertificateEncodingException with the specified detail message.

Inherited Member Summary
Methods inherited from class java.lang.Throwable
getMessage, getLocalizedMessage, toString, printStackTrace, printStackTrace, printStackTrace, fillInStackTrace
Methods inherited from class java.lang.Object
getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize

Constructors

CertificateEncodingException()

```
public CertificateEncodingException()
```

Constructs a CertificateEncodingException with no detail message. A detail message is a String that describes this particular exception.

CertificateEncodingException(String)

```
public CertificateEncodingException(java.lang.String message)
```

Constructs a CertificateEncodingException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

`message` - the detail message.

java.security.cert
CertificateException

Syntax
```
public class CertificateException extends java.security.GeneralSecurityException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--java.security.cert.CertificateException
```

Direct Known Subclasses:
CertificateEncodingException[14],CertificateExpiredException[18],CertificateNotYetValidException[33],CertificateParsingException[35]

All Implemented Interfaces:
java.io.Serializable

Description
This exception indicates one of a variety of certificate problems.

See Also:
Certificate[8]

Member Summary

Constructors

public	CertificateException()[17]
public	CertificateException(String)[17]

Constructs a certificate exception with the given detail message.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

getMessage, getLocalizedMessage, toString, printStackTrace, printStackTrace, printStackTrace, fillInStackTrace

Methods inherited from class java.lang.Object

getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize

Constructors

CertificateException()

```
public CertificateException()
```

CertificateException(String)

```
public CertificateException(java.lang.String msg)
```

Constructs a certificate exception with the given detail message. A detail message is a String that describes this particular exception.

Parameters:

`msg` - the detail message.

java.security.cert
CertificateExpiredException

Syntax
```
public class CertificateExpiredException extends CertificateException16

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--CertificateException16
                          |
                          +--java.security.cert.CertificateExpiredException
```

All Implemented Interfaces: java.io.Serializable

Description
Certificate Expired Exception. This is thrown whenever the current `Date` or the specified `Date` is after the `notAfter` date/time specified in the validity period of the certificate.

Member Summary	
Constructors	
`public`	`CertificateExpiredException()19` Constructs a CertificateExpiredException with no detail message.
`public`	`CertificateExpiredException(String)19` Constructs a CertificateExpiredException with the specified detail message.

Inherited Member Summary
Methods inherited from class java.lang.Throwable
`getMessage, getLocalizedMessage, toString, printStackTrace, printStackTrace, print-StackTrace, fillInStackTrace`
Methods inherited from class java.lang.Object
`getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertificateExpiredException()

```
public CertificateExpiredException()
```

Constructs a CertificateExpiredException with no detail message. A detail message is a String that describes this particular exception.

CertificateExpiredException(String)

```
public CertificateExpiredException(java.lang.String message)
```

Constructs a CertificateExpiredException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

`message` - the detail message.

java.security.cert
CertificateFactory

Syntax

```
public class CertificateFactory

java.lang.Object
  |
  +--java.security.cert.CertificateFactory
```

Description

This class defines the functionality of a certificate factory, which is used to generate certificate, certification path (`CertPath`) and certificate revocation list (CRL) objects from their encodings.

For encodings consisting of multiple certificates, use `generateCertificates` when you want to parse a collection of possibly unrelated certificates. Otherwise, use `generateCertPath` when you want to generate a `CertPath` (a certificate chain) and subsequently validate it with a `CertPathValidator`.

A certificate factory for X.509 must return certificates that are an instance of `java.security.cert.X509Certificate`, and CRLs that are an instance of `java.security.cert.X509CRL`.

The following example reads a file with Base64 encoded certificates, which are each bounded at the beginning by ————-BEGIN CERTIFICATE————-, and bounded at the end by ————-END CERTIFICATE————-. We convert the `FileInputStream` (which does not support `mark` and `reset`) to a `ByteArrayInputStream` (which supports those methods), so that each call to `generateCertificate` consumes only one certificate, and the read position of the input stream is positioned to the next certificate in the file:

```
FileInputStream fis = new FileInputStream(filename);
DataInputStream dis = new DataInputStream(fis);
CertificateFactory cf = CertificateFactory.getInstance("X.509");
byte[] bytes = new byte[dis.available()];
dis.readFully(bytes);
ByteArrayInputStream bais = new ByteArrayInputStream(bytes);
while (bais.available() > 0) {
   Certificate cert = cf.generateCertificate(bais);
   System.out.println(cert.toString());
}
```

The following example parses a PKCS#7-formatted certificate reply stored in a file and extracts all the certificates from it:

```
FileInputStream fis = new FileInputStream(filename);
CertificateFactory cf = CertificateFactory.getInstance("X.509");
Collection c = cf.generateCertificates(fis);
Iterator i = c.iterator();
while (i.hasNext()) {
   Certificate cert = (Certificate)i.next();
   System.out.println(cert);
}
```

Since: 1.2

See Also: `Certificate`$_{8}$, `X509Certificate`$_{132}$, `CertPath`$_{37}$, `CRL`$_{85}$, `X509CRL`$_{161}$

Member Summary

Constructors

protected	`CertificateFactory(CertificateFactorySpi, Provider, String)`[22] Creates a CertificateFactory object of the given type, and encapsulates the given provider implementation (SPI object) in it.

Methods

public final Certificate	`generateCertificate(InputStream)`[22] Generates a certificate object and initializes it with the data read from the input stream `inStream`.
public final Collection	`generateCertificates(InputStream)`[23] Returns a (possibly empty) collection view of the certificates read from the given input stream `inStream`.
public final CertPath	`generateCertPath(InputStream)`[23] Generates a `CertPath` object and initializes it with the data read from the `InputStream` inStream.
public final CertPath	`generateCertPath(InputStream, String)`[23] Generates a `CertPath` object and initializes it with the data read from the `InputStream` inStream.
public final CertPath	`generateCertPath(List)`[24] Generates a `CertPath` object and initializes it with a `List` of `Certificate`s.
public final CRL	`generateCRL(InputStream)`[24] Generates a certificate revocation list (CRL) object and initializes it with the data read from the input stream `inStream`.
public final Collection	`generateCRLs(InputStream)`[25] Returns a (possibly empty) collection view of the CRLs read from the given input stream `inStream`.
public final Iterator	`getCertPathEncodings()`[25] Returns an iteration of the `CertPath` encodings supported by this certificate factory, with the default encoding first.
public static final CertificateFactory	`getInstance(String)`[25] Generates a certificate factory object that implements the specified certificate type.
public static final CertificateFactory	`getInstance(String, String)`[26] Generates a certificate factory object for the specified certificate type from the specified provider.
public final Provider	`getProvider()`[26] Returns the provider of this certificate factory.
public final String	`getType()`[26] Returns the name of the certificate type associated with this certificate factory.

CertificateFactory(CertificateFactorySpi, Provider, String)

Inherited Member Summary
Methods inherited from class java.lang.Object `getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertificateFactory(CertificateFactorySpi, Provider, String)

```
protected CertificateFactory(CertificateFactorySpi[27] certFacSpi,
            java.security.Provider provider, java.lang.String type)
```

Creates a CertificateFactory object of the given type, and encapsulates the given provider implementation (SPI object) in it.

Parameters:

`certFacSpi` - the provider implementation.

`provider` - the provider.

`type` - the certificate type.

Methods

generateCertificate(InputStream)

```
public final Certificate[8] generateCertificate(java.io.InputStream inStream)
```

Generates a certificate object and initializes it with the data read from the input stream `inStream`.

In order to take advantage of the specialized certificate format supported by this certificate factory, the returned certificate object can be typecast to the corresponding certificate class. For example, if this certificate factory implements X.509 certificates, the returned certificate object can be typecast to the `X509Certificate` class.

In the case of a certificate factory for X.509 certificates, the certificate provided in `inStream` must be DER-encoded and may be supplied in binary or printable (Base64) encoding. If the certificate is provided in Base64 encoding, it must be bounded at the beginning by ————————-BEGIN CERTIFICATE————————-, and must be bounded at the end by ————————-END CERTIFICATE————————-.

Note that if the given input stream does not support `mark` and `reset`, this method will consume the entire input stream. Otherwise, each call to this method consumes one certificate and the read position of the input stream is positioned to the next available byte after the inherent end-of-certificate marker. If the data in the input stream does not contain an inherent end-of-certificate marker (other than EOF) and there is trailing data after the certificate is parsed, a `CertificateException` is thrown.

Parameters:

`inStream` - an input stream with the certificate data.

Returns: a certificate object initialized with the data from the input stream.

Throws:
`CertificateException`[16] - on parsing errors.

---

generateCertificates(InputStream)

```
public final java.util.Collection generateCertificates(java.io.InputStream inStream)
```

Returns a (possibly empty) collection view of the certificates read from the given input stream `inStream`.

In order to take advantage of the specialized certificate format supported by this certificate factory, each element in the returned collection view can be typecast to the corresponding certificate class. For example, if this certificate factory implements X.509 certificates, the elements in the returned collection can be typecast to the `X509Certificate` class.

In the case of a certificate factory for X.509 certificates, `inStream` may contain a sequence of DER-encoded certificates in the formats described for `generateCertificate(InputStream)`[22]. In addition, `inStream` may contain a PKCS#7 certificate chain. This is a PKCS#7 *SignedData* object, with the only significant field being *certificates*. In particular, the signature and the contents are ignored. This format allows multiple certificates to be downloaded at once. If no certificates are present, an empty collection is returned.

Note that if the given input stream does not support `mark` and `reset`, this method will consume the entire input stream.

Parameters:
`inStream` - the input stream with the certificates.

Returns: a (possibly empty) collection view of java.security.cert.Certificate objects initialized with the data from the input stream.

Throws:
`CertificateException`[16] - on parsing errors.

---

generateCertPath(InputStream)

```
public final CertPath[37] generateCertPath(java.io.InputStream inStream)
```

Generates a `CertPath` object and initializes it with the data read from the `InputStream` inStream. The data is assumed to be in the default encoding. The name of the default encoding is the first element of the `Iterator` returned by the `getCertPathEncodings()`[25] method.

Parameters:
`inStream` - an `InputStream` containing the data

Returns: a `CertPath` initialized with the data from the `InputStream`

Throws:
`CertificateException`[16] - if an exception occurs while decoding

Since: 1.4

---

generateCertPath(InputStream, String)

```
public final CertPath[37] generateCertPath(java.io.InputStream inStream,
            java.lang.String encoding)
```

Generates a `CertPath` object and initializes it with the data read from the `InputStream` inStream. The data is assumed to be in the specified encoding. See Appendix A in the Java Certification Path API Specification for information about standard encoding names and their formats.

Parameters:

`inStream` - an `InputStream` containing the data

`encoding` - the encoding used for the data

Returns: a `CertPath` initialized with the data from the `InputStream`

Throws:

`CertificateException`$_{16}$ - if an exception occurs while decoding or the encoding requested is not supported

Since: 1.4

---

generateCertPath(List)

`public final CertPath`$_{37}$ **`generateCertPath`**`(java.util.List certificates)`

Generates a `CertPath` object and initializes it with a `List` of `Certificate`s.

The certificates supplied must be of a type supported by the `CertificateFactory`. They will be copied out of the supplied `List` object.

Parameters:

`certificates` - a `List` of `Certificate`s

Returns: a `CertPath` initialized with the supplied list of certificates

Throws:

`CertificateException`$_{16}$ - if an exception occurs

Since: 1.4

---

generateCRL(InputStream)

`public final CRL`$_{85}$ **`generateCRL`**`(java.io.InputStream inStream)`

Generates a certificate revocation list (CRL) object and initializes it with the data read from the input stream `inStream`.

In order to take advantage of the specialized CRL format supported by this certificate factory, the returned CRL object can be typecast to the corresponding CRL class. For example, if this certificate factory implements X.509 CRLs, the returned CRL object can be typecast to the `X509CRL` class.

Note that if the given input stream does not support `mark` and `reset`, this method will consume the entire input stream. Otherwise, each call to this method consumes one CRL and the read position of the input stream is positioned to the next available byte after the the inherent end-of-CRL marker. If the data in the input stream does not contain an inherent end-of-CRL marker (other than EOF) and there is trailing data after the CRL is parsed, a `CRLException` is thrown.

Parameters:

`inStream` - an input stream with the CRL data.

Returns: a CRL object initialized with the data from the input stream.

Throws:

`CRLException`$_{87}$ - on parsing errors.

generateCRLs(InputStream)

```
public final java.util.Collection generateCRLs(java.io.InputStream inStream)
```

Returns a (possibly empty) collection view of the CRLs read from the given input stream `inStream`.

In order to take advantage of the specialized CRL format supported by this certificate factory, each element in the returned collection view can be typecast to the corresponding CRL class. For example, if this certificate factory implements X.509 CRLs, the elements in the returned collection can be typecast to the `X509CRL` class.

In the case of a certificate factory for X.509 CRLs, `inStream` may contain a sequence of DER-encoded CRLs. In addition, `inStream` may contain a PKCS#7 CRL set. This is a PKCS#7 *SignedData* object, with the only significant field being *crls*. In particular, the signature and the contents are ignored. This format allows multiple CRLs to be downloaded at once. If no CRLs are present, an empty collection is returned.

Note that if the given input stream does not support `mark` and `reset`, this method will consume the entire input stream.

Parameters:
`inStream` - the input stream with the CRLs.

Returns: a (possibly empty) collection view of java.security.cert.CRL objects initialized with the data from the input stream.

Throws:
`CRLException`[87] - on parsing errors.

getCertPathEncodings()

```
public final java.util.Iterator getCertPathEncodings()
```

Returns an iteration of the `CertPath` encodings supported by this certificate factory, with the default encoding first. See Appendix A in the Java Certification Path API Specification for information about standard encoding names and their formats.

Attempts to modify the returned `Iterator` via its `remove` method result in an `UnsupportedOperationException`.

Returns: an `Iterator` over the names of the supported `CertPath` encodings (as `String`s)

Since: 1.4

getInstance(String)

```
public static final CertificateFactory[20] getInstance(java.lang.String type)
```

Generates a certificate factory object that implements the specified certificate type. If the default provider package provides an implementation of the requested certificate type, an instance of certificate factory containing that implementation is returned. If the type is not available in the default package, other packages are searched.

Parameters:
`type` - the name of the requested certificate type. See Appendix A in the Java Cryptography Architecture API Specification & Reference for information about standard certificate types.

Returns: a certificate factory object for the specified type.

Throws:

`CertificateException`[16] - if the requested certificate type is not available in the default provider package or any of the other provider packages that were searched.

---

getInstance(String, String)

```
public static final CertificateFactory[20] getInstance(java.lang.String type,
            java.lang.String provider)
```

Generates a certificate factory object for the specified certificate type from the specified provider.

Parameters:

`type` - the certificate type

`provider` - the name of the provider.

Returns: a certificate factory object for the specified type.

Throws:

`CertificateException`[16] - if the certificate type is not available from the specified provider.

`NoSuchProviderException` - if the provider has not been configured.

See Also: `java.security.Provider`

---

getProvider()

```
public final java.security.Provider getProvider()
```

Returns the provider of this certificate factory.

Returns: the provider of this certificate factory.

---

getType()

```
public final java.lang.String getType()
```

Returns the name of the certificate type associated with this certificate factory.

Returns: the name of the certificate type associated with this certificate factory.

java.security.cert
CertificateFactorySpi

Syntax

```
public abstract class CertificateFactorySpi
```

```
java.lang.Object
  |
  +--java.security.cert.CertificateFactorySpi
```

Description

This class defines the *Service Provider Interface* (**SPI**) for the `CertificateFactory` class. All the abstract methods in this class must be implemented by each cryptographic service provider who wishes to supply the implementation of a certificate factory for a particular certificate type, e.g., X.509.

Certificate factories are used to generate certificate, certification path (`CertPath`) and certificate revocation list (CRL) objects from their encodings.

A certificate factory for X.509 must return certificates that are an instance of `java.security.cert.X509Certificate`, and CRLs that are an instance of `java.security.cert.X509CRL`.

Since: 1.2

See Also: `CertificateFactory`$_{20}$, `Certificate`$_{8}$, `X509Certificate`$_{132}$, `CertPath`$_{37}$, `CRL`$_{85}$, `X509CRL`$_{161}$

Member Summary

Constructors

public	`CertificateFactorySpi()`$_{28}$

Methods

public abstract Certificate	`engineGenerateCertificate(InputStream)`$_{28}$ Generates a certificate object and initializes it with the data read from the input stream `inStream`.
public abstract Collection	`engineGenerateCertificates(InputStream)`$_{29}$ Returns a (possibly empty) collection view of the certificates read from the given input stream `inStream`.
public CertPath	`engineGenerateCertPath(InputStream)`$_{29}$ Generates a `CertPath` object and initializes it with the data read from the `InputStream` inStream.
public CertPath	`engineGenerateCertPath(InputStream, String)`$_{30}$ Generates a `CertPath` object and initializes it with the data read from the `InputStream` inStream.
public CertPath	`engineGenerateCertPath(List)`$_{30}$ Generates a `CertPath` object and initializes it with a `List` of `Certificate`s.

Member Summary	
public abstract CRL	engineGenerateCRL(InputStream)[30] Generates a certificate revocation list (CRL) object and initializes it with the data read from the input stream `inStream`.
public abstract Collection	engineGenerateCRLs(InputStream)[31] Returns a (possibly empty) collection view of the CRLs read from the given input stream `inStream`.
public Iterator	engineGetCertPathEncodings()[31] Returns an iteration of the `CertPath` encodings supported by this certificate factory, with the default encoding first.

Inherited Member Summary
Methods inherited from class java.lang.Object `getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertificateFactorySpi()

```
public CertificateFactorySpi()
```

Methods

engineGenerateCertificate(InputStream)

```
public abstract Certificate[8] engineGenerateCertificate(java.io.InputStream inStream)
```

Generates a certificate object and initializes it with the data read from the input stream `inStream`.

In order to take advantage of the specialized certificate format supported by this certificate factory, the returned certificate object can be typecast to the corresponding certificate class. For example, if this certificate factory implements X.509 certificates, the returned certificate object can be typecast to the `X509Certificate` class.

In the case of a certificate factory for X.509 certificates, the certificate provided in `inStream` must be DER-encoded and may be supplied in binary or printable (Base64) encoding. If the certificate is provided in Base64 encoding, it must be bounded at the beginning by ————————-BEGIN CERTIFICATE————————-, and must be bounded at the end by ————————-END CERTIFICATE————————-.

Note that if the given input stream does not support `mark` and `reset`, this method will consume the entire input stream. Otherwise, each call to this method consumes one certificate and the read position of the input stream is positioned to the next available byte after the the inherent end-of-certificate marker. If the data in the input stream does not contain an inherent end-of-certificate marker (other than EOF) and there is trailing data after the certificate is parsed, a `CertificateException` is thrown.

Parameters:
`inStream` - an input stream with the certificate data.

Returns: a certificate object initialized with the data from the input stream.

Throws:
`CertificateException`[16] - on parsing errors.

---

engineGenerateCertificates(InputStream)

```
public abstract java.util.Collection engineGenerateCertificates(java.io.InputStream
            inStream)
```

Returns a (possibly empty) collection view of the certificates read from the given input stream `inStream`.

In order to take advantage of the specialized certificate format supported by this certificate factory, each element in the returned collection view can be typecast to the corresponding certificate class. For example, if this certificate factory implements X.509 certificates, the elements in the returned collection can be typecast to the `X509Certificate` class.

In the case of a certificate factory for X.509 certificates, `inStream` may contain a single DER-encoded certificate in the formats described for `generateCertificate(InputStream)`[22]. In addition, `inStream` may contain a PKCS#7 certificate chain. This is a PKCS#7 *SignedData* object, with the only significant field being *certificates*. In particular, the signature and the contents are ignored. This format allows multiple certificates to be downloaded at once. If no certificates are present, an empty collection is returned.

Note that if the given input stream does not support `mark` and `reset`, this method will consume the entire input stream.

Parameters:
`inStream` - the input stream with the certificates.

Returns: a (possibly empty) collection view of java.security.cert.Certificate objects initialized with the data from the input stream.

Throws:
`CertificateException`[16] - on parsing errors.

---

engineGenerateCertPath(InputStream)

```
public CertPath[37] engineGenerateCertPath(java.io.InputStream inStream)
```

Generates a `CertPath` object and initializes it with the data read from the `InputStream` inStream. The data is assumed to be in the default encoding.

Parameters:
`inStream` - an `InputStream` containing the data

Returns: a `CertPath` initialized with the data from the `InputStream`

Throws:

`CertificateException`[16] - if an exception occurs while decoding

Since: 1.4

---

engineGenerateCertPath(InputStream, String)

```
public CertPath[37] engineGenerateCertPath(java.io.InputStream inStream,
            java.lang.String encoding)
```

Generates a `CertPath` object and initializes it with the data read from the `InputStream` inStream. The data is assumed to be in the specified encoding.

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 cannot be `abstract` and by default throws an `UnsupportedOperationException`.

Parameters:

`inStream` - an `InputStream` containing the data

`encoding` - the encoding used for the data

Returns: a `CertPath` initialized with the data from the `InputStream`

Throws:

`CertificateException`[16] - if an exception occurs while decoding or the encoding requested is not supported

`UnsupportedOperationException` - if the method is not supported

Since: 1.4

---

engineGenerateCertPath(List)

```
public CertPath[37] engineGenerateCertPath(java.util.List certificates)
```

Generates a `CertPath` object and initializes it with a `List` of `Certificate`s.

The certificates supplied must be of a type supported by the `CertificateFactory`. They will be copied out of the supplied `List` object.

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 cannot be `abstract` and by default throws an `UnsupportedOperationException`.

Parameters:

`certificates` - a `List` of `Certificate`s

Returns: a `CertPath` initialized with the supplied list of certificates

Throws:

`CertificateException`[16] - if an exception occurs

`UnsupportedOperationException` - if the method is not supported

Since: 1.4

---

engineGenerateCRL(InputStream)

```
public abstract CRL[85] engineGenerateCRL(java.io.InputStream inStream)
```

Generates a certificate revocation list (CRL) object and initializes it with the data read from the input stream `inStream`.

In order to take advantage of the specialized CRL format supported by this certificate factory, the returned CRL object can be typecast to the corresponding CRL class. For example, if this certificate factory implements X.509 CRLs, the returned CRL object can be typecast to the `X509CRL` class.

Note that if the given input stream does not support `mark` and `reset`, this method will consume the entire input stream. Otherwise, each call to this method consumes one CRL and the read position of the input stream is positioned to the next available byte after the the inherent end-of-CRL marker. If the data in the input stream does not contain an inherent end-of-CRL marker (other than EOF) and there is trailing data after the CRL is parsed, a `CRLException` is thrown.

Parameters:
`inStream` - an input stream with the CRL data.

Returns: a CRL object initialized with the data from the input stream.

Throws:
`CRLException`$_{87}$ - on parsing errors.

---

engineGenerateCRLs(InputStream)

```
public abstract java.util.Collection engineGenerateCRLs(java.io.InputStream inStream)
```

Returns a (possibly empty) collection view of the CRLs read from the given input stream `inStream`.

In order to take advantage of the specialized CRL format supported by this certificate factory, each element in the returned collection view can be typecast to the corresponding CRL class. For example, if this certificate factory implements X.509 CRLs, the elements in the returned collection can be typecast to the `X509CRL` class.

In the case of a certificate factory for X.509 CRLs, `inStream` may contain a single DER-encoded CRL. In addition, `inStream` may contain a PKCS#7 CRL set. This is a PKCS#7 *SignedData* object, with the only significant field being *crls*. In particular, the signature and the contents are ignored. This format allows multiple CRLs to be downloaded at once. If no CRLs are present, an empty collection is returned.

Note that if the given input stream does not support `mark` and `reset`, this method will consume the entire input stream.

Parameters:
`inStream` - the input stream with the CRLs.

Returns: a (possibly empty) collection view of java.security.cert.CRL objects initialized with the data from the input stream.

Throws:
`CRLException`$_{87}$ - on parsing errors.

---

engineGetCertPathEncodings()

```
public java.util.Iterator engineGetCertPathEncodings()
```

Returns an iteration of the `CertPath` encodings supported by this certificate factory, with the default encoding first. See Appendix A in the Java Certification Path API Specification for information about standard encoding names.

Attempts to modify the returned `Iterator` via its `remove` method result in an `UnsupportedOperationException`.

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 cannot be `abstract` and by default throws an `UnsupportedOperationException`.

Returns: an `Iterator` over the names of the supported `CertPath` encodings (as `String`s)

Throws:

`UnsupportedOperationException` - if the method is not supported

Since: 1.4

java.security.cert
CertificateNotYetValidException

Syntax
```
public class CertificateNotYetValidException extends CertificateException16

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--CertificateException16
                          |
                          +--java.security.cert.CertificateNotYetValidException
```

All Implemented Interfaces: java.io.Serializable

Description
Certificate is not yet valid exception. This is thrown whenever the current `Date` or the specified `Date` is before the `notBefore` date/time in the Certificate validity period.

Member Summary	
Constructors	
`public`	`CertificateNotYetValidException()`34 Constructs a CertificateNotYetValidException with no detail message.
`public`	`CertificateNotYetValidException(String)`34 Constructs a CertificateNotYetValidException with the specified detail message.

Inherited Member Summary
Methods inherited from class java.lang.Throwable
`getMessage, getLocalizedMessage, toString, printStackTrace, printStackTrace, print-StackTrace, fillInStackTrace`
Methods inherited from class java.lang.Object
`getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertificateNotYetValidException()

```
public CertificateNotYetValidException()
```

Constructs a CertificateNotYetValidException with no detail message. A detail message is a String that describes this particular exception.

CertificateNotYetValidException(String)

```
public CertificateNotYetValidException(java.lang.String message)
```

Constructs a CertificateNotYetValidException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

`message` - the detail message.

java.security.cert

CertificateParsingException

Syntax

```
public class CertificateParsingException extends CertificateException16

java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--CertificateException16
                          |
                          +--java.security.cert.CertificateParsingException
```

All Implemented Interfaces: java.io.Serializable

Description

Certificate Parsing Exception. This is thrown whenever an invalid DER-encoded certificate is parsed or unsupported DER features are found in the Certificate.

Member Summary	
Constructors	
public	CertificateParsingException()36 Constructs a CertificateParsingException with no detail message.
public	CertificateParsingException(String)36 Constructs a CertificateParsingException with the specified detail message.

Inherited Member Summary
Methods inherited from class java.lang.Throwable
getMessage, getLocalizedMessage, toString, printStackTrace, printStackTrace, printStackTrace, fillInStackTrace
Methods inherited from class java.lang.Object
getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize

Constructors

CertificateParsingException()

```
public CertificateParsingException()
```

Constructs a CertificateParsingException with no detail message. A detail message is a String that describes this particular exception.

CertificateParsingException(String)

```
public CertificateParsingException(java.lang.String message)
```

Constructs a CertificateParsingException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

`message` - the detail message.

java.security.cert
CertPath

Syntax

```
public abstract class CertPath
```

```
java.lang.Object
  |
  +--java.security.cert.CertPath
```

Description

An immutable sequence of certificates (a certification path).

This is an abstract class that defines the methods common to all `CertPath`s. Subclasses can handle different kinds of certificates (X.509, PGP, etc.).

All CertPath objects have a type, a list of `Certificate`s, and one or more supported encodings. Because the `CertPath` class is immutable, a `CertPath` cannot change in any externally visible way after being constructed. This stipulation applies to all public fields and methods of this class and any added by subclasses.

The type is a `String` that identifies the type of `Certificate`s in the certification path. For each certificate `cert` in a certification path `certPath`, `cert.getType().equals(certPath.getType())` must be `true`.

The list of `Certificate`s is an ordered `List` of zero or more `Certificate`s. This `List` and all of the `Certificate`s contained in it must be immutable.

Each `CertPath` object must support one or more encodings so that the object can be translated into a byte array for storage or transmission to other parties. Preferably, these encodings should be well-documented standards (such as PKCS#7). One of the encodings supported by a `CertPath` is considered the default encoding. This encoding is used if no encoding is explicitly requested (for the `getEncoded()`$_{39}$ method, for instance).

`CertPath` objects can be created with a `CertificateFactory` or they can be returned by other classes, such as a `CertPathBuilder`.

By convention, X.509 `CertPath`s (consisting of `X509Certificate`s), are ordered from target to trust anchor. That is, the issuer of one certificate is the subject of the following one. However, unvalidated X.509 `CertPath`s may not follow this convention. PKIX `CertPathValidator`s will detect any departure from this convention and throw a `CertPathValidatorException`.

Concurrent Access

All `CertPath` objects must be thread-safe. That is, multiple threads may concurrently invoke the methods defined in this class on a single `CertPath` object (or more than one) with no ill effects. This is also true for the `List` returned by `CertPath.getCertificates`.

Requiring `CertPath` objects to be immutable and thread-safe allows them to be passed around to various pieces of code without worrying about coordinating access. Providing this thread-safety is generally not difficult, since the `CertPath` and `List` objects in question are immutable.

Since: 1.4

See Also: `CertificateFactory`$_{20}$, `CertPathBuilder`$_{41}$

Member Summary

Constructors

protected	CertPath(String)[38]
	Creates a `CertPath` of the specified type.

Methods

public boolean	equals(Object)[39]
	Compares this certification path for equality with the specified object.
public abstract List	getCertificates()[39]
	Returns the list of certificates in this certification path.
public abstract byte	getEncoded()[39]
	Returns the encoded form of this certification path, using the default encoding.
public abstract byte	getEncoded(String)[39]
	Returns the encoded form of this certification path, using the specified encoding.
public abstract Iterator	getEncodings()[40]
	Returns an iteration of the encodings supported by this certification path, with the default encoding first.
public String	getType()[40]
	Returns the type of `Certificate`s in this certification path.
public int	hashCode()[40]
	Returns the hashcode for this certification path.
public String	toString()[40]
	Returns a string representation of this certification path.

Inherited Member Summary

Methods inherited from class java.lang.Object

`getClass, clone, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertPath(String)

```
protected CertPath(java.lang.String type)
```

Creates a `CertPath` of the specified type.

This constructor is protected because most users should use a `CertificateFactory` to create `CertPath`s.

Parameters:
`type` - the standard name of the type of `Certificate`s in this path

Methods

---

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this certification path for equality with the specified object. Two `CertPath`s are equal if and only if their types are equal and their certificate `List`s (and by implication the `Certificate`s in those `List`s) are equal. A `CertPath` is never equal to an object that is not a `CertPath`.

This algorithm is implemented by this method. If it is overridden, the behavior specified here must be maintained.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:
`other` - the object to test for equality with this certification path

Returns: true if the specified object is equal to this certification path, false otherwise

---

getCertificates()

```
public abstract java.util.List getCertificates()
```

Returns the list of certificates in this certification path. The `List` returned must be immutable and thread-safe.

Returns: an immutable `List` of `Certificate`s (may be empty, but not null)

---

getEncoded()

```
public abstract byte[] getEncoded()
```

Returns the encoded form of this certification path, using the default encoding.

Returns: the encoded bytes

Throws:
`CertificateEncodingException`[14] - if an encoding error occurs

---

getEncoded(String)

```
public abstract byte[] getEncoded(java.lang.String encoding)
```

Returns the encoded form of this certification path, using the specified encoding.

Parameters:
`encoding` - the name of the encoding to use

getEncodings()

Returns: the encoded bytes

Throws:

`CertificateEncodingException`[14] - if an encoding error occurs or the encoding requested is not supported

---

getEncodings()

```
public abstract java.util.Iterator getEncodings()
```

Returns an iteration of the encodings supported by this certification path, with the default encoding first. Attempts to modify the returned `Iterator` via its `remove` method result in an `UnsupportedOperationException`.

Returns: an `Iterator` over the names of the supported encodings (as Strings)

---

getType()

```
public java.lang.String getType()
```

Returns the type of `Certificate`s in this certification path. This is the same string that would be returned by `getType()`[10] for all `Certificate`s in the certification path.

Returns: the type of `Certificate`s in this certification path (never null)

---

hashCode()

```
public int hashCode()
```

Returns the hashcode for this certification path. The hash code of a certification path is defined to be the result of the following calculation:

```
hashCode = path.getType().hashCode();
hashCode = 31*hashCode + path.getCertificates().hashCode();
```

This ensures that `path1.equals(path2)` implies that `path1.hashCode()==path2.hashCode()` for any two certification paths, `path1` and `path2`, as required by the general contract of `Object.hashCode`.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: the hashcode value for this certification path

---

toString()

```
public java.lang.String toString()
```

Returns a string representation of this certification path. This calls the `toString` method on each of the `Certificate`s in the path.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a string representation of this certification path

java.security.cert
CertPathBuilder

Syntax

```
public class CertPathBuilder

java.lang.Object
  |
  +--java.security.cert.CertPathBuilder
```

Description

A class for building certification paths (also known as certificate chains).

This class uses a provider-based architecture, as described in the Java Cryptography Architecture. To create a `CertPathBuilder`, call one of the static `getInstance` methods, passing in the algorithm name of the `CertPathBuilder` desired and optionally the name of the provider desired.

Once a `CertPathBuilder` object has been created, certification paths can be constructed by calling the `build(CertPathParameters)`$_{42}$ method and passing it an algorithm-specific set of parameters. If successful, the result (including the `CertPath` that was built) is returned in an object that implements the `CertPathBuilderResult` interface.

Concurrent Access

The static methods of this class are guaranteed to be thread-safe. Multiple threads may concurrently invoke the static methods defined in this class with no ill effects.

However, this is not true for the non-static methods defined by this class. Unless otherwise documented by a specific provider, threads that need to access a single `CertPathBuilder` instance concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating a different `CertPathBuilder` instance need not synchronize.

Since: 1.4

See Also: `CertPath`$_{37}$

Member Summary

Constructors

`protected`	`CertPathBuilder(CertPathBuilderSpi, Provider, String)`$_{42}$
	Creates a `CertPathBuilder` object of the given algorithm, and encapsulates the given provider implementation (SPI object) in it.

Methods

`public final CertPathBuilderResult`	`build(CertPathParameters)`$_{42}$
	Attempts to build a certification path using the specified algorithm parameter set.
`public final String`	`getAlgorithm()`$_{43}$
	Returns the name of the algorithm of this `CertPathBuilder`.

Member Summary	
public static final String	`getDefaultType()`[43] Returns the default `CertPathBuilder` type as specified in the Java security properties file, or the string "PKIX" if no such property exists.
public static CertPathBuilder	`getInstance(String)`[43] Returns a `CertPathBuilder` object that implements the specified algorithm.
public static CertPathBuilder	`getInstance(String, String)`[44] Returns a `CertPathBuilder` object that implements the specified algorithm, as supplied by the specified provider.
public final Provider	`getProvider()`[44] Returns the provider of this `CertPathBuilder`.

Inherited Member Summary
Methods inherited from class java.lang.Object `getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertPathBuilder(CertPathBuilderSpi, Provider, String)

```
protected CertPathBuilder(CertPathBuilderSpi[51] builderSpi,
            java.security.Provider provider, java.lang.String algorithm)
```

Creates a `CertPathBuilder` object of the given algorithm, and encapsulates the given provider implementation (SPI object) in it.

Parameters:

`builderSpi` - the provider implementation

`provider` - the provider

`algorithm` - the algorithm name

Methods

build(CertPathParameters)

```
public final CertPathBuilderResult[49] build(CertPathParameters[53] params)
```

Attempts to build a certification path using the specified algorithm parameter set.

Parameters:
`params` - the algorithm parameters

Returns: the result of the build algorithm

Throws:
`CertPathBuilderException`[45] - if the builder is unable to construct a certification path that satisfies the specified parameters

`InvalidAlgorithmParameterException` - if the specified parameters are inappropriate for this `CertPathBuilder`

---

getAlgorithm()

```
public final java.lang.String getAlgorithm()
```

Returns the name of the algorithm of this `CertPathBuilder`.

Returns: the name of the algorithm of this `CertPathBuilder`

---

getDefaultType()

```
public static final java.lang.String getDefaultType()
```

Returns the default `CertPathBuilder` type as specified in the Java security properties file, or the string "PKIX" if no such property exists. The Java security properties file is located in the file named <JAVA_HOME>/lib/security/java.security, where <JAVA_HOME> refers to the directory where the SDK was installed.

The default `CertPathBuilder` type can be used by applications that do not want to use a hard-coded type when calling one of the `getInstance` methods, and want to provide a default type in case a user does not specify its own.

The default `CertPathBuilder` type can be changed by setting the value of the "certpathbuilder.type" security property (in the Java security properties file) to the desired type.

Returns: the default `CertPathBuilder` type as specified in the Java security properties file, or the string "PKIX" if no such property exists.

---

getInstance(String)

```
public static CertPathBuilder[41] getInstance(java.lang.String algorithm)
```

Returns a `CertPathBuilder` object that implements the specified algorithm.

If the default provider package provides an implementation of the specified `CertPathBuilder` algorithm, an instance of `CertPathBuilder` containing that implementation is returned. If the requested algorithm is not available in the default package, other packages are searched.

Parameters:
`algorithm` - the name of the requested `CertPathBuilder` algorithm

Returns: a `CertPathBuilder` object that implements the specified algorithm

Throws:
`NoSuchAlgorithmException` - if the requested algorithm is not available in the default provider package or any of the other provider packages that were searched

getInstance(String, String)

```
public static CertPathBuilder41 getInstance(java.lang.String algorithm,
            java.lang.String provider)
```

Returns a `CertPathBuilder` object that implements the specified algorithm, as supplied by the specified provider.

Parameters:

`algorithm` - the name of the requested `CertPathBuilder` algorithm

`provider` - the name of the provider

Returns: a `CertPathBuilder` object that implements the specified algorithm, as supplied by the specified provider

Throws:

`NoSuchAlgorithmException` - if the requested algorithm is not available from the specified provider

`NoSuchProviderException` - if the provider has not been configured

getProvider()

```
public final java.security.Provider getProvider()
```

Returns the provider of this `CertPathBuilder`.

Returns: the provider of this `CertPathBuilder`

java.security.cert
CertPathBuilderException

Syntax

```
public class CertPathBuilderException extends java.security.GeneralSecurityException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--java.security.cert.CertPathBuilderException
```

All Implemented Interfaces: java.io.Serializable

Description

An exception indicating one of a variety of problems encountered when building a certification path with a `CertPathBuilder`.

A `CertPathBuilderException` provides support for wrapping exceptions. The `getInternalException()`[47] method returns the internal exception, if any, that caused this exception to be thrown.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: `CertPathBuilder`[41]

Member Summary

Constructors

`public`	`CertPathBuilderException()`[46] Creates a `CertPathBuilderException` with `null` as its detail message.
`public`	`CertPathBuilderException(Exception)`[46] Creates a `CertPathBuilderException` that wraps the specified internal exception.
`public`	`CertPathBuilderException(String)`[47] Creates a `CertPathBuilderException` with the given detail message.

Member Summary

public	`CertPathBuilderException(String, Exception)`[47]
	Creates a `CertPathBuilderException` with the given detail message and internal exception.

Methods

public Exception	`getInternalException()`[47]
	Returns the internal (wrapped) exception, or `null` if there is no internal exception.
public String	`getMessage()`[47]
	Returns the detail message, including the message from the internal (wrapped) exception if there is one.
public void	`printStackTrace()`[47]
	Prints a stack trace to `System.err`, including the internal exception, if any.
public void	`printStackTrace(PrintStream)`[48]
	Prints a stack trace to a `PrintStream`, including the internal exception, if any.
public void	`printStackTrace(PrintWriter)`[48]
	Prints a stack trace to a `PrintWriter`, including the internal exception, if any.
public String	`toString()`[48]
	Returns a string describing this exception, including a description of the internal (wrapped) exception if there is one.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

`getLocalizedMessage, fillInStackTrace`

Methods inherited from class java.lang.Object

`getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertPathBuilderException()

```
public CertPathBuilderException()
```

Creates a `CertPathBuilderException` with `null` as its detail message.

CertPathBuilderException(Exception)

```
public CertPathBuilderException(java.lang.Exception e)
```

Creates a `CertPathBuilderException` that wraps the specified internal exception. This allows any exception to be converted into a `CertPathBuilderException`, while retaining information about the wrapped exception, which may be useful for debugging.

Parameters:
`e` - the internal exception

CertPathBuilderException(String)

```
public CertPathBuilderException(java.lang.String msg)
```

Creates a `CertPathBuilderException` with the given detail message. The detail message is a `String` that describes this particular exception in more detail.

Parameters:
`msg` - the detail message

CertPathBuilderException(String, Exception)

```
public CertPathBuilderException(java.lang.String msg, java.lang.Exception e)
```

Creates a `CertPathBuilderException` with the given detail message and internal exception.

Parameters:
`msg` - the detail message

`e` - the internal exception

Methods

---

getInternalException()

```
public java.lang.Exception getInternalException()
```

Returns the internal (wrapped) exception, or `null` if there is no internal exception.

Returns: the internal exception, or `null` if there is no internal exception

---

getMessage()

```
public java.lang.String getMessage()
```

Returns the detail message, including the message from the internal (wrapped) exception if there is one.

Overrides: java.lang.Throwable.getMessage() in class java.lang.Throwable

Returns: the detail message, or `null` if neither the message nor internal exception were specified

---

printStackTrace()

```
public void printStackTrace()
```

Prints a stack trace to `System.err`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace() in class java.lang.Throwable

---

printStackTrace(PrintStream)

```
public void printStackTrace(java.io.PrintStream ps)
```

Prints a stack trace to a `PrintStream`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace(java.io.PrintStream) in class java.lang.Throwable

Parameters:

`ps` - the `PrintStream` to use for output

---

printStackTrace(PrintWriter)

```
public void printStackTrace(java.io.PrintWriter pw)
```

Prints a stack trace to a `PrintWriter`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace(java.io.PrintWriter) in class java.lang.Throwable

Parameters:

`pw` - the `PrintWriter` to use for output

---

toString()

```
public java.lang.String toString()
```

Returns a string describing this exception, including a description of the internal (wrapped) exception if there is one.

Overrides: java.lang.Throwable.toString() in class java.lang.Throwable

Returns: a string representation of this `CertPathBuilderException`

java.security.cert
CertPathBuilderResult

Syntax
```
public interface CertPathBuilderResult extends java.lang.Cloneable
```

All Superinterfaces: java.lang.Cloneable

All Known Implementing Classes: PKIXCertPathBuilderResult[100]

Description
A specification of the result of a certification path builder algorithm. All results returned by the build(CertPathParameters)[42] method must implement this interface.

At a minimum, a `CertPathBuilderResult` contains the `CertPath` built by the `CertPathBuilder` instance. Implementations of this interface may add methods to return implementation or algorithm specific information, such as debugging information or certification path validation results.

Concurrent Access

Unless otherwise specified, the methods defined in this interface are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: CertPathBuilder[41]

Member Summary		
Methods		
	`public Object`	clone()[49] Makes a copy of this `CertPathBuilderResult`.
	`public CertPath`	getCertPath()[50] Returns the built certification path.

Methods

clone()
```
public java.lang.Object clone()
```

Makes a copy of this `CertPathBuilderResult`. Changes to the copy will not affect the original and vice versa.

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: a copy of this `CertPathBuilderResult`

---

getCertPath()

`public CertPath`$_{37}$ **`getCertPath`**`()`

Returns the built certification path.

Returns: the certification path (never `null`)

java.security.cert
CertPathBuilderSpi

Syntax

```
public abstract class CertPathBuilderSpi
```

```
java.lang.Object
  |
  +--java.security.cert.CertPathBuilderSpi
```

Description

The *Service Provider Interface* (**SPI**) for the `CertPathBuilder`[41] class. All `CertPathBuilder` implementations must include a class (the SPI class) that extends this class (`CertPathBuilderSpi`) and implements all of its methods. In general, instances of this class should only be accessed through the `CertPathBuilder` class. For details, see the Java Cryptography Architecture.

Concurrent Access

Instances of this class need not be protected against concurrent access from multiple threads. Threads that need to access a single `CertPathBuilderSpi` instance concurrently should synchronize amongst themselves and provide the necessary locking before calling the wrapping `CertPathBuilder` object.

However, implementations of `CertPathBuilderSpi` may still encounter concurrency issues, since multiple threads each manipulating a different `CertPathBuilderSpi` instance need not synchronize.

Since: 1.4

Member Summary

Constructors

`public`	`CertPathBuilderSpi()`[52]
	The default constructor.

Methods

`public abstract CertPathBuilderResult`	`engineBuild(CertPathParameters)`[52]
	Attempts to build a certification path using the specified algorithm parameter set.

Inherited Member Summary

Methods inherited from class java.lang.Object

`getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertPathBuilderSpi()

`public CertPathBuilderSpi()`

The default constructor.

Methods

engineBuild(CertPathParameters)

`public abstract CertPathBuilderResult`$_{49}$ `engineBuild(CertPathParameters`$_{53}$ `params)`

Attempts to build a certification path using the specified algorithm parameter set.

Parameters:
`params` - the algorithm parameters

Returns: the result of the build algorithm

Throws:
`CertPathBuilderException`$_{45}$ - if the builder is unable to construct a certification path that satisfies the specified parameters

`InvalidAlgorithmParameterException` - if the specified parameters are inappropriate for this `CertPathBuilder`

java.security.cert
CertPathParameters

Syntax
```
public interface CertPathParameters extends java.lang.Cloneable
```

All Superinterfaces: java.lang.Cloneable

All Known Implementing Classes: `PKIXParameters`[112]

Description
A specification of certification path algorithm parameters. The purpose of this interface is to group (and provide type safety for) all `CertPath` parameter specifications. All `CertPath` parameter specifications must implement this interface.

Since: 1.4

See Also: `validate(CertPath, CertPathParameters)`[57], `build(CertPathParameters)`[42]

Member Summary	
Methods	
`public Object`	`clone()`[53] Makes a copy of this `CertPathParameters`.

Methods

clone()

`public java.lang.Object clone()`

Makes a copy of this `CertPathParameters`. Changes to the copy will not affect the original and vice versa.

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: a copy of this `CertPathParameters`

java.security.cert
CertPathValidator

Syntax

```
public class CertPathValidator
```

```
java.lang.Object
  |
  +--java.security.cert.CertPathValidator
```

Description

A class for validating certification paths (also known as certificate chains).

This class uses a provider-based architecture, as described in the Java Cryptography Architecture. To create a `CertPathValidator`, call one of the static `getInstance` methods, passing in the algorithm name of the `CertPathValidator` desired and optionally the name of the provider desired.

Once a `CertPathValidator` object has been created, it can be used to validate certification paths by calling the `validate(CertPath, CertPathParameters)`$_{57}$ method and passing it the `CertPath` to be validated and an algorithm-specific set of parameters. If successful, the result is returned in an object that implements the `CertPathValidatorResult` interface.

Concurrent Access

The static methods of this class are guaranteed to be thread-safe. Multiple threads may concurrently invoke the static methods defined in this class with no ill effects.

However, this is not true for the non-static methods defined by this class. Unless otherwise documented by a specific provider, threads that need to access a single `CertPathValidator` instance concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating a different `CertPathValidator` instance need not synchronize.

Since: 1.4

See Also: `CertPath`$_{37}$

Member Summary

Constructors

`protected`	`CertPathValidator(CertPathValidatorSpi, Provider, String)`$_{55}$
	Creates a `CertPathValidator` object of the given algorithm, and encapsulates the given provider implementation (SPI object) in it.

Methods

`public final String`	`getAlgorithm()`$_{55}$
	Returns the algorithm name of this `CertPathValidator`.

Member Summary	
public static final String	getDefaultType()[56] Returns the default CertPathValidator type as specified in the Java security properties file, or the string "PKIX" if no such property exists.
public static CertPathValidator	getInstance(String)[56] Returns a CertPathValidator object that implements the specified algorithm.
public static CertPathValidator	getInstance(String, String)[56] Returns a CertPathValidator object that implements the specified algorithm, as supplied by the specified provider.
public final Provider	getProvider()[57] Returns the Provider of this CertPathValidator.
public final CertPathValidatorResult	validate(CertPath, CertPathParameters)[57] Validates the specified certification path using the specified algorithm parameter set.

Inherited Member Summary
Methods inherited from class java.lang.Object
getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize

Constructors

CertPathValidator(CertPathValidatorSpi, Provider, String)

```
protected CertPathValidator(CertPathValidatorSpi[64] validatorSpi,
          java.security.Provider provider, java.lang.String algorithm)
```

Creates a CertPathValidator object of the given algorithm, and encapsulates the given provider implementation (SPI object) in it.

Parameters:

validatorSpi - the provider implementation

provider - the provider

algorithm - the algorithm name

Methods

getAlgorithm()

getDefaultType()

```
public final java.lang.String getAlgorithm()
```

Returns the algorithm name of this `CertPathValidator`.

Returns: the algorithm name of this `CertPathValidator`

---

getDefaultType()

```
public static final java.lang.String getDefaultType()
```

Returns the default `CertPathValidator` type as specified in the Java security properties file, or the string "PKIX" if no such property exists. The Java security properties file is located in the file named <JAVA_HOME>/lib/security/java.security, where <JAVA_HOME> refers to the directory where the SDK was installed.

The default `CertPathValidator` type can be used by applications that do not want to use a hard-coded type when calling one of the `getInstance` methods, and want to provide a default type in case a user does not specify its own.

The default `CertPathValidator` type can be changed by setting the value of the "certpathvalidator.type" security property (in the Java security properties file) to the desired type.

Returns: the default `CertPathValidator` type as specified in the Java security properties file, or the string "PKIX" if no such property exists.

---

getInstance(String)

```
public static CertPathValidator54 getInstance(java.lang.String algorithm)
```

Returns a `CertPathValidator` object that implements the specified algorithm.

If the default provider package provides an implementation of the specified `CertPathValidator` algorithm, an instance of `CertPathValidator` containing that implementation is returned. If the requested algorithm is not available in the default package, other packages are searched.

Parameters:

`algorithm` - the name of the requested `CertPathValidator` algorithm

Returns: a `CertPathValidator` object that implements the specified algorithm

Throws:

`NoSuchAlgorithmException` - if the requested algorithm is not available in the default provider package or any of the other provider packages that were searched

---

getInstance(String, String)

```
public static CertPathValidator54 getInstance(java.lang.String algorithm,
            java.lang.String provider)
```

Returns a `CertPathValidator` object that implements the specified algorithm, as supplied by the specified provider.

Parameters:

`algorithm` - the name of the requested `CertPathValidator` algorithm

`provider` - the name of the provider

Returns: a `CertPathValidator` object that implements the specified algorithm, as supplied by the specified provider

Throws:

`NoSuchAlgorithmException` - if the requested algorithm is not available from the specified provider

`NoSuchProviderException` - if the provider has not been configured

---

getProvider()

```
public final java.security.Provider getProvider()
```

Returns the `Provider` of this `CertPathValidator`.

Returns: the `Provider` of this `CertPathValidator`

---

validate(CertPath, CertPathParameters)

```
public final CertPathValidatorResult63 validate(CertPath37 certPath,
            CertPathParameters53 params)
```

Validates the specified certification path using the specified algorithm parameter set.

The `CertPath` specified must be of a type that is supported by the validation algorithm, otherwise an `InvalidAlgorithmParameterException` will be thrown. For example, a `CertPathValidator` that implements the PKIX algorithm validates `CertPath` objects of type X.509.

Parameters:

`certPath` - the `CertPath` to be validated

`params` - the algorithm parameters

Returns: the result of the validation algorithm

Throws:

`CertPathValidatorException`[58] - if the `CertPath` does not validate

`InvalidAlgorithmParameterException` - if the specified parameters or the type of the specified `CertPath` are inappropriate for this `CertPathValidator`

java.security.cert
CertPathValidatorException

Syntax

```
public class CertPathValidatorException extends java.security.GeneralSecurityException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--java.security.cert.CertPathValidatorException
```

All Implemented Interfaces: java.io.Serializable

Description

An exception indicating one of a variety of problems encountered when validating a certification path. A `CertPathValidatorException` provides support for retrieving the certification path that was being validated when the exception was thrown, as well as retrieving the index of the certificate in the certification path that caused the exception to be thrown.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: `CertPathValidator` 54

Member Summary

Constructors

public	`CertPathValidatorException()` 59
public	`CertPathValidatorException(Exception)` 59 Creates a `CertPathValidatorException` that wraps the specified exception.
public	`CertPathValidatorException(String)` 60 Creates a `CertPathValidatorException` with the given detail message.
public	`CertPathValidatorException(String, Exception)` 60 Creates a `CertPathValidatorException` with the given detail message and internal exception.

Member Summary

public	`CertPathValidatorException(String, Exception, CertPath, int)`[60]
	Creates a `CertPathValidatorException` with the given detail message, internal exception, certification path, and index.

Methods

public CertPath	`getCertPath()`[61]
	Returns the certification path that was being validated when the exception was thrown
public int	`getIndex()`[61]
	Returns the index of the certificate in the certification path that caused the exception to be thrown.
public Exception	`getInternalException()`[61]
	Returns the internal (wrapped) exception, or `null` if there is no internal exception.
public void	`printStackTrace()`[61]
	Prints a stack trace to `System.err`, including the internal exception, if any.
public void	`printStackTrace(PrintStream)`[61]
	Prints a stack trace to a `PrintStream`, including the internal exception, if any.
public void	`printStackTrace(PrintWriter)`[61]
	Prints a stack trace to a `PrintWriter`, including the internal exception, if any.
public String	`toString()`[62]
	Returns a string describing this exception, including a description of the internal (wrapped) exception if there is one.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

`getMessage, getLocalizedMessage, fillInStackTrace`

Methods inherited from class java.lang.Object

`getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertPathValidatorException()

```
public CertPathValidatorException()
```

CertPathValidatorException(Exception)

```
public CertPathValidatorException(java.lang.Exception e)
```

Creates a `CertPathValidatorException` that wraps the specified exception. This allows any exception to be converted into a `CertPathValidatorException`, while retaining information about the wrapped exception, which may be useful for debugging.

Parameters:

`e` - the internal exception

CertPathValidatorException(String)

```
public CertPathValidatorException(java.lang.String msg)
```

Creates a `CertPathValidatorException` with the given detail message. A detail message is a `String` that describes this particular exception.

Parameters:

`msg` - the detail message

CertPathValidatorException(String, Exception)

```
public CertPathValidatorException(java.lang.String msg, java.lang.Exception e)
```

Creates a `CertPathValidatorException` with the given detail message and internal exception.

Parameters:

`msg` - the detail message

`e` - the internal exception

CertPathValidatorException(String, Exception, CertPath, int)

```
public CertPathValidatorException(java.lang.String msg, java.lang.Exception e,
            CertPath37 certPath, int index)
```

Creates a `CertPathValidatorException` with the given detail message, internal exception, certification path, and index.

Parameters:

`msg` - the detail message (or `null` if none)

`e` - the internal exception (or `null` if none)

`certPath` - the certification path that was in the process of being validated when the error was encountered

`index` - the index of the certificate in the certification path that caused the error (or -1 if not applicable). Note that the list of certificates in a `CertPath` is zero based.

Throws:

`IndexOutofBoundsException` - if the index is out of range `(index < -1 || (certPath != null && index >= certPath.getCertificates().size())`

`IllegalArgumentException` - if `certPath` is `null` and `index` is not -1

Methods

---

getCertPath()

`public CertPath`[37] **`getCertPath`**`()`

Returns the certification path that was being validated when the exception was thrown

Returns: the `CertPath` that was being validated when the exception was thrown (or `null` if not specified)

---

getIndex()

`public int` **`getIndex`**`()`

Returns the index of the certificate in the certification path that caused the exception to be thrown. Note that the list of certificates in a `CertPath` is zero based. If no index has been set, -1 is returned.

Returns: the index that has been set, or -1 if none has been set

---

getInternalException()

`public java.lang.Exception` **`getInternalException`**`()`

Returns the internal (wrapped) exception, or `null` if there is no internal exception.

Returns: the internal exception, or `null` if there is no internal exception

---

printStackTrace()

`public void` **`printStackTrace`**`()`

Prints a stack trace to `System.err`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace() in class java.lang.Throwable

---

printStackTrace(PrintStream)

`public void` **`printStackTrace`**`(java.io.PrintStream ps)`

Prints a stack trace to a `PrintStream`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace(java.io.PrintStream) in class java.lang.Throwable

Parameters:
`ps` - the `PrintStream` to use for output

---

printStackTrace(PrintWriter)

`public void` **`printStackTrace`**`(java.io.PrintWriter pw)`

Prints a stack trace to a `PrintWriter`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace(java.io.PrintWriter) in class java.lang.Throwable

Parameters:

`pw` - the `PrintWriter` to use for output

---

toString()

```
public java.lang.String toString()
```

Returns a string describing this exception, including a description of the internal (wrapped) exception if there is one.

Overrides: java.lang.Throwable.toString() in class java.lang.Throwable

Returns: a string representation of this `CertPathValidatorException`

java.security.cert
CertPathValidatorResult

Syntax
```
public interface CertPathValidatorResult extends java.lang.Cloneable
```

All Superinterfaces: java.lang.Cloneable

All Known Implementing Classes: PKIXCertPathValidatorResult[108]

Description
A specification of the result of a certification path validator algorithm.

The purpose of this interface is to group (and provide type safety for) all certification path validator results. All results returned by the validate(CertPath, CertPathParameters)[57] method must implement this interface.

Since: 1.4

See Also: CertPathValidator[54]

Member Summary	
Methods	
public Object	clone()[63] Makes a copy of this CertPathValidatorResult.

Methods

---

clone()

```
public java.lang.Object clone()
```

Makes a copy of this CertPathValidatorResult. Changes to the copy will not affect the original and vice versa.

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: a copy of this CertPathValidatorResult

java.security.cert
CertPathValidatorSpi

Syntax

```
public abstract class CertPathValidatorSpi
```

```
java.lang.Object
  |
  +--java.security.cert.CertPathValidatorSpi
```

Description

The *Service Provider Interface* (**SPI**) for the `CertPathValidator`[54] class. All `CertPathValidator` implementations must include a class (the SPI class) that extends this class (`CertPathValidatorSpi`) and implements all of its methods. In general, instances of this class should only be accessed through the `CertPathValidator` class. For details, see the Java Cryptography Architecture.

Concurrent Access

Instances of this class need not be protected against concurrent access from multiple threads. Threads that need to access a single `CertPathValidatorSpi` instance concurrently should synchronize amongst themselves and provide the necessary locking before calling the wrapping `CertPathValidator` object.

However, implementations of `CertPathValidatorSpi` may still encounter concurrency issues, since multiple threads each manipulating a different `CertPathValidatorSpi` instance need not synchronize.

Since: 1.4

Member Summary

Constructors

`public`	`CertPathValidatorSpi()`[65] The default constructor.

Methods

`public abstract CertPathValidatorResult`	`engineValidate(CertPath, CertPathParameters)`[65] Validates the specified certification path using the specified algorithm parameter set.

Inherited Member Summary

Methods inherited from class java.lang.Object

`getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertPathValidatorSpi()

```
public CertPathValidatorSpi()
```

The default constructor.

Methods

engineValidate(CertPath, CertPathParameters)

```
public abstract CertPathValidatorResult63 engineValidate(CertPath37 certPath,
            CertPathParameters53 params)
```

Validates the specified certification path using the specified algorithm parameter set.

The `CertPath` specified must be of a type that is supported by the validation algorithm, otherwise an `InvalidAlgorithmParameterException` will be thrown. For example, a `CertPathValidator` that implements the PKIX algorithm validates `CertPath` objects of type X.509.

Parameters:

`certPath` - the `CertPath` to be validated

`params` - the algorithm parameters

Returns: the result of the validation algorithm

Throws:

`CertPathValidatorException58` - if the `CertPath` does not validate

`InvalidAlgorithmParameterException` - if the specified parameters or the type of the specified `CertPath` are inappropriate for this `CertPathValidator`

java.security.cert
CertSelector

Syntax
```
public interface CertSelector extends java.lang.Cloneable
```

All Superinterfaces: java.lang.Cloneable

All Known Implementing Classes: `X509CertSelector`[141]

Description
A selector that defines a set of criteria for selecting `Certificate`s. Classes that implement this interface are often used to specify which `Certificate`s should be retrieved from a `CertStore`.

Concurrent Access

Unless otherwise specified, the methods defined in this interface are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: `Certificate`[8], `CertStore`[68], `getCertificates(CertSelector)`[70]

Member Summary		
Methods		
	`public Object`	`clone()`[66] Makes a copy of this `CertSelector`.
	`public boolean`	`match(Certificate)`[67] Decides whether a `Certificate` should be selected.

Methods

clone()

`public java.lang.Object clone()`

Makes a copy of this `CertSelector`. Changes to the copy will not affect the original and vice versa.

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: a copy of this `CertSelector`

match(Certificate)

```
public boolean match(Certificate[8] cert)
```

Decides whether a `Certificate` should be selected.

Parameters:

`cert` - the `Certificate` to be checked

Returns: `true` if the `Certificate` should be selected, `false` otherwise

java.security.cert
CertStore

Syntax

```
public class CertStore

java.lang.Object
  |
  +--java.security.cert.CertStore
```

Description

A class for retrieving `Certificate`s and `CRL`s from a repository.

This class uses a provider-based architecture, as described in the Java Cryptography Architecture. To create a `CertStore`, call one of the static `getInstance` methods, passing in the type of `CertStore` desired and optionally the name of the provider desired. Then call the `init(CertStoreParameters)`$_{72}$ method to initialize the `CertStore`.

Once the `CertStore` has been initialized, it can be used to retrieve `Certificate`s and `CRL`s by calling its `getCertificates(CertSelector)`$_{70}$ and `getCRLs(CRLSelector)`$_{70}$ methods.

Unlike a `KeyStore` , which provides access to a cache of private keys and trusted certificates, a `CertStore` is designed to provide access to a potentially vast repository of untrusted certificates and CRLs. For example, an LDAP implementation of `CertStore` provides access to certificates and CRLs stored in one or more directories using the LDAP protocol and the schema as defined in the RFC service attribute. See Appendix A in the Java Certification Path API Specification for more information about standard `CertStore` types.

Concurrent Access

The `getCertificates` and `getCRLs` methods of all `CertStore` objects must be thread-safe. That is, multiple threads may concurrently invoke these methods on a single `CertStore` object (or more than one) with no ill effects. This allows a `CertPathBuilder` to search for a CRL while simultaneously searching for further certificates, for instance.

The static methods of this class are also guaranteed to be thread-safe. Multiple threads may concurrently invoke the static methods defined in this class with no ill effects.

The `init` method should not be called concurrently with any other method nor should multiple threads invoke the `init` method concurrently on a single `CertStore` object. Otherwise, the results are undefined.

Since: 1.4

Member Summary

Constructors

`protected`	`CertStore(CertStoreSpi, Provider, String)`$_{69}$
	Creates a `CertStore` object of the given type, and encapsulates the given provider implementation (SPI object) in it.

Methods

Member Summary	
public final Collection	getCertificates(CertSelector)[70] Returns a Collection of Certificates that match the specified selector.
public final Collection	getCRLs(CRLSelector)[70] Returns a Collection of CRLs that match the specified selector.
public static final String	getDefaultType()[70] Returns the default CertStore type as specified in the Java security properties file, or the string “LDAP” if no such property exists.
public static CertStore	getInstance(String)[71] Returns a CertStore object that implements the specified CertStore type.
public static CertStore	getInstance(String, String)[71] Creates a CertStore object that implements the specified CertStore type, as supplied by the specified provider.
public final Provider	getProvider()[71] Returns the provider of this CertStore.
public final String	getType()[72] Returns the type of this CertStore.
public final void	init(CertStoreParameters)[72] Initializes this CertStore with the specified parameters.

Inherited Member Summary
Methods inherited from class java.lang.Object
getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize

Constructors

CertStore(CertStoreSpi, Provider, String)

```
protected CertStore(CertStoreSpi[78] storeSpi, java.security.Provider provider,
            java.lang.String type)
```

Creates a CertStore object of the given type, and encapsulates the given provider implementation (SPI object) in it.

Parameters:

storeSpi - the provider implementation

provider - the provider

type - the type

Methods

getCertificates(CertSelector)

```
public final java.util.Collection getCertificates(CertSelector66 selector)
```

Returns a `Collection` of `Certificate`s that match the specified selector. If no `Certificate`s match the selector, an empty `Collection` will be returned.

For some `CertStore` types, the resulting `Collection` may not contain **all** of the `Certificate`s that match the selector. For instance, an LDAP `CertStore` may not search all entries in the directory. Instead, it may just search entries that are likely to contain the `Certificate`s it is looking for.

Some `CertStore` implementations (especially LDAP `CertStore`s) may throw a `CertStoreException` unless a non-null `CertSelector` is provided that includes specific criteria that can be used to find the certificates. Issuer and/or subject names are especially useful criteria.

Parameters:

`selector` - A `CertSelector` used to select which `Certificate`s should be returned. Specify `null` to return all `Certificate`s (if supported).

Returns: A `Collection` of `Certificate`s that match the specified selector (never `null`)

Throws:

`CertStoreException`73 - if an exception occurs

getCRLs(CRLSelector)

```
public final java.util.Collection getCRLs(CRLSelector89 selector)
```

Returns a `Collection` of `CRL`s that match the specified selector. If no `CRL`s match the selector, an empty `Collection` will be returned.

For some `CertStore` types, the resulting `Collection` may not contain **all** of the `CRL`s that match the selector. For instance, an LDAP `CertStore` may not search all entries in the directory. Instead, it may just search entries that are likely to contain the `CRL`s it is looking for.

Some `CertStore` implementations (especially LDAP `CertStore`s) may throw a `CertStoreException` unless a non-null `CRLSelector` is provided that includes specific criteria that can be used to find the CRLs. Issuer names and/or the certificate to be checked are especially useful.

Parameters:

`selector` - A `CRLSelector` used to select which `CRL`s should be returned. Specify `null` to return all `CRL`s (if supported).

Returns: A `Collection` of `CRL`s that match the specified selector (never `null`)

Throws:

`CertStoreException`73 - if an exception occurs

getDefaultType()

```
public static final java.lang.String getDefaultType()
```

Returns the default `CertStore` type as specified in the Java security properties file, or the string “LDAP” if no such property exists. The Java security properties file is located in the file named <JAVA_HOME>/lib/security/java.security, where <JAVA_HOME> refers to the directory where the SDK was installed.

The default `CertStore` type can be used by applications that do not want to use a hard-coded type when calling one of the `getInstance` methods, and want to provide a default `CertStore` type in case a user does not specify its own.

The default `CertStore` type can be changed by setting the value of the “certstore.type” security property (in the Java security properties file) to the desired type.

Returns: the default `CertStore` type as specified in the Java security properties file, or the string “LDAP” if no such property exists.

---

getInstance(String)

```
public static CertStore[68] getInstance(java.lang.String type)
```

Returns a `CertStore` object that implements the specified `CertStore` type.

If the default provider package provides an implementation of the specified `CertStore` type, an instance of `CertStore` containing that implementation is returned. If the requested type is not available in the default package, other packages are searched.

Parameters:

`type` - the name of the requested `CertStore` type

Returns: a `CertStore` object that implements the specified `CertStore` type

Throws:

`NoSuchAlgorithmException` - if the requested type is not available in the default provider package or any of the other provider packages that were searched

---

getInstance(String, String)

```
public static CertStore[68] getInstance(java.lang.String type, java.lang.String provider)
```

Creates a `CertStore` object that implements the specified `CertStore` type, as supplied by the specified provider.

Parameters:

`type` - the requested CertStore type

`provider` - the name of the provider

Returns: a `CertStore` object that implements the specified type, as supplied by the specified provider

Throws:

`NoSuchAlgorithmException` - if the requested type is not available from the specified provider

`NoSuchProviderException` - if the provider has not been configured

---

getProvider()

```
public final java.security.Provider getProvider()
```

Returns the provider of this `CertStore`.

Returns: the provider of this `CertStore`

getType()

```
public final java.lang.String getType()
```

Returns the type of this `CertStore`.

Returns: the type of this `CertStore`

init(CertStoreParameters)

```
public final void init(CertStoreParameters77 params)
```

Initializes this `CertStore` with the specified parameters. The type of parameters needed may vary between different types of `CertStores`.

If this method is called while another thread is concurrently calling the `getCertificates` or `getCRLs` method on the same `CertStore` object, the results are undefined.

Parameters:
`params` - the algorithm parameters

Throws:
`InvalidAlgorithmParameterException` - if the given algorithm parameters are inappropriate for this `CertStore`

java.security.cert
CertStoreException

Syntax

```
public class CertStoreException extends java.security.GeneralSecurityException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--java.security.cert.CertStoreException
```

All Implemented Interfaces: java.io.Serializable

Description

An exception indicating one of a variety of problems retrieving certificates and CRLs from a `CertStore`.

A `CertStoreException` provides support for wrapping exceptions. The `getInternalException()`[75] method returns the internal exception, if any, that caused this exception to be thrown.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: `CertStore`[68]

Member Summary

Constructors

public	`CertStoreException()`[74]
	Creates a `CertStoreException` with `null` as its detail message.
public	`CertStoreException(Exception)`[74]
	Creates a `CertStoreException` that wraps the specified internal exception.
public	`CertStoreException(String)`[75]
	Creates a `CertStoreException` with the given detail message.
public	`CertStoreException(String, Exception)`[75]
	Creates a `CertStoreException` with the given detail message and internal exception.

Member Summary

Methods

public Exception	`getInternalException()`[75] Returns the internal (wrapped) exception, or `null` if there is no internal exception.
public String	`getMessage()`[75] Returns the detail message, including the message from the internal (wrapped) exception if there is one.
public void	`printStackTrace()`[75] Prints a stack trace to `System.err`, including the internal exception, if any.
public void	`printStackTrace(PrintStream)`[75] Prints a stack trace to a `PrintStream`, including the internal exception, if any.
public void	`printStackTrace(PrintWriter)`[76] Prints a stack trace to a `PrintWriter`, including the internal exception, if any.
public String	`toString()`[76] Returns a string describing this exception, including a description of the internal (wrapped) exception if there is one.

Inherited Member Summary

Methods inherited from class java.lang.Throwable

`getLocalizedMessage, fillInStackTrace`

Methods inherited from class java.lang.Object

`getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertStoreException()

`public` **`CertStoreException`**`()`

Creates a `CertStoreException` with `null` as its detail message.

CertStoreException(Exception)

`public` **`CertStoreException`**`(java.lang.Exception e)`

Creates a `CertStoreException` that wraps the specified internal exception. This allows any exception to be converted into a `CertStoreException`, while retaining information about the wrapped exception, which may be useful for debugging.

Parameters:

`e` - the internal exception

CertStoreException(String)

```
public CertStoreException(java.lang.String msg)
```

Creates a `CertStoreException` with the given detail message. A detail message is a `String` that describes this particular exception.

Parameters:

`msg` - the detail message

CertStoreException(String, Exception)

```
public CertStoreException(java.lang.String msg, java.lang.Exception e)
```

Creates a `CertStoreException` with the given detail message and internal exception.

Parameters:

`msg` - the detail message

`e` - the internal exception

Methods

---

getInternalException()

```
public java.lang.Exception getInternalException()
```

Returns the internal (wrapped) exception, or `null` if there is no internal exception.

Returns: the internal exception, or `null` if there is no internal exception

---

getMessage()

```
public java.lang.String getMessage()
```

Returns the detail message, including the message from the internal (wrapped) exception if there is one.

Overrides: java.lang.Throwable.getMessage() in class java.lang.Throwable

Returns: the detail message, or `null` if neither the message nor internal exception were specified

---

printStackTrace()

```
public void printStackTrace()
```

Prints a stack trace to `System.err`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace() in class java.lang.Throwable

---

printStackTrace(PrintStream)

```
public void printStackTrace(java.io.PrintStream ps)
```

Prints a stack trace to a `PrintStream`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace(java.io.PrintStream) in class java.lang.Throwable

Parameters:

`ps` - the `PrintStream` to use for output

---

printStackTrace(PrintWriter)

```
public void printStackTrace(java.io.PrintWriter pw)
```

Prints a stack trace to a `PrintWriter`, including the internal exception, if any.

Overrides: java.lang.Throwable.printStackTrace(java.io.PrintWriter) in class java.lang.Throwable

Parameters:

`pw` - the `PrintWriter` to use for output

---

toString()

```
public java.lang.String toString()
```

Returns a string describing this exception, including a description of the internal (wrapped) exception if there is one.

Overrides: java.lang.Throwable.toString() in class java.lang.Throwable

Returns: a string representation of this `CertStoreException`

java.security.cert
CertStoreParameters

Syntax
```
public interface CertStoreParameters extends java.lang.Cloneable
```

All Superinterfaces: java.lang.Cloneable

All Known Implementing Classes:
`CollectionCertStoreParameters`[81],`LDAPCertStoreParameters`[91]

Description
A specification of `CertStore` parameters.

The purpose of this interface is to group (and provide type safety for) all `CertStore` parameter specifications. All `CertStore` parameter specifications must implement this interface.

Since: 1.4

See Also: `init(CertStoreParameters)`[72]

Member Summary	
Methods	
`public Object`	`clone()`[77] Makes a copy of this `CertStoreParameters`.

Methods

---

clone()

`public java.lang.Object` **`clone`**`()`

Makes a copy of this `CertStoreParameters`. Changes to the copy will not affect the original and vice versa.

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: a copy of this `CertStoreParameters`

java.security.cert
CertStoreSpi

Syntax

```
public abstract class CertStoreSpi
```

```
java.lang.Object
  |
  +--java.security.cert.CertStoreSpi
```

Description

The *Service Provider Interface* (**SPI**) for the `CertStore`[68] class. All `CertStore` implementations must include a class (the SPI class) that extends this class (`CertStoreSpi`) and implements all of its methods. In general, instances of this class should only be accessed through the `CertStore` class. For details, see the Java Cryptography Architecture.

Concurrent Access

The `getCertificates` and `getCRLs` methods of all `CertStoreSpi` objects must be thread-safe. That is, multiple threads may concurrently invoke these methods on a single `CertStoreSpi` object (or more than one) with no ill effects. This allows a `CertPathBuilder` to search for a CRL while simultaneously searching for further certificates, for instance.

The `engineInit` method should not be called concurrently with any other method nor should multiple threads invoke the `engineInit` method concurrently on a single `CertStoreSpi` object. Otherwise, the results are undefined.

Simple `CertStoreSpi` implementations will probably ensure thread safety by adding a `synchronized` keyword to their `engineGetCertificates` and `engineGetCRLs` methods. More sophisticated ones may allow truly concurrent access.

Since: 1.4

Member Summary

Constructors

public	`CertStoreSpi()`[79]
	The default constructor.

Methods

public abstract Collection	`engineGetCertificates(CertSelector)`[79]
	Returns a `Collection` of `Certificate`s that match the specified selector.
public abstract Collection	`engineGetCRLs(CRLSelector)`[79]
	Returns a `Collection` of `CRL`s that match the specified selector.
public abstract void	`engineInit(CertStoreParameters)`[80]
	Initializes this `CertStore` with the specified parameters.

Inherited Member Summary
Methods inherited from class java.lang.Object
`getClass, hashCode, equals, clone, toString, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CertStoreSpi()

`public CertStoreSpi()`

The default constructor.

Methods

engineGetCertificates(CertSelector)

`public abstract java.util.Collection engineGetCertificates(CertSelector`[66] `selector)`

Returns a `Collection` of `Certificate`s that match the specified selector. If no `Certificate`s match the selector, an empty `Collection` will be returned.

For some `CertStore` types, the resulting `Collection` may not contain **all** of the `Certificate`s that match the selector. For instance, an LDAP `CertStore` may not search all entries in the directory. Instead, it may just search entries that are likely to contain the `Certificate`s it is looking for.

Some `CertStore` implementations (especially LDAP `CertStore`s) may throw a `CertStoreException` unless a non-null `CertSelector` is provided that includes specific criteria that can be used to find the certificates. Issuer and/or subject names are especially useful criteria.

Parameters:
`selector` - A `CertSelector` used to select which `Certificate`s should be returned. Specify `null` to return all `Certificate`s (if supported).

Returns: A `Collection` of `Certificate`s that match the specified selector (never `null`)

Throws:
`CertStoreException`[73] - if an exception occurs

engineGetCRLs(CRLSelector)

`public abstract java.util.Collection engineGetCRLs(CRLSelector`[89] `selector)`

Returns a `Collection` of `CRL`s that match the specified selector. If no `CRL`s match the selector, an empty `Collection` will be returned.

For some `CertStore` types, the resulting `Collection` may not contain **all** of the `CRL`s that match the selector. For instance, an LDAP `CertStore` may not search all entries in the directory. Instead, it may just search entries that are likely to contain the `CRL`s it is looking for.

Some `CertStore` implementations (especially LDAP `CertStore`s) may throw a `CertStoreException` unless a non-null `CRLSelector` is provided that includes specific criteria that can be used to find the CRLs. Issuer names and/or the certificate to be checked are especially useful.

Parameters:

`selector` - A `CRLSelector` used to select which `CRL`s should be returned. Specify `null` to return all `CRL`s (if supported).

Returns: A `Collection` of `CRL`s that match the specified selector (never `null`)

Throws:

`CertStoreException`[73] - if an exception occurs

---

engineInit(CertStoreParameters)

```
public abstract void engineInit(CertStoreParameters[77] params)
```

Initializes this `CertStore` with the specified parameters. The type of parameters needed may vary between different types of `CertStores`.

Parameters:

`params` - the algorithm parameters

Throws:

`InvalidAlgorithmParameterException` - if the given algorithm parameters are inappropriate for this `CertStore`

java.security.cert
CollectionCertStoreParameters

Syntax

```
public class CollectionCertStoreParameters implements CertStoreParameters77

java.lang.Object
  |
  +--java.security.cert.CollectionCertStoreParameters
```

All Implemented Interfaces: CertStoreParameters77,java.lang.Cloneable

Description

Parameters used as input for the Collection `CertStore` algorithm.

This class is used to provide necessary configuration parameters to implementations of the Collection `CertStore` algorithm. The only parameter included in this class is the `Collection` from which the `CertStore` will retrieve certificates and CRLs.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: `java.util.Collection`, CertStore68

Member Summary

Constructors

public	CollectionCertStoreParameters()82 Creates an instance of `CollectionCertStoreParameters` with the default parameter values (an empty and immutable `Collection`).
public	CollectionCertStoreParameters(Collection)82 Creates an instance of `CollectionCertStoreParameters` which will allow certificates and CRLs to be retrieved from the specified `Collection`.

Methods

public Object	clone()83 Returns a copy of this object.
public boolean	equals(Object)83 Compares this object for equality with the specified object.
public Collection	getCollection()83 Returns the `Collection` from which `Certificate`s and `CRL`s are retrieved.

Member Summary	
public int	hashCode()[83] Returns a hash code value for this object.
public void	setCollection(Collection)[83] Sets the Collection from which Certificates and CRLs will be retrieved.
public String	toString()[84] Returns a formatted string describing the parameters.

Inherited Member Summary
Methods inherited from class java.lang.Object getClass, notify, notifyAll, wait, wait, wait, finalize

Constructors

CollectionCertStoreParameters()

public **CollectionCertStoreParameters**()

Creates an instance of CollectionCertStoreParameters with the default parameter values (an empty and immutable Collection).

CollectionCertStoreParameters(Collection)

public **CollectionCertStoreParameters**(java.util.Collection collection)

Creates an instance of CollectionCertStoreParameters which will allow certificates and CRLs to be retrieved from the specified Collection. If the specified Collection contains an object that is not a Certificate or CRL, that object will be ignored by the Collection CertStore.

The Collection is **not** copied. Instead, a reference is used. This allows the caller to subsequently add or remove Certificates or CRLs from the Collection, thus changing the set of Certificates or CRLs available to the Collection CertStore. The Collection CertStore will not modify the contents of the Collection.

If the Collection will be modified by one thread while another thread is calling a method of a Collection CertStore that has been initialized with this Collection, the Collection must have fail-fast iterators.

Parameters:
collection - a Collection of Certificates and CRLs

Throws:
NullPointerException - if collection is null

Methods

---

clone()

`public java.lang.Object clone()`

Returns a copy of this object.

Specified By: `clone()`$_{83}$ in interface `CollectionCertStoreParameters`$_{81}$

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: the copy

---

equals(Object)

`public boolean equals(java.lang.Object other)`

Compares this object for equality with the specified object. If the other object is not a `CollectionCertStoreParameters` object, we return `false`. Otherwise, we return `true` if the parameters are equal.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:
`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

---

getCollection()

`public java.util.Collection getCollection()`

Returns the `Collection` from which `Certificate`s and `CRL`s are retrieved. This is **not** a copy of the `Collection`, it is a reference. This allows the caller to subsequently add or remove `Certificates` or `CRL`s from the `Collection`.

Returns: the `Collection` (never null)

---

hashCode()

`public int hashCode()`

Returns a hash code value for this object.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: a hash code value

---

setCollection(Collection)

`public void setCollection(java.util.Collection collection)`

Sets the `Collection` from which `Certificate`s and `CRL`s will be retrieved. This `Collection` replaces any `Collection` previously supplied to this object. If the specified `Collection` contains an object that is not a `Certificate` or `CRL`, that object will be ignored by the Collection `CertStore`.

The `Collection` is **not** copied. Instead, a reference is used. This allows the caller to subsequently add or remove `Certificates` or `CRL`s from the `Collection`, thus changing the set of `Certificates` or `CRL`s available to the Collection `CertStore`.

If the `Collection` will be modified by one thread while another thread is calling a method of a Collection `CertStore` that has been initialized with this `Collection`, the `Collection` must have fail-fast iterators.

Parameters:
`collection` - the specified `Collection`

Throws:
`NullPointerException` - if `collection` is `null`

---

toString()

```
public java.lang.String toString()
```

Returns a formatted string describing the parameters.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a formatted string describing the parameters

java.security.cert
CRL

Syntax

```
public abstract class CRL
```

```
java.lang.Object
  |
  +--java.security.cert.CRL
```

Direct Known Subclasses: X509CRL[161]

Description

This class is an abstraction of certificate revocation lists (CRLs) that have different formats but important common uses. For example, all CRLs share the functionality of listing revoked certificates, and can be queried on whether or not they list a given certificate.

Specialized CRL types can be defined by subclassing off of this abstract class.

Since: 1.2

See Also: X509CRL[161], CertificateFactory[20]

Member Summary

Constructors

protected	CRL(String)[86] Creates a CRL of the specified type.

Methods

public final String	getType()[86] Returns the type of this CRL.
public abstract boolean	isRevoked(Certificate)[86] Checks whether the given certificate is on this CRL.
public abstract String	toString()[86] Returns a string representation of this CRL.

Inherited Member Summary

Methods inherited from class java.lang.Object

getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize

Constructors

CRL(String)

```
protected CRL(java.lang.String type)
```

Creates a CRL of the specified type.

Parameters:

`type` - the standard name of the CRL type. See Appendix A in the Java Cryptography Architecture API Specification & Reference for information about standard CRL types.

Methods

getType()

```
public final java.lang.String getType()
```

Returns the type of this CRL.

Returns: the type of this CRL.

isRevoked(Certificate)

```
public abstract boolean isRevoked(Certificate[8] cert)
```

Checks whether the given certificate is on this CRL.

Parameters:

`cert` - the certificate to check for.

Returns: true if the given certificate is on this CRL, false otherwise.

toString()

```
public abstract java.lang.String toString()
```

Returns a string representation of this CRL.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a string representation of this CRL.

java.security.cert
CRLException

Syntax
```
public class CRLException extends java.security.GeneralSecurityException
```

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--java.security.GeneralSecurityException
                    |
                    +--java.security.cert.CRLException
```

All Implemented Interfaces: java.io.Serializable

Description
CRL (Certificate Revocation List) Exception

Member Summary	
Constructors	
`public`	`CRLException()`[88] Constructs a CRLException with no detail message.
`public`	`CRLException(String)`[88] Constructs a CRLException with the specified detail message.

Inherited Member Summary
Methods inherited from class java.lang.Throwable
`getMessage, getLocalizedMessage, toString, printStackTrace, printStackTrace, print-StackTrace, fillInStackTrace`
Methods inherited from class java.lang.Object
`getClass, hashCode, equals, clone, notify, notifyAll, wait, wait, wait, finalize`

Constructors

CRLException()

```
public CRLException()
```

Constructs a CRLException with no detail message. A detail message is a String that describes this particular exception.

CRLException(String)

```
public CRLException(java.lang.String message)
```

Constructs a CRLException with the specified detail message. A detail message is a String that describes this particular exception.

Parameters:

`message` - the detail message.

java.security.cert
CRLSelector

Syntax
```
public interface CRLSelector extends java.lang.Cloneable
```

All Superinterfaces: java.lang.Cloneable

All Known Implementing Classes: X509CRLSelector[171]

Description
A selector that defines a set of criteria for selecting CRLs. Classes that implement this interface are often used to specify which CRLs should be retrieved from a CertStore.

Concurrent Access

Unless otherwise specified, the methods defined in this interface are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: CRL[85], CertStore[68], getCRLs(CRLSelector)[70]

Member Summary		
Methods		
public Object	clone()[89]	Makes a copy of this CRLSelector.
public boolean	match(CRL)[90]	Decides whether a CRL should be selected.

Methods

---

clone()

```
public java.lang.Object clone()
```

Makes a copy of this CRLSelector. Changes to the copy will not affect the original and vice versa.

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: a copy of this CRLSelector

match(CRL)

```
public boolean match(CRL85 crl)
```

Decides whether a `CRL` should be selected.

Parameters:

`crl` - the `CRL` to be checked

Returns: `true` if the `CRL` should be selected, `false` otherwise

java.security.cert
LDAPCertStoreParameters

Syntax

```
public class LDAPCertStoreParameters implements CertStoreParameters77

java.lang.Object
  |
  +--java.security.cert.LDAPCertStoreParameters
```

All Implemented Interfaces: CertStoreParameters77,java.lang.Cloneable

Description

Parameters used as input for the LDAP `CertStore` algorithm.

This class is used to provide necessary configuration parameters (server name and port number) to implementations of the LDAP `CertStore` algorithm.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: CertStore68

Member Summary

Constructors

public	LDAPCertStoreParameters()92 Creates an instance of `LDAPCertStoreParameters` with the default parameter values (server name "localhost", port 389).
public	LDAPCertStoreParameters(String, int)92 Creates an instance of `LDAPCertStoreParameters` with the specified parameter values.

Methods

public Object	clone()93 Returns a copy of this object.
public boolean	equals(Object)93 Compares this object for equality with the specified object.
public int	getPort()93 Returns the port number of the LDAP server.

LDAPCertStoreParameters()

Member Summary	
public String	`getServerName()`[93] Returns the DNS name of the LDAP server.
public int	`hashCode()`[93] Returns a hash code value for this object.
public void	`setPort(int)`[93] Sets the port number of the LDAP server.
public void	`setServerName(String)`[94] Sets the DNS name of the LDAP server.
public String	`toString()`[94] Returns a formatted string describing the parameters.

Inherited Member Summary
Methods inherited from class java.lang.Object
`getClass, notify, notifyAll, wait, wait, wait, finalize`

Constructors

LDAPCertStoreParameters()

`public` **`LDAPCertStoreParameters`**`()`

Creates an instance of `LDAPCertStoreParameters` with the default parameter values (server name “localhost”, port 389).

LDAPCertStoreParameters(String, int)

`public` **`LDAPCertStoreParameters`**`(java.lang.String serverName, int port)`

Creates an instance of `LDAPCertStoreParameters` with the specified parameter values.

Parameters:

`serverName` - the DNS name of the LDAP server

`port` - the port number of the LDAP server

Throws:

`NullPointerException` - if `serverName` is `null`

Methods

---

clone()

```
public java.lang.Object clone()
```

Returns a copy of this object.

Specified By: clone()[93] in interface LDAPCertStoreParameters[91]

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: the copy

---

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this object for equality with the specified object. If the other object is not an `LDAPCertStoreParameters` object, we return `false`. Otherwise, we return `true` if the parameters are equal.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:

`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

---

getPort()

```
public int getPort()
```

Returns the port number of the LDAP server.

Returns: the port number

---

getServerName()

```
public java.lang.String getServerName()
```

Returns the DNS name of the LDAP server.

Returns: the name (not `null`)

---

hashCode()

```
public int hashCode()
```

Returns a hash code value for this object.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: a hash code value

---

setPort(int)

```
public void setPort(int port)
```

Sets the port number of the LDAP server.

Parameters:

`port` - the port number

---

setServerName(String)

```
public void setServerName(java.lang.String serverName)
```

Sets the DNS name of the LDAP server.

Parameters:
`serverName` - the name

Throws:
`NullPointerException` - if `serverName` is `null`

---

toString()

```
public java.lang.String toString()
```

Returns a formatted string describing the parameters.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a formatted string describing the parameters

java.security.cert
PKIXBuilderParameters

Syntax

```
public class PKIXBuilderParameters extends PKIXParameters112

java.lang.Object
  |
  +--PKIXParameters112
        |
        +--java.security.cert.PKIXBuilderParameters
```

All Implemented Interfaces: CertPathParameters53,java.lang.Cloneable

Description

Parameters used as input for the PKIX `CertPathBuilder` algorithm.

A PKIX `CertPathBuilder` uses these parameters to `build(CertPathParameters)`42 a `CertPath` which has been validated according to the PKIX certification path validation algorithm.

To instantiate a `PKIXBuilderParameters` object, an application must specify the *most-trusted CA* as defined by the PKIX certification path validation algorithm. The most-trusted CA can be specified using one of several constructors. First, an application can call either `PKIXBuilderParameters(PublicKey, String, CertSelector)`97 or `PKIXBuilderParameters(PublicKey, String, boolean[], CertSelector)`97 , specifying the public key, distinguished name, and optionally the subject unique identifier of the most-trusted CA. Alternatively, if more than one CA is trusted, an application can call `PKIXBuilderParameters(Set, CertSelector)`98 , specifying a `Set` of trusted `X509Certificate`s. Finally, an application can call `PKIXBuilderParameters(KeyStore, CertSelector)`97 , specifying a `KeyStore` instance containing trusted certificate entries, each of which will be considered as a most-trusted CA.

In addition, an application must specify constraints on the target certificate that the `CertPathBuilder` will attempt to build a path to. The constraints are specified as a `CertSelector` object. These constraints should provide the `CertPathBuilder` with enough search criteria to find the target certificate. Minimal criteria for an `X509Certificate` usually include the subject name and/or one or more subject alternative names. If enough criteria is not specified, the `CertPathBuilder` may throw a `CertPathBuilderException`.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: `CertPathBuilder`41

Member Summary

Constructors

public	PKIXBuilderParameters(KeyStore, CertSelector)$_{97}$
	Creates an instance of `PKIXBuilderParameters` that populates the set of most-trusted CA certificates from the trusted certificate entries contained in the specified `KeyStore`.
public	PKIXBuilderParameters(PublicKey, String, boolean[], CertSelector)$_{97}$
	Creates an instance of `PKIXBuilderParameters` with the specified parameter values.
public	PKIXBuilderParameters(PublicKey, String, CertSelector)$_{97}$
	Creates an instance of `PKIXBuilderParameters` where the most-trusted CA is specified as a public key and distinguished name.
public	PKIXBuilderParameters(Set, CertSelector)$_{98}$
	Creates an instance of `PKIXBuilderParameters` with the specified `Set` of most-trusted CA certificate(s).

Methods

public boolean	equals(Object)$_{98}$
	Compares this object for equality with the specified object.
public int	getMaxPathLength()$_{98}$
	Returns the value of the maximum number of CA certificates that may exist in a certification path.
public int	hashCode()$_{99}$
	Returns a hash code value for this object.
public void	setMaxPathLength(int)$_{99}$
	Sets the value of the maximum number of CA certificates that may exist in a certification path.
public String	toString()$_{99}$
	Returns a formatted string describing the parameters.

Inherited Member Summary

Methods inherited from class PKIXParameters$_{112}$

getTrustedCerts()$_{119}$, setTrustedCerts(Set)$_{123}$, getInitialPolicies()$_{118}$, setInitialPolicies(Set)$_{122}$, getPublicKey()$_{118}$, setCAPublicKeyAndName(PublicKey, String)$_{120}$, setCAPublicKeyAndName(PublicKey, String, boolean[])$_{120}$, getCAName()$_{117}$, setCertStores(List)$_{121}$, addCertStore(CertStore)$_{117}$, getCertStores()$_{117}$, getInitialUniqueID()$_{118}$, setRevocationEnabled(boolean)$_{123}$, isRevocationEnabled()$_{120}$, setExplicitPolicyRequired(boolean)$_{122}$, isExplicitPolicyRequired()$_{119}$, setPolicyMappingInhibited(boolean)$_{122}$, isPolicyMappingInhibited()$_{119}$, setAnyPolicyInhibited(boolean)$_{120}$, isAnyPolicyInhibited()$_{119}$, setPolicyQualifiersRejected(boolean)$_{122}$, getPolicyQualifiersRejected()$_{118}$, getDate()$_{118}$, setDate(Date)$_{121}$, setCertPathCheckers(List)$_{121}$, getCertPathCheckers()$_{117}$, addCertPathChecker(PKIXCertPathChecker)$_{116}$, getSigProvider()$_{119}$, setSigProvider(String)$_{123}$, getTargetCertConstraints()$_{119}$, setTargetCertConstraints(CertSelector)$_{123}$, clone()$_{117}$

Inherited Member Summary
Methods inherited from class java.lang.Object
`getClass, notify, notifyAll, wait, wait, wait, finalize`

Constructors

PKIXBuilderParameters(KeyStore, CertSelector)

```
public PKIXBuilderParameters(java.security.KeyStore keystore,
            CertSelector[66] targetConstraints)
```

Creates an instance of `PKIXBuilderParameters` that populates the set of most-trusted CA certificates from the trusted certificate entries contained in the specified `KeyStore`. Only keystore entries that contain trusted `X509Certificates` are considered; all other certificate types are ignored.

Parameters:

`keystore` - A `KeyStore` from which the set of most-trusted CA certificates will be populated.

`targetConstraints` - a `CertSelector` specifying the constraints on the target certificate

Throws:

`KeyStoreException` - if the keystore has not been initialized

`NullPointerException` - if the keystore is `null`

PKIXBuilderParameters(PublicKey, String, boolean[], CertSelector)

```
public PKIXBuilderParameters(java.security.PublicKey pubKey, java.lang.String caName,
            boolean[] uniqueID, CertSelector[66] targetConstraints)
```

Creates an instance of `PKIXBuilderParameters` with the specified parameter values.

Parameters:

`pubKey` - the public key of the most-trusted CA

`caName` - the distinguished name of the most-trusted CA in RFC 2253 `String` format

`uniqueID` - the unique identifier of the most-trusted CA. The array is cloned to protect against subsequent modifications.

`targetConstraints` - a `CertSelector` specifying the constraints on the target certificate

Throws:

`IOException` - if an error occurs parsing the `caName`

PKIXBuilderParameters(PublicKey, String, CertSelector)

```
public PKIXBuilderParameters(java.security.PublicKey pubKey, java.lang.String caName,
            CertSelector[66] targetConstraints)
```

Creates an instance of `PKIXBuilderParameters` where the most-trusted CA is specified as a public key and distinguished name. The `caName` parameter should contain an X.500 distinguished name in RFC 2253 `String` format.

Parameters:

`pubKey` - the public key of the most-trusted CA

`caName` - the X.500 distinguished name of the most-trusted CA

`targetConstraints` - a `CertSelector` specifying the constraints on the target certificate

Throws:

`IOException` - if an error occurs parsing the `caName`

PKIXBuilderParameters(Set, CertSelector)

```
public PKIXBuilderParameters(java.util.Set trustedCerts,
            CertSelector[66] targetConstraints)
```

Creates an instance of `PKIXBuilderParameters` with the specified `Set` of most-trusted CA certificate(s). Each element of the set is a trusted `X509Certificate`.

Note that the `Set` is copied to protect against subsequent modifications.

Parameters:

`trustedCerts` - a `Set` of `X509Certificate`s

`targetConstraints` - a `CertSelector` specifying the constraints on the target certificate

Throws:

`ClassCastException` - if any of the elements in the set are not of type `java.security.cert.X509Certificate`

Methods

---

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this object for equality with the specified object. If the other object is not an `PKIXBuilderParameters`, return `false`. Otherwise, return `true` if the parameters of the objects are equal.

Overrides: `equals(Object)`[117] in class `PKIXParameters`[112]

Parameters:

`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

---

getMaxPathLength()

```
public int getMaxPathLength()
```

Returns the value of the maximum number of CA certificates that may exist in a certification path.

Returns: the maximum number of CA certificates that may exist in a certification path, or -1 if there is no limit

---

hashCode()

```
public int hashCode()
```

Returns a hash code value for this object.

Overrides: `hashCode()`$_{119}$ in class `PKIXParameters`$_{112}$

Returns: a hash code value

---

setMaxPathLength(int)

```
public void setMaxPathLength(int maxPathLength)
```

Sets the value of the maximum number of CA certificates that may exist in a certification path. A PKIX `CertPathBuilder` instance must not build paths longer than the length specified.

A value of 0 implies that the path can only contain a single end-entity certificate. A value of -1 implies that the path length is unconstrained (i.e. there is no maximum). The default maximum path length, if not specified, is 5. Setting a value less than -1 will cause an exception to be thrown.

If any of the CA certificates contain the `BasicConstraintsExtension`, the value of the `pathLenConstraint` field of the extension overrides the maximum path length parameter whenever the result is a certification path of smaller length.

Parameters:
`maxPathLength` - the maximum number of CA certificates that may exist in a certification path

Throws:
`InvalidAlgorithmParameterException` - if the maximum path length parameter is set to a value less than -1

---

toString()

```
public java.lang.String toString()
```

Returns a formatted string describing the parameters.

Overrides: `toString()`$_{124}$ in class `PKIXParameters`$_{112}$

Returns: a formatted string describing the parameters

java.security.cert
PKIXCertPathBuilderResult

Syntax

```
public class PKIXCertPathBuilderResult extends PKIXCertPathValidatorResult108 implements
    CertPathBuilderResult49

java.lang.Object
  |
  +--PKIXCertPathValidatorResult108
        |
        +--java.security.cert.PKIXCertPathBuilderResult
```

All Implemented Interfaces:

CertPathBuilderResult49,CertPathValidatorResult63,java.lang.Cloneable

Description

This class represents the successful result of the PKIX certification path builder algorithm. All certification paths that are built and returned using this algorithm are also validated according to the PKIX certification path validation algorithm.

Instances of `PKIXCertPathBuilderResult` are returned by the `build` method of `CertPathBuilder` objects implementing the PKIX algorithm.

All `PKIXCertPathBuilderResult` objects contain the certification path constructed by the build algorithm, the valid policy tree and subject public key resulting from the build algorithm, and the certificate containing the public key of the “most-trusted CA” that was used as a trust anchor by the build algorithm.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: CertPathBuilderResult49

Member Summary

Constructors

public	PKIXCertPathBuilderResult(CertPath, Certificate, PolicyNode, PublicKey)101

Creates an instance of `PKIXCertPathBuilderResult` containing the specified parameters.

Methods

Member Summary	
public Object	clone()[102] Returns a copy of this object.
public boolean	equals(Object)[102] Compares this object for equality with the specified object.
public CertPath	getCertPath()[102] Returns the built and validated certification path.
public int	hashCode()[102] Returns a hash code value for this object.
public String	toString()[102] Return a printable representation of this PKIXCertPathBuilderResult.

Inherited Member Summary
Methods inherited from class PKIXCertPathValidatorResult[108]
getTrustedCert()[110], getPolicyTree()[110], getPublicKey()[110]
Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait, finalize

Constructors

PKIXCertPathBuilderResult(CertPath, Certificate, PolicyNode, PublicKey)

```
public PKIXCertPathBuilderResult(CertPath[37] certPath, Certificate[8] trustedCert,
            PolicyNode[125] policyTree, java.security.PublicKey subjectPublicKey)
```

Creates an instance of PKIXCertPathBuilderResult containing the specified parameters.

Parameters:

certPath - the validated CertPath

trustedCert - the Certificate containing the public key of the most-trusted CA. Specify null when the public key is not supplied in the form of a certificate.

policyTree - the valid policy tree, or null if there are no valid policies

subjectPublicKey - the public key of the subject

Throws:

NullPointerException - if certPath or subjectPublicKey are null

Methods

---

clone()

```
public java.lang.Object clone()
```

Returns a copy of this object.

Specified By: `clone()`[102] in interface `PKIXCertPathBuilderResult`[100]

Specified By: `clone()`[102] in interface `PKIXCertPathBuilderResult`[100]

Overrides: `clone()`[110] in class `PKIXCertPathValidatorResult`[108]

Returns: the copy

---

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this object for equality with the specified object. If the other object is not an instance of `PKIXCertPathBuilderResult`, return `false`. Otherwise, return `true` if the parameters of the objects are equal.

Overrides: `equals(Object)`[110] in class `PKIXCertPathValidatorResult`[108]

Parameters:
`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

---

getCertPath()

```
public CertPath[37] getCertPath()
```

Returns the built and validated certification path. The `CertPath` object does not include the certificate containing the most-trusted CA's public key. Instead, use the `getTrustedCert()`[110] method to obtain the most-trusted CA certificate.

Specified By: `getCertPath()`[102] in interface `PKIXCertPathBuilderResult`[100]

Returns: the built and validated `CertPath` (never `null`)

---

hashCode()

```
public int hashCode()
```

Returns a hash code value for this object.

Overrides: `hashCode()`[110] in class `PKIXCertPathValidatorResult`[108]

Returns: a hash code value

---

toString()

```
public java.lang.String toString()
```

Return a printable representation of this `PKIXCertPathBuilderResult`.

Overrides: `toString()`[111] in class `PKIXCertPathValidatorResult`[108]

Returns: a `String` describing the contents of this `PKIXCertPathBuilderResult`

java.security.cert
PKIXCertPathChecker

Syntax

```
public abstract class PKIXCertPathChecker implements java.lang.Cloneable
```

```
java.lang.Object
  |
  +--java.security.cert.PKIXCertPathChecker
```

All Implemented Interfaces: java.lang.Cloneable

Description

An abstract class that performs one or more checks on an `X509Certificate`.

A concrete implementation of the `PKIXCertPathChecker` class can be created to extend the PKIX certification path validation algorithm. For example, an implementation may check for and process a critical private extension of each certificate in a certification path.

Instances of `PKIXCertPathChecker` are passed as parameters using the `setCertPathCheckers(List)`$_{121}$ or `addCertPathChecker(PKIXCertPathChecker)`$_{116}$ methods of the `PKIXParameters` and `PKIXBuilderParameters` class. Each of the `PKIXCertPathCheckers` `check(Certificate, Collection)`$_{106}$ methods will be called, in turn, for each certificate processed by a PKIX `CertPathValidator` or `CertPathBuilder` implementation.

A `PKIXCertPathChecker` may be called multiple times on successive certificates in a certification path. Concrete subclasses are expected to maintain any internal state that may be necessary to check successive certificates. The `init(boolean)`$_{106}$ method is used to initialize the internal state of the checker so that the certificates of a new certification path may be checked. A stateful implementation **must** override the `clone()`$_{106}$ method if necessary in order to allow a PKIX `CertPathBuilder` to efficiently backtrack and try other paths. In these situations, the `CertPathBuilder` is able to restore prior path validation states by restoring the cloned `PKIXCertPathCheckers`.

The order in which the certificates are presented to the `PKIXCertPathChecker` may be either in the forward direction (from target to most-trusted CA) or in the reverse direction (from most-trusted CA to target). A `PKIXCertPathChecker` implementation **must** support reverse checking (the ability to perform its checks when it is presented with certificates in the reverse direction) and **may** support forward checking (the ability to perform its checks when it is presented with certificates in the forward direction). The `isForwardCheckingSupported()`$_{107}$ method indicates whether forward checking is supported.

Additional input parameters required for executing the check may be specified through constructors of concrete implementations of this class.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: `PKIXParameters`$_{112}$, `PKIXBuilderParameters`$_{95}$

Member Summary

Constructors

protected	`PKIXCertPathChecker()`$_{105}$ Default constructor

Methods

public abstract void	`check(Certificate, Collection)`$_{106}$ Performs the check(s) on the specified certificate using its internal state and removes any critical extensions that it processes from the specified collection of OID strings that represent the unresolved critical extensions.
public Object	`clone()`$_{106}$ Returns a clone of this object.
public abstract Set	`getSupportedExtensions()`$_{106}$ Returns an immutable `Set` of X.509 certificate extensions that this `PKIXCertPathChecker` supports (i.e.
public abstract void	`init(boolean)`$_{106}$ Initializes the internal state of this `PKIXCertPathChecker`.
public abstract boolean	`isForwardCheckingSupported()`$_{107}$ Indicates if forward checking is supported.

Inherited Member Summary

Methods inherited from class java.lang.Object

`getClass, hashCode, equals, toString, notify, notifyAll, wait, wait, wait, finalize`

Constructors

PKIXCertPathChecker()

`protected` **`PKIXCertPathChecker`**`()`

Default constructor

Methods

check(Certificate, Collection)

```
public abstract void check(Certificate[8] cert, java.util.Collection unresolvedCritExts)
```

Performs the check(s) on the specified certificate using its internal state and removes any critical extensions that it processes from the specified collection of OID strings that represent the unresolved critical extensions. The certificates are presented in the order specified by the `init` method.

Parameters:

`cert` - the `Certificate` to be checked

`unresolvedCritExts` - a `Collection` of OID strings representing the current set of unresolved critical extensions

Throws:

`CertPathValidatorException`[58] - if the specified certificate does not pass the check

clone()

```
public java.lang.Object clone()
```

Returns a clone of this object. Calls the `Object.clone()` method. All subclasses which maintain state must support and override this method, if necessary.

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: a copy of this `PKIXCertPathChecker`

getSupportedExtensions()

```
public abstract java.util.Set getSupportedExtensions()
```

Returns an immutable `Set` of X.509 certificate extensions that this `PKIXCertPathChecker` supports (i.e. recognizes, is able to process), or `null` if no extensions are supported.

Each element of the set is a `String` representing the Object Identifier (OID) of the X.509 extension that is supported. The OID is represented by a set of nonnegative integers separated by periods.

All X.509 certificate extensions that a `PKIXCertPathChecker` might possibly be able to process should be included in the set.

Returns: an immutable `Set` of X.509 extension OIDs (in `String` format) supported by this `PKIXCertPathChecker`, or `null` if no extensions are supported

init(boolean)

```
public abstract void init(boolean forward)
```

Initializes the internal state of this `PKIXCertPathChecker`.

The `forward` flag specifies the order that certificates will be passed to the `check(Certificate, Collection)`[106] method (forward or reverse). A `PKIXCertPathChecker` **must** support reverse checking and **may** support forward checking.

Parameters:

`forward` - the order that certificates are presented to the `check` method. If `true`, certificates are presented from target to most-trusted CA (forward); if `false`, from most-trusted CA to target (reverse).

Throws:

`CertPathValidatorException`[58] - if this `PKIXCertPathChecker` is unable to check certificates in the specified order; it should never be thrown if the forward flag is false since reverse checking must be supported

---

isForwardCheckingSupported()

```
public abstract boolean isForwardCheckingSupported()
```

Indicates if forward checking is supported. Forward checking refers to the ability of the `PKIXCertPathChecker` to perform its checks when certificates are presented to the `check` method in the forward direction (from target to most-trusted CA).

Returns: `true` if forward checking is supported, `false` otherwise

java.security.cert
PKIXCertPathValidatorResult

Syntax

```
public class PKIXCertPathValidatorResult implements CertPathValidatorResult63

java.lang.Object
  |
  +--java.security.cert.PKIXCertPathValidatorResult
```

Direct Known Subclasses: PKIXCertPathBuilderResult100

All Implemented Interfaces: CertPathValidatorResult63,java.lang.Cloneable

Description

This class represents the successful result of the PKIX certification path validation algorithm.

Instances of `PKIXCertPathValidatorResult` are returned by the `validate(CertPath, CertPathParameters)`57 method of `CertPathValidator` objects implementing the PKIX algorithm.

All `PKIXCertPathValidatorResult` objects contain the valid policy tree and subject public key resulting from the validation algorithm, as well as the certificate containing the public key of the "most-trusted CA" that was used as a trust anchor by the validation algorithm.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: CertPathValidatorResult63

Member Summary

Constructors

public	PKIXCertPathValidatorResult(Certificate, PolicyNode, PublicKey)109 Creates an instance of `PKIXCertPathValidatorResult` containing the specified parameters.

Methods

public Object	clone()110 Returns a copy of this object.
public boolean	equals(Object)110 Compares this object for equality with the specified object.

Member Summary	
public PolicyNode	`getPolicyTree()`[110] Returns the valid policy tree resulting from the PKIX certification path validation algorithm.
public PublicKey	`getPublicKey()`[110] Returns the public key of the subject (target) of the certification path, including any inherited public key parameters if applicable.
public Certificate	`getTrustedCert()`[110] Returns the certificate containing the public key of the most-trusted CA.
public int	`hashCode()`[110] Returns a hash code value for this object.
public String	`toString()`[111] Return a printable representation of this `PKIXCertPathValidatorResult`.

Inherited Member Summary
Methods inherited from class java.lang.Object
`getClass, notify, notifyAll, wait, wait, wait, finalize`

Constructors

PKIXCertPathValidatorResult(Certificate, PolicyNode, PublicKey)

```
public PKIXCertPathValidatorResult(Certificate[8] trustedCert, PolicyNode[125] policyTree,
            java.security.PublicKey subjectPublicKey)
```

Creates an instance of `PKIXCertPathValidatorResult` containing the specified parameters.

Parameters:

`trustedCert` - - the `Certificate` containing the public key of the most-trusted CA. Specify `null` when the public key is not supplied in the form of a certificate.

`policyTree` - the valid policy tree, or `null` if there are no valid policies

`subjectPublicKey` - the public key of the subject

Throws:

`NullPointerException` - if `subjectPublicKey` is `null`

Methods

---

clone()

```
public java.lang.Object clone()
```

Returns a copy of this object.

Specified By: `clone()`[110] in interface `PKIXCertPathValidatorResult`[108]

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: the copy

---

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this object for equality with the specified object. If the other object is not an instance of `PKIXCertPathValidatorResult`, return `false`. Otherwise, return `true` if the parameters of the objects are equal.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:
`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

---

getPolicyTree()

```
public PolicyNode[125] getPolicyTree()
```

Returns the valid policy tree resulting from the PKIX certification path validation algorithm. The `PolicyNode` object that is returned and any field that it returns through public methods is immutable.

Returns: the root node of the valid policy tree, or `null` if there are no valid policies

---

getPublicKey()

```
public java.security.PublicKey getPublicKey()
```

Returns the public key of the subject (target) of the certification path, including any inherited public key parameters if applicable.

Returns: the public key of the subject (never `null`)

---

getTrustedCert()

```
public Certificate[8] getTrustedCert()
```

Returns the certificate containing the public key of the most-trusted CA. Returns `null` when the trusted public key is not supplied in the form of a certificate.

Returns: a `Certificate` containing the public key of the most-trusted CA (or `null`)

---

hashCode()

```
public int hashCode()
```

Returns a hash code value for this object.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: a hash code value

---

toString()

```
public java.lang.String toString()
```

Return a printable representation of this `PKIXCertPathValidatorResult`.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a `String` describing the contents of this `PKIXCertPathValidatorResult`

java.security.cert
PKIXParameters

Syntax

```
public class PKIXParameters implements CertPathParameters53

java.lang.Object
  |
  +--java.security.cert.PKIXParameters
```

Direct Known Subclasses: PKIXBuilderParameters95

All Implemented Interfaces: CertPathParameters53,java.lang.Cloneable

Description

Parameters used as input for the PKIX `CertPathValidator` algorithm.

A PKIX `CertPathValidator` uses these parameters to validate a `CertPath` according to the PKIX certification path validation algorithm.

To instantiate a `PKIXParameters` object, an application must specify the *most-trusted CA* as defined by the PKIX certification path validation algorithm. The most-trusted CA can be specified using one of several constructors. First, an application can call either PKIXParameters(PublicKey, String)115 or PKIXParameters(PublicKey, String, boolean[])116 , specifying the public key, distinguished name, and optionally the subject unique identifier of the most-trusted CA. Alternatively, if more than one CA is trusted, an application can call PKIXParameters(Set)116 , specifying a `Set` of trusted `X509Certificate`s. Finally, an application can call PKIXParameters(KeyStore)115 , specifying a `KeyStore` instance containing trusted certificate entries, each of which will be considered as a most-trusted CA.

Once a `PKIXParameters` object has been created, other parameters can be specified (by calling setInitialPolicies(Set)122 or setDate(Date)121 , for instance) and then the `PKIXParameters` is passed along with the `CertPath` to be validated to validate(CertPath, CertPathParameters)57 .

Any parameter that is not set (or is set to `null`) will be set to the default value for that parameter. The default value for the `date` parameter is `null`, which indicates the current time when the path is validated. The default for the remaining parameters is the least constrained.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: CertPathValidator54

Member Summary

Constructors

`public` `PKIXParameters(KeyStore)`[115]

Creates an instance of `PKIXParameters` that populates the set of most-trusted CA certificates from the trusted certificate entries contained in the specified `KeyStore`.

`public` `PKIXParameters(PublicKey, String)`[115]

Creates an instance of `PKIXParameters` where the most-trusted CA is specified as a public key and distinguished name.

`public` `PKIXParameters(PublicKey, String, boolean[])`[116]

Creates an instance of `PKIXParameters` with the specified parameter values.

`public` `PKIXParameters(Set)`[116]

Creates an instance of `PKIXParameters` with the specified `Set` of most-trusted CA certificates.

Methods

`public void` `addCertPathChecker(PKIXCertPathChecker)`[116]

Adds a `PKIXCertPathChecker` to the list of certification path checkers.

`public void` `addCertStore(CertStore)`[117]

Adds a `CertStore` to the end of the list of `CertStore`s used in finding certificates and CRLs.

`public Object` `clone()`[117]

Makes a copy of this `PKIXParameters` object.

`public boolean` `equals(Object)`[117]

Compares this object for equality with the specified object.

`public String` `getCAName()`[117]

Returns the name of the most-trusted CA in RFC 2253 `String` format.

`public List` `getCertPathCheckers()`[117]

Returns the `List` of certification path checkers.

`public List` `getCertStores()`[117]

Returns an immutable `List` of `CertStore`s that are used to find certificates and CRLs.

`public Date` `getDate()`[118]

Returns the time for which the validity of the certification path should be determined.

`public Set` `getInitialPolicies()`[118]

Returns an immutable `Set` of initial policy identifiers (OID strings), indicating that any one of these policies would be acceptable to the certificate user for the purposes of certification path processing.

`public boolean` `getInitialUniqueID()`[118]

Returns the unique identifier of the most-trusted CA, or `null` if not set.

`public boolean` `getPolicyQualifiersRejected()`[118]

Gets the PolicyQualifiersRejected flag.

`public PublicKey` `getPublicKey()`[118]

Returns the public key of the most-trusted CA.

Member Summary

public String	`getSigProvider()`[119] Returns the signature provider's name, or `null` if not set.
public CertSelector	`getTargetCertConstraints()`[119] Returns the required constraints on the target certificate.
public Set	`getTrustedCerts()`[119] Returns an immutable `Set` of the most-trusted CA certificates.
public int	`hashCode()`[119] Returns a hash code value for this object.
public boolean	`isAnyPolicyInhibited()`[119] Checks whether the any policy OID should be processed if it is included in a certificate.
public boolean	`isExplicitPolicyRequired()`[119] Checks if explicit policy is required.
public boolean	`isPolicyMappingInhibited()`[119] Checks if policy mapping is inhibited.
public boolean	`isRevocationEnabled()`[120] Checks the RevocationEnabled flag.
public void	`setAnyPolicyInhibited(boolean)`[120] Sets state to determine if the any policy OID should be processed if it is included in a certificate.
public void	`setCAPublicKeyAndName(PublicKey, String)`[120] Sets the public key and name of the most-trusted CA.
public void	`setCAPublicKeyAndName(PublicKey, String, boolean[])`[120] Sets the public key, name and unique identifier of the CA.
public void	`setCertPathCheckers(List)`[121] Sets a `List` of additional certification path checkers.
public void	`setCertStores(List)`[121] Sets the list of `CertStore`s to be used in finding certificates and CRLs.
public void	`setDate(Date)`[121] Sets the time for which the validity of the certification path should be determined.
public void	`setExplicitPolicyRequired(boolean)`[122] Sets the ExplicitPolicyRequired flag.
public void	`setInitialPolicies(Set)`[122] Sets the `Set` of initial policy identifiers (OID strings), indicating that any one of these policies would be acceptable to the certificate user for the purposes of certification path processing.
public void	`setPolicyMappingInhibited(boolean)`[122] Sets the PolicyMappingInhibited flag.
public void	`setPolicyQualifiersRejected(boolean)`[122] Sets the PolicyQualifiersRejected flag.

Member Summary	
public void	setRevocationEnabled(boolean)[123] Sets the RevocationEnabled flag.
public void	setSigProvider(String)[123] Sets the signature provider's name.
public void	setTargetCertConstraints(CertSelector)[123] Sets the required constraints on the target certificate.
public void	setTrustedCerts(Set)[123] Sets the Set of most-trusted CA certificates.
public String	toString()[124] Returns a formatted string describing the parameters.

Inherited Member Summary
Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait, finalize

Constructors

PKIXParameters(KeyStore)

```
public PKIXParameters(java.security.KeyStore keystore)
```

Creates an instance of PKIXParameters that populates the set of most-trusted CA certificates from the trusted certificate entries contained in the specified KeyStore. Only keystore entries that contain trusted X509Certificates are considered; all other certificate types are ignored.

Parameters:

keystore - a KeyStore from which the set of most-trusted CA certificates will be populated

Throws:

KeyStoreException - if the keystore has not been initialized.

NullPointerException - if the keystore is null

PKIXParameters(PublicKey, String)

```
public PKIXParameters(java.security.PublicKey pubKey, java.lang.String caName)
```

Creates an instance of PKIXParameters where the most-trusted CA is specified as a public key and distinguished name.

Parameters:

pubKey - the public key of the most-trusted CA

`caName` - the X.500 distinguished name of the most-trusted CA in RFC 2253 `String` format

Throws:

`IOException` - if an error occurs parsing the `caName`

PKIXParameters(PublicKey, String, boolean[])

```
public PKIXParameters(java.security.PublicKey pubKey, java.lang.String caName,
            boolean[] uniqueID)
```

Creates an instance of `PKIXParameters` with the specified parameter values.

Parameters:

`pubKey` - the public key of the most-trusted CA

`caName` - the X.500 distinguished name of the most-trusted CA in RFC 2253 `String` format

`uniqueID` - the unique identifier of the most-trusted CA subject unique identifier. The array is cloned to protect against subsequent modifications.

Throws:

`IOException` - if an error occurs parsing the `caName`

PKIXParameters(Set)

```
public PKIXParameters(java.util.Set trustedCerts)
```

Creates an instance of `PKIXParameters` with the specified `Set` of most-trusted CA certificates. Each element of the set is a a trusted `X509Certificiate`.

Note that the `Set` is copied to protect against subsequent modifications.

Parameters:

`trustedCerts` - a `Set` of `X509Certificate`s

Throws:

`ClassCastException` - if any of the elements in the set are not of type `java.security.cert.X509Certificate`

Methods

---

addCertPathChecker(PKIXCertPathChecker)

```
public void addCertPathChecker(PKIXCertPathChecker104 checker)
```

Adds a `PKIXCertPathChecker` to the list of certification path checkers. See the `setCertPathCheckers(List)`121 method for more details.

Note that the `PKIXCertPathChecker` is cloned to protect against subsequent modifications.

Parameters:

`checker` - a `PKIXCertPathChecker` to add to the list of checks. If `null`, the checker is ignored (not added to list).

addCertStore(CertStore)

```
public void addCertStore(CertStore store)
```

Adds a `CertStore` to the end of the list of `CertStore`s used in finding certificates and CRLs.

Parameters:
`store` - the `CertStore` to add. If `null`, the store is ignored (not added to list).

clone()

```
public java.lang.Object clone()
```

Makes a copy of this `PKIXParameters` object. Changes to the copy will not affect the original and vice versa.

Specified By: `clone()` in interface `PKIXParameters`

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: a copy of this `PKIXParameters` object

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this object for equality with the specified object. If the other object is not an `PKIXParameters`, return `false`. Otherwise, return `true` if the parameters of the objects are equal.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:
`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

getCAName()

```
public java.lang.String getCAName()
```

Returns the name of the most-trusted CA in RFC 2253 `String` format.

Returns: the X.500 distinguished name of the most-trusted CA, or `null` if not set

getCertPathCheckers()

```
public java.util.List getCertPathCheckers()
```

Returns the `List` of certification path checkers. The returned `List` is immutable, and each `PKIXCertPathChecker` in the `List` is cloned to protect against subsequent modifications.

Returns: an immutable `List` of `PKIXCertPathChecker`s (may be empty, but not `null`)

getCertStores()

```
public java.util.List getCertStores()
```

Returns an immutable `List` of `CertStore`s that are used to find certificates and CRLs.

Returns: an immutable `List` of `CertStore`s (may be empty, but never `null`)

---

getDate()

`public java.util.Date getDate()`

Returns the time for which the validity of the certification path should be determined. If `null`, the current time is used.

Note that the `Date` returned is copied to protect against subsequent modifications.

Returns: the `Date`, or `null` if not set

---

getInitialPolicies()

`public java.util.Set getInitialPolicies()`

Returns an immutable `Set` of initial policy identifiers (OID strings), indicating that any one of these policies would be acceptable to the certificate user for the purposes of certification path processing. The default return value is an empty `Set`, which is interpreted as meaning that any policy would be acceptable.

Returns: an immutable `Set` of initial policy OIDs in `String` format, or an empty `Set` (implying any policy is acceptable). Never returns `null`.

---

getInitialUniqueID()

`public boolean[] getInitialUniqueID()`

Returns the unique identifier of the most-trusted CA, or `null` if not set. Note that the array returned is copied to protect against subsequent modifications.

Returns: a boolean array containing the unique identifier of the most-trusted CA, or `null` if not set

---

getPolicyQualifiersRejected()

`public boolean getPolicyQualifiersRejected()`

Gets the PolicyQualifiersRejected flag. If this flag is true, certificates that include policy qualifiers are rejected. If the flag is false, certificates are not rejected on this basis.

When a `PKIXParameters` object is created, this flag is set to true. This setting reflects the most common (and simplest) strategy for processing policy qualifiers. Applications that want to use a more sophisticated policy must set this flag to false.

Returns: the current value of the PolicyQualifiersRejected flag

---

getPublicKey()

`public java.security.PublicKey getPublicKey()`

Returns the public key of the most-trusted CA.

Returns: the public key of the most-trusted CA, or `null` if not set

getSigProvider()

```
public java.lang.String getSigProvider()
```

Returns the signature provider's name, or `null` if not set.

Returns: the signature provider's name (or `null`)

getTargetCertConstraints()

```
public CertSelector[66] getTargetCertConstraints()
```

Returns the required constraints on the target certificate. The constraints are returned as an instance of `CertSelector`. If `null`, no constraints are defined.

Note that the `CertSelector` returned is cloned to protect against subsequent modifications.

Returns: a `CertSelector` specifying the constraints on the target certificate (or `null`)

getTrustedCerts()

```
public java.util.Set getTrustedCerts()
```

Returns an immutable `Set` of the most-trusted CA certificates.

Returns: an immutable `Set` of `X509Certificate`s or `null` if not specified

hashCode()

```
public int hashCode()
```

Returns a hash code value for this object.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: a hash code value

isAnyPolicyInhibited()

```
public boolean isAnyPolicyInhibited()
```

Checks whether the any policy OID should be processed if it is included in a certificate.

Returns: `true` if the any policy OID is inhibited, `false` otherwise

isExplicitPolicyRequired()

```
public boolean isExplicitPolicyRequired()
```

Checks if explicit policy is required. If this flag is true, an acceptable policy needs to be explicitly identified in every certificate. By default, the ExplicitPolicyRequired flag is false.

Returns: `true` if explicit policy is required, `false` otherwise

isPolicyMappingInhibited()

```
public boolean isPolicyMappingInhibited()
```

Checks if policy mapping is inhibited. If this flag is true, policy mapping is inhibited. By default, policy mapping is not inhibited (the flag is false).

Returns: true if policy mapping is inhibited, false otherwise

---

isRevocationEnabled()

```
public boolean isRevocationEnabled()
```

Checks the RevocationEnabled flag. If this flag is true, the default revocation checking mechanism of the underlying PKIX service provider will be used. If this flag is false, the default revocation checking mechanism will be disabled (not used). See the `setRevocationEnabled(boolean)`[123] method for more details on setting the value of this flag.

Returns: the current value of the RevocationEnabled flag

---

setAnyPolicyInhibited(boolean)

```
public void setAnyPolicyInhibited(boolean val)
```

Sets state to determine if the any policy OID should be processed if it is included in a certificate. By default, the any policy OID is not inhibited (`isAnyPolicyInhibited()`[119] returns `false`).

Parameters:

`val` - `true` if the any policy OID is to be inhibited, `false` otherwise

---

setCAPublicKeyAndName(PublicKey, String)

```
public void setCAPublicKeyAndName(java.security.PublicKey pubKey,
            java.lang.String caName)
```

Sets the public key and name of the most-trusted CA. Any current value for the `trustedCerts` parameter is cleared (set to `null`).

Parameters:

`pubKey` - the public key of the most-trusted CA

`caName` - the X.500 distinguished name of the most-trusted CA in RFC 2253 `String` format

Throws:

`IOException` - if an error occurs parsing the `caName`

---

setCAPublicKeyAndName(PublicKey, String, boolean[])

```
public void setCAPublicKeyAndName(java.security.PublicKey pubKey,
            java.lang.String caName, boolean[] uniqueID)
```

Sets the public key, name and unique identifier of the CA. Any current value for the `trustedCerts` parameter is cleared (set to `null`).

Parameters:

`pubKey` - the public key of the most-trusted CA

`caName` - the X.500 distinguished name of the most-trusted CA in RFC 2253 `String` format

`uniqueID` - a boolean array containing the unique identifier of the most-trusted CA. Note that the array is copied to protect against subsequent modifications.

Throws:
`IOException` - if an error occurs parsing the `caName`

---

setCertPathCheckers(List)

`public void setCertPathCheckers(java.util.List checkers)`

Sets a `List` of additional certification path checkers. If the specified `List` contains an object that is not a `PKIXCertPathChecker`, it is ignored.

Each `PKIXCertPathChecker` specified implements additional checks on a certificate. Typically, these are checks to process and verify private extensions contained in certificates. Each `PKIXCertPathChecker` should be instantiated with any initialization parameters needed to execute the check.

This method allows sophisticated applications to extend a PKIX `CertPathValidator` or `CertPathBuilder`. Each of the specified `PKIXCertPathChecker`s will be called, in turn, by a PKIX `CertPathValidator` or `CertPathBuilder` for each certificate processed or validated.

Regardless of whether these additional `PKIXCertPathChecker`s are set, a PKIX `CertPathValidator` or `CertPathBuilder` must perform all of the required PKIX checks on each certificate. The one exception to this rule is if the RevocationEnabled flag is set to false (see the `setRevocationEnabled(boolean)`[123] method).

Note that the `List` supplied here is copied and each `PKIXCertPathChecker` in the list is cloned to protect against subsequent modifications.

Parameters:
`checkers` - a `List` of `PKIXCertPathChecker`s. May be `null`, in which case no additional checkers will be used.

Throws:
`ClassCastException` - if any of the elements in the list are not of type `java.security.cert.PKIXCertPathChecker`

---

setCertStores(List)

`public void setCertStores(java.util.List stores)`

Sets the list of `CertStore`s to be used in finding certificates and CRLs. May be `null`, in which case no `CertStore`s will be used. The first `CertStore`s in the list may be preferred to those that appear later.

Note that the `List` is copied to protect against subsequent modifications.

Parameters:
`stores` - a `List` of `CertStore`s (or `null`)

Throws:
`ClassCastException` - if any of the elements in the list are not of type `java.security.cert.CertStore`

---

setDate(Date)

`public void setDate(java.util.Date date)`

Sets the time for which the validity of the certification path should be determined. If `null`, the current time is used.

Note that the `Date` supplied here is copied to protect against subsequent modifications.

Parameters:
`date` - the `Date`, or `null` for the current time

---

setExplicitPolicyRequired(boolean)

```
public void setExplicitPolicyRequired(boolean val)
```

Sets the ExplicitPolicyRequired flag. If this flag is true, an acceptable policy needs to be explicitly identified in every certificate. By default, the ExplicitPolicyRequired flag is false.

Parameters:
`val` - `true` if explicit policy is to be required, `false` otherwise

---

setInitialPolicies(Set)

```
public void setInitialPolicies(java.util.Set initialPolicies)
```

Sets the `Set` of initial policy identifiers (OID strings), indicating that any one of these policies would be acceptable to the certificate user for the purposes of certification path processing. By default, any policy is acceptable (i.e. all policies), so a user that wants to allow any policy as acceptable does not need to call this method, or can call it with an empty `Set` (or `null`).

Note that the `Set` is copied to protect against subsequent modifications.

Parameters:
`initialPolicies` - a `Set` of initial policy OIDs in `String` format (or `null`)

Throws:
`ClassCastException` - if any of the elements in the set are not of type `String`

---

setPolicyMappingInhibited(boolean)

```
public void setPolicyMappingInhibited(boolean val)
```

Sets the PolicyMappingInhibited flag. If this flag is true, policy mapping is inhibited. By default, policy mapping is not inhibited (the flag is false).

Parameters:
`val` - `true` if policy mapping is to be inhibited, `false` otherwise

---

setPolicyQualifiersRejected(boolean)

```
public void setPolicyQualifiersRejected(boolean qualifiersRejected)
```

Sets the PolicyQualifiersRejected flag. If this flag is true, certificates that include policy qualifiers are rejected. If the flag is false, certificates are not rejected on this basis.

When a `PKIXParameters` object is created, this flag is set to true. This setting reflects the most common (and simplest) strategy for processing policy qualifiers. Applications that want to use a more sophisticated policy must set this flag to false.

Note that the PKIX certification path validation algorithm specifies that any policy qualifier in a certificate policies extension that is marked critical must be processed and validated. Otherwise the certification path

must be rejected. If the policyQualifiersRejected flag is set to false, it is up to the application to validate all policy qualifiers in this manner in order to be PKIX compliant.

Parameters:
`qualifiersRejected` - the new value of the PolicyQualifiersRejected flag

See Also: `PolicyQualifierInfo`[128]

---

setRevocationEnabled(boolean)

```
public void setRevocationEnabled(boolean val)
```

Sets the RevocationEnabled flag. If this flag is true, the default revocation checking mechanism of the underlying PKIX service provider will be used. If this flag is false, the default revocation checking mechanism will be disabled (not used).

When a `PKIXParameters` object is created, this flag is set to true. This setting reflects the most common strategy for checking revocation, since each service provider must support revocation checking to be PKIX compliant. Sophisticated applications should set this flag to false when it is not practical to use a PKIX service provider's default revocation checking mechanism or when an alternative revocation checking mechanism is to be substituted (by also calling the `addCertPathChecker(PKIXCertPathChecker)`[116] or `setCertPathCheckers(List)`[121] methods).

Parameters:
`val` - the new value of the RevocationEnabled flag

---

setSigProvider(String)

```
public void setSigProvider(java.lang.String sigProvider)
```

Sets the signature provider's name. The specified provider will be preferred when creating `Signature` objects. If `null` or not set, the first provider found supporting the algorithm will be used.

Parameters:
`sigProvider` - the signature provider's name (or `null`)

---

setTargetCertConstraints(CertSelector)

```
public void setTargetCertConstraints(CertSelector[66] selector)
```

Sets the required constraints on the target certificate. The constraints are specified as an instance of `CertSelector`. If `null`, no constraints are defined.

Note that the `CertSelector` specified is cloned to protect against subsequent modifications.

Parameters:
`selector` - a `CertSelector` specifying the constraints on the target certificate (or `null`)

---

setTrustedCerts(Set)

```
public void setTrustedCerts(java.util.Set trustedCerts)
```

Sets the `Set` of most-trusted CA certificates. Any current values for the `pubKey`, `caName` or `uniqueID` parameters are cleared (set to `null`).

Note that the `Set` is copied to protect against subsequent modifications.

toString()

Parameters:

`trustedCerts` - a `Set` of `X509Certificate`s

Throws:

`ClassCastException` - if any of the elements in the set are not of type `java.security.cert.X509Certificate`

---

toString()

```
public java.lang.String toString()
```

Returns a formatted string describing the parameters.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a formatted string describing the parameters.

java.security.cert
PolicyNode

Syntax
```
public interface PolicyNode
```

Description
A valid policy tree node resulting from the PKIX certification path validation algorithm.

One of the outputs of the PKIX certification path validation algorithm is a valid policy tree, which includes the policies that were determined to be valid, how this determination was reached, and any policy qualifiers encountered. This tree is of depth *n*, where *n* is the length of the certification path that has been validated.

Most applications will not need to examine the valid policy tree. They can achieve their policy processing goals by setting the policy-related parameters in `PKIXParameters`. However, the valid policy tree is available for more sophisticated applications, especially those that process policy qualifiers.

`getPolicyTree()`[110] returns the root node of the valid policy tree. The tree can be traversed using the `getChildren()`[126] and `getParent()`[126] methods. Data about a particular node can be retrieved using other methods of `PolicyNode`.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

Member Summary

Methods

`public Iterator`	`getChildren()`[126] Returns an iterator over the children of this node.
`public int`	`getDepth()`[126] Returns the depth of this node in the valid policy tree.
`public Set`	`getExpectedPolicies()`[126] Returns the set of expected policies that would satisfy this node's valid policy in the next certificate to be processed.
`public PolicyNode`	`getParent()`[126] Returns the parent of this node, or `null` if this is the root node.
`public Set`	`getPolicyQualifiers()`[126] Returns the set of policy qualifiers associated with the valid policy represented by this node.
`public String`	`getValidPolicy()`[127] Returns the valid policy represented by this node.

Member Summary	
public boolean	isCritical()[127] Returns the criticality indicator of the certificate policy extension in the most recently processed certificate.

Methods

getChildren()

`public java.util.Iterator getChildren()`

Returns an iterator over the children of this node. Any attempts to modify the children of this node through the `Iterator`'s remove method must throw an `UnsupportedOperationException`.

Returns: an iterator over the children of this node

getDepth()

`public int getDepth()`

Returns the depth of this node in the valid policy tree.

Returns: the depth of this node (never less than 0)

getExpectedPolicies()

`public java.util.Set getExpectedPolicies()`

Returns the set of expected policies that would satisfy this node's valid policy in the next certificate to be processed.

Returns: an immutable `Set` of expected policy `String` OIDs. For the root node, this is a `Set` with the single value "any-policy".

getParent()

`public PolicyNode[125] getParent()`

Returns the parent of this node, or `null` if this is the root node.

Returns: the parent of this node, or `null` if this is the root node

getPolicyQualifiers()

`public java.util.Set getPolicyQualifiers()`

Returns the set of policy qualifiers associated with the valid policy represented by this node.

Returns: an immutable `Set` of `PolicyQualifierInfo`s. For the root node, this is always an empty `Set`.

getValidPolicy()

```
public java.lang.String getValidPolicy()
```

Returns the valid policy represented by this node.

Returns: the `String` OID of the valid policy represented by this node. For the root node, this is the the special value “any-policy”.

isCritical()

```
public boolean isCritical()
```

Returns the criticality indicator of the certificate policy extension in the most recently processed certificate.

Returns: `true` if extension marked critical, `false` otherwise. For the root node, `false` is always returned.

java.security.cert
PolicyQualifierInfo

Syntax

```
public class PolicyQualifierInfo implements java.lang.Cloneable

java.lang.Object
  |
  +--java.security.cert.PolicyQualifierInfo
```

All Implemented Interfaces: java.lang.Cloneable

Description

A policy qualifier represented by the ASN.1 PolicyQualifierInfo structure.

The ASN.1 definition is as follows:

```
PolicyQualifierInfo ::= SEQUENCE {
     policyQualifierId       PolicyQualifierId,
     qualifier               ANY DEFINED BY policyQualifierId }
```

The certificate policies extension in a certificate contains one or more encoded PolicyQualifierInfo structures, each of which consists of an object identifier (OID) and optional qualifiers. In an end-entity certificate, these qualifiers indicate the policy under which the certificate has been issued and the purposes for which the certificate may be used. In a CA certificate, these qualifiers limit the set of policies for certification paths which include this certificate.

Instances of this class are returned as elements of the `Set` of policy qualifiers returned by the `getPolicyQualifiers()`$_{126}$ method. This allows applications with specific policy requirements to process and validate each policy qualifier. Applications that need to process policy qualifiers should explicitly set the policyQualifiersRejected flag to false (by calling the `setPolicyQualifiersRejected(boolean)`$_{122}$ method) before validating a certification path.

Note that the PKIX certification path validation algorithm specifies that any policy qualifier in a certificate policies extension that is marked critical must be processed and validated. Otherwise the certification path must be rejected. If the policyQualifiersRejected flag is set to false, it is up to the application to validate all policy qualifiers in this manner in order to be PKIX compliant.

Since: 1.4

Member Summary

Constructors

public	`PolicyQualifierInfo(byte[])`$_{129}$
	Creates an instance of `PolicyQualifierInfo` from the encoded bytes.

Methods

public Object	`clone()`$_{129}$
	Returns a copy of this object.

Member Summary	
public boolean	equals(Object)[130] Compares this object for equality with the specified object.
public byte	getEncoded()[130] Returns the ASN.1 DER encoded form of this PolicyQualifierInfo.
public byte	getPolicyQualifier()[130] Returns the ASN.1 DER encoded form of the qualifier component of the PolicyQualifierInfo.
public String	getPolicyQualifierId()[130] Returns the policyQualifier Object Identifier as a String.
public int	hashCode()[130] Returns a hash code value for this object.
public String	toString()[131] Return a printable representation of the PolicyQualifierInfo.

Inherited Member Summary
Methods inherited from class java.lang.Object
getClass, notify, notifyAll, wait, wait, wait, finalize

Constructors

PolicyQualifierInfo(byte[])

```
public PolicyQualifierInfo(byte[] encoded)
```

Creates an instance of PolicyQualifierInfo from the encoded bytes. The encoded byte array is copied on construction.

Parameters:
encoded - a byte array containing the qualifier in DER encoding

Throws:
IOException - thrown if the byte array does not represent a valid and parsable policy qualifier

Methods

clone()

```
public java.lang.Object clone()
```

Returns a copy of this object.

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: the copy

---

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this object for equality with the specified object. If the other object is not an `PolicyQualifierInfo`, return `false`. Otherwise, return `true` if the parameters of the objects are equal.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:
`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

---

getEncoded()

```
public byte[] getEncoded()
```

Returns the ASN.1 DER encoded form of this PolicyQualifierInfo.

Returns: the object in a DER encoded byte array (never `null`). Note that a copy is returned, so the data is cloned each time this method is called.

---

getPolicyQualifier()

```
public byte[] getPolicyQualifier()
```

Returns the ASN.1 DER encoded form of the qualifier component of the PolicyQualifierInfo.

Returns: the data part of the object in a DER encoded byte array. Note that a copy is returned, so the data is cloned each time this method is called.

---

getPolicyQualifierId()

```
public java.lang.String getPolicyQualifierId()
```

Returns the policyQualifier Object Identifier as a String.

Returns: the OID (never `null`)

---

hashCode()

```
public int hashCode()
```

Returns a hash code value for this object.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: a hash code value

toString()

```
public java.lang.String toString()
```

Return a printable representation of the `PolicyQualifierInfo`.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a `String` describing the contents of the `PolicyQualifierInfo`

java.security.cert
X509Certificate

Syntax

```
public abstract class X509Certificate extends Certificate8 implements X509Extension178

java.lang.Object
  |
  +--Certificate8
        |
        +--java.security.cert.X509Certificate
```

All Implemented Interfaces: java.io.Serializable,X509Extension178

Description

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`[8], `CertificateFactory`[20], `X509Extension`[178]

Member Summary

Constructors

protected	`X509Certificate()`[135]
	Constructor for X.509 certificates.

Methods

public abstract void	`checkValidity()`[135]
	Checks that the certificate is currently valid.
public abstract void	`checkValidity(Date)`[135]
	Checks that the given date is within the certificate's validity period.
public abstract int	`getBasicConstraints()`[135]
	Gets the certificate constraints path length from the critical `BasicConstraints` extension, (OID = 2.5.29.19).
public abstract Principal	`getIssuerDN()`[136]
	Gets the `issuer` (issuer distinguished name) value from the certificate.
public abstract boolean	`getIssuerUniqueID()`[136]
	Gets the `issuerUniqueID` value from the certificate.
public abstract boolean	`getKeyUsage()`[137]
	Gets a boolean array representing bits of the `KeyUsage` extension, (OID = 2.5.29.15).
public abstract Date	`getNotAfter()`[137]
	Gets the `notAfter` date from the validity period of the certificate.

Member Summary

public abstract Date	getNotBefore()[137] Gets the `notBefore` date from the validity period of the certificate.
public abstract BigInteger	getSerialNumber()[138] Gets the `serialNumber` value from the certificate.
public abstract String	getSigAlgName()[138] Gets the signature algorithm name for the certificate signature algorithm.
public abstract String	getSigAlgOID()[138] Gets the signature algorithm OID string from the certificate.
public abstract byte	getSigAlgParams()[139] Gets the DER-encoded signature algorithm parameters from this certificate's signature algorithm.
public abstract byte	getSignature()[139] Gets the `signature` value (the raw signature bits) from the certificate.
public abstract Principal	getSubjectDN()[139] Gets the `subject` (subject distinguished name) value from the certificate.
public abstract boolean	getSubjectUniqueID()[139] Gets the `subjectUniqueID` value from the certificate.
public abstract byte	getTBSCertificate()[139] Gets the DER-encoded certificate information, the `tbsCertificate` from this certificate.
public abstract int	getVersion()[140] Gets the `version` (version number) value from the certificate.

Inherited Member Summary

Inner classes inherited from class Certificate[8]

Certificate.CertificateRep[12]

Methods inherited from class Certificate[8]

getType()[10], equals(Object)[9], hashCode()[10], getEncoded()[10], verify(PublicKey)[11], verify(PublicKey, String)[11], toString()[10], getPublicKey()[10], writeReplace()[11]

Methods inherited from class java.lang.Object

getClass, clone, notify, notifyAll, wait, wait, wait, finalize

Methods inherited from interface X509Extension[178]

hasUnsupportedCriticalExtension()[180], getCriticalExtensionOIDs()[179], getNonCriticalExtensionOIDs()[180], getExtensionValue(String)[179]

Constructors

X509Certificate()

```
protected X509Certificate()
```

Constructor for X.509 certificates.

Methods

checkValidity()

```
public abstract void checkValidity()
```

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`[18] - if the certificate has expired.

`CertificateNotYetValidException`[33] - if the certificate is not yet valid.

checkValidity(Date)

```
public abstract void checkValidity(java.util.Date date)
```

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`[18] - if the certificate has expired with respect to the `date` supplied.

`CertificateNotYetValidException`[33] - if the certificate is not yet valid with respect to the `date` supplied.

See Also: `checkValidity()`[135]

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.

---

getIssuerDN()

```
public abstract java.security.Principal getIssuerDN()
```

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.

---

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.

---

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.

---

getNotAfter()

```
public abstract java.util.Date getNotAfter()
```

Gets the `notAfter` date from the validity period of the certificate. See `getNotBefore()`$_{137}$ for relevant ASN.1 definitions.

Returns: the end date of the validity period.

See Also: `checkValidity()`$_{135}$

---

getNotBefore()

```
public abstract java.util.Date getNotBefore()
```

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

getSerialNumber()

```
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()`[135]

---

getSerialNumber()

```
public abstract java.math.BigInteger 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.

---

getSigAlgName()

```
public abstract java.lang.String 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 java.lang.String getSigAlgOID()
```

Gets the signature algorithm OID string from the certificate. An OID is represented by a set of positive 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()`[138] 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` and instantiate with the name returned by `getSigAlgName()`[138] .

See `getSigAlgName()`[138] for relevant ASN.1 definitions.

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

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.

getSubjectDN()

```
public abstract java.security.Principal getSubjectDN()
```

Gets the `subject` (subject distinguished name) value from the certificate. The ASN.1 definition for this is:

```
subject    Name
```

See `getIssuerDN()`[136] for `Name` and other relevant definitions.

Returns: a Principal whose name is the subject name.

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.

getTBSCertificate()

```
public abstract byte[] getTBSCertificate()
```

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`[14] - if an encoding error occurs.

---

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.

java.security.cert
X509CertSelector

Syntax

```
public class X509CertSelector implements CertSelector66

java.lang.Object
  |
  +--java.security.cert.X509CertSelector
```

All Implemented Interfaces: CertSelector66,java.lang.Cloneable

Description

A `CertSelector` that selects `X509Certificates` that match all specified criteria. This class is particularly useful when selecting certificates from a `CertStore` to build a PKIX-compliant certification path.

When first constructed, an `X509CertSelector` has no criteria enabled and each of the `get` methods return a default value (`null`, or `-1` for the `getBasicConstraints()`147 method). Therefore, the `match(Certificate)`152 method would return `true` for any `X509Certificate`. Typically, several criteria are enabled (by calling `setIssuer(String)`155 or `setKeyUsage(boolean[])`155 , for instance) and then the `X509CertSelector` is passed to `getCertificates(CertSelector)`70 or some similar method.

Several criteria can be enabled (by calling `setIssuer(String)`155 and `setSerialNumber(BigInteger)`157 , for example) such that the `match` method usually uniquely matches a single `X509Certificate`. We say usually, since it is possible for two issuing CAs to have the same distinguished name and each issue a certificate with the same serial number. Other unique combinations include the issuer, subject, subjectKeyIdentifier and/or the subjectPublicKey criteria.

Please refer to RFC 2459 for definitions of the X.509 certificate extensions mentioned below.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: CertSelector66, X509Certificate132

Member Summary

Constructors

public	`X509CertSelector()`145
	Creates an `X509CertSelector`.

Methods

Member Summary

public void	`addPathToName(int, byte[])`[145] Adds a name to the pathToNames criterion.
public void	`addPathToName(int, String)`[145] Adds a name to the pathToNames criterion.
public void	`addSubjectAlternativeName(int, byte[])`[146] Adds a name to the subjectAlternativeNames criterion.
public void	`addSubjectAlternativeName(int, String)`[146] Adds a name to the subjectAlternativeNames criterion.
public Object	`clone()`[147] Returns a copy of this object.
public boolean	`equals(Object)`[147] Compares this object for equality with the specified object.
public byte	`getAuthorityKeyIdentifier()`[147] Returns the authorityKeyIdentifier criterion.
public int	`getBasicConstraints()`[147] Returns the basic constraints constraint.
public X509Certificate	`getCertificate()`[148] Returns the certificateEquals criterion.
public Date	`getCertificateValid()`[148] Returns the certificateValid criterion.
public Set	`getExtendedKeyUsage()`[148] Returns the extendedKeyUsage criterion.
public byte	`getIssuerAsBytes()`[148] Returns the issuer criterion as a byte array.
public String	`getIssuerAsString()`[148] Returns the issuer criterion as a `String`.
public boolean	`getKeyUsage()`[149] Returns the keyUsage criterion.
public boolean	`getMatchAllSubjectAltNames()`[149] Indicates if the `X509Certificate` must contain all or at least one of the subjectAlternativeNames specified in the `setSubjectAlternativeNames(Collection)`[158] or `addSubjectAlternativeName(int, String)`[146] methods.
public byte	`getNameConstraints()`[149] Returns the name constraints criterion.
public Collection	`getPathToNames()`[149] Returns a copy of the pathToNames criterion.
public Set	`getPolicy()`[150] Returns the policy criterion.

Member Summary

public Date	getPrivateKeyValid() 150 Returns the privateKeyValid criterion.
public BigInteger	getSerialNumber() 150 Returns the serialNumber criterion.
public Collection	getSubjectAlternativeNames() 150 Returns a copy of the subjectAlternativeNames criterion.
public byte	getSubjectAsBytes() 151 Returns the subject criterion as a byte array.
public String	getSubjectAsString() 151 Returns the subject criterion as a `String`.
public byte	getSubjectKeyIdentifier() 151 Returns the subjectKeyIdentifier criterion.
public PublicKey	getSubjectPublicKey() 152 Returns the subjectPublicKey criterion.
public String	getSubjectPublicKeyAlgID() 152 Returns the subjectPublicKeyAlgID criterion.
public int	hashCode() 152 Returns a hash code value for this object.
public boolean	match(Certificate) 152 Decides whether a `Certificate` should be selected.
public void	setAuthorityKeyIdentifier(byte[]) 152 Sets the authorityKeyIdentifier criterion.
public void	setBasicConstraints(int) 153 Sets the basic constraints constraint.
public void	setCertificate(X509Certificate) 153 Sets the certificateEquals criterion.
public void	setCertificateValid(Date) 154 Sets the certificateValid criterion.
public void	setExtendedKeyUsage(Set) 154 Sets the extendedKeyUsage criterion.
public void	setIssuer(byte[]) 154 Sets the issuer criterion.
public void	setIssuer(String) 155 Sets the issuer criterion.
public void	setKeyUsage(boolean[]) 155 Sets the keyUsage criterion.

Member Summary

public void	setMatchAllSubjectAltNames(boolean)[155] Enables/disables matching all of the subjectAlternativeNames specified in the setSubjectAlternativeNames(Collection)[158] or addSubjectAlternativeName(int, String)[146] methods.
public void	setNameConstraints(byte[])[155] Sets the name constraints criterion.
public void	setPathToNames(Collection)[156] Sets the pathToNames criterion.
public void	setPolicy(Set)[157] Sets the policy constraint.
public void	setPrivateKeyValid(Date)[157] Sets the privateKeyValid criterion.
public void	setSerialNumber(BigInteger)[157] Sets the serialNumber criterion.
public void	setSubject(byte[])[157] Sets the subject criterion.
public void	setSubject(String)[158] Sets the subject criterion.
public void	setSubjectAlternativeNames(Collection)[158] Sets the subjectAlternativeNames criterion.
public void	setSubjectKeyIdentifier(byte[])[158] Sets the subjectKeyIdentifier criterion.
public void	setSubjectPublicKey(byte[])[159] Sets the subjectPublicKey criterion.
public void	setSubjectPublicKey(PublicKey)[159] Sets the subjectPublicKey criterion.
public void	setSubjectPublicKeyAlgID(String)[160] Sets the subjectPublicKeyAlgID criterion.
public String	toString()[160] Return a printable representation of the CertSelector.

Inherited Member Summary

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait, finalize

Constructors

X509CertSelector()

```
public X509CertSelector()
```

Creates an `X509CertSelector`. Initially, no criteria are set so any `X509Certificate` will match.

Methods

addPathToName(int, byte[])

```
public void addPathToName(int type, byte[] name)
```

Adds a name to the pathToNames criterion. The `X509Certificate` must not include name constraints that would prohibit building a path to the specified name.

This method allows the caller to add a name to the set of names which the `X509Certificates`'s name constraints must permit. The specified name is added to any previous value for the pathToNames criterion. If the name is a duplicate, it may be ignored.

The name is provided as a byte array. This byte array should contain the DER encoded name, as it would appear in the GeneralName structure defined in RFC 2459 and X.509. The ASN.1 definition of this structure appears in the documentation for `addSubjectAlternativeName(int, byte[])`[146] .

Note that the byte array supplied here is cloned to protect against subsequent modifications.

Parameters:

`type` - the name type (0-8, as specified in RFC 2459, section 4.2.1.7)

`name` - a byte array containing the name in ASN.1 DER encoded form

Throws:

`IOException` - if a parsing error occurs

addPathToName(int, String)

```
public void addPathToName(int type, java.lang.String name)
```

Adds a name to the pathToNames criterion. The `X509Certificate` must not include name constraints that would prohibit building a path to the specified name.

This method allows the caller to add a name to the set of names which the `X509Certificates`'s name constraints must permit. The specified name is added to any previous value for the pathToNames criterion. If the name is a duplicate, it may be ignored.

The name is provided in string format. RFC 822, DNS, and URI names use the well-established string formats for those types (subject to the restrictions included in RFC 2459). IPv4 address names are supplied using dotted quad notation. OID address names are represented as a series of nonnegative integers separated by periods. And directory names (distinguished names) are supplied in RFC 1779 format. No standard string format is defined for otherNames, X.400 names, EDI party names, IPv6 address names, or any other type of names. They should be specified using the `addPathToName(int, byte[])`[145] method.

addSubjectAlternativeName(int, byte[])

Parameters:

`type` - the name type (0-8, as specified in RFC 2459, section 4.2.1.7)

`name` - the name in string form

Throws:

`IOException` - if a parsing error occurs

---

addSubjectAlternativeName(int, byte[])

```
public void addSubjectAlternativeName(int type, byte[] name)
```

Adds a name to the subjectAlternativeNames criterion. The `X509Certificate` must contain all or at least one of the specified subjectAlternativeNames, depending on the value of the matchAllNames flag (see `setMatchAllSubjectAltNames(boolean)` 155).

This method allows the caller to add a name to the set of subject alternative names. The specified name is added to any previous value for the subjectAlternativeNames criterion. If the specified name is a duplicate, it may be ignored.

The name is provided as a byte array. This byte array should contain the DER encoded name, as it would appear in the GeneralName structure defined in RFC 2459 and X.509. The encoded byte array should only contain the encoded value of the name, and should not include the tag associated with the name in the GeneralName structure. The ASN.1 definition of this structure appears below.

```
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}
```

Note that the byte array supplied here is cloned to protect against subsequent modifications.

Parameters:

`type` - the name type (0-8, as listed above)

`name` - a byte array containing the name in ASN.1 DER encoded form

Throws:

`IOException` - if a parsing error occurs

---

addSubjectAlternativeName(int, String)

```
public void addSubjectAlternativeName(int type, java.lang.String name)
```

Adds a name to the subjectAlternativeNames criterion. The `X509Certificate` must contain all or at least one of the specified subjectAlternativeNames, depending on the value of the matchAllNames flag (see `setMatchAllSubjectAltNames(boolean)` 155).

This method allows the caller to add a name to the set of subject alternative names. The specified name is added to any previous value for the subjectAlternativeNames criterion. If the specified name is a duplicate, it may be ignored.

The name is provided in string format. RFC 822, DNS, and URI names use the well-established string formats for those types (subject to the restrictions included in RFC 2459). IPv4 address names are supplied

using dotted quad notation. OID address names are represented as a series of nonnegative integers separated by periods. And directory names (distinguished names) are supplied in RFC 1779 format. No standard string format is defined for otherNames, X.400 names, EDI party names, IPv6 address names, or any other type of names. They should be specified using the `addSubjectAlternativeName(int, byte[])`[146] method.

Parameters:

`type` - the name type (0-8, as specified in RFC 2459, section 4.2.1.7)

`name` - the name in string form (not `null`)

Throws:

`IOException` - if a parsing error occurs

---

clone()

```
public java.lang.Object clone()
```

Returns a copy of this object.

Specified By: `clone()`[147] in interface `X509CertSelector`[141]

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: the copy

---

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this object for equality with the specified object. If the other object is not an `X509CertSelector`, return `false`. Otherwise, return `true` if the parameters of the objects are equal.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:

`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

---

getAuthorityKeyIdentifier()

```
public byte[] getAuthorityKeyIdentifier()
```

Returns the authorityKeyIdentifier criterion. The `X509Certificate` must contain a AuthorityKeyIdentifier extension with the specified value. If `null`, no authorityKeyIdentifier check will be done.

Note that the byte array returned is cloned to protect against subsequent modifications.

Returns: the key identifier (or `null`)

---

getBasicConstraints()

```
public int getBasicConstraints()
```

Returns the basic constraints constraint. If the value is greater than or equal to zero, the `X509Certificates` must include a basicConstraints extension with a pathLen of at least this value. If

the value is -2, only end-entity certificates are accepted. If the value is -1, no basicConstraints check is done.

Returns: the value for the basic constraints constraint

---

getCertificate()

`public X509Certificate`$_{132}$ **`getCertificate`**`()`

Returns the certificateEquals criterion. The specified `X509Certificate` must be equal to the `X509Certificate` passed to the `match` method. If `null`, this check is not applied.

Returns: the `X509Certificate` to match (or `null`)

---

getCertificateValid()

`public java.util.Date` **`getCertificateValid`**`()`

Returns the certificateValid criterion. The specified date must fall within the certificate validity period for the `X509Certificate`. If `null`, no certificateValid check will be done.

Note that the `Date` returned is cloned to protect against subsequent modifications.

Returns: the `Date` to check (or `null`)

---

getExtendedKeyUsage()

`public java.util.Set` **`getExtendedKeyUsage`**`()`

Returns the extendedKeyUsage criterion. The `X509Certificate` must allow the specified key purposes in its extended key usage extension. If the `keyPurposeSet` returned is empty or `null`, no extendedKeyUsage check will be done. Note that an `X509Certificate` that has no extendedKeyUsage extension implicitly allows all key purposes.

Returns: an immutable `Set` of key purpose OIDs in string format (or `null`)

---

getIssuerAsBytes()

`public byte[]` **`getIssuerAsBytes`**`()`

Returns the issuer criterion as a byte array. This distinguished name must match the issuer distinguished name in the `X509Certificate`. If `null`, the issuer criterion is disabled and any issuer distinguished name will do.

If the value returned is not `null`, it is a byte array containing a single DER encoded distinguished name, as defined in X.501. The ASN.1 notation for this structure is supplied in the documentation for `setIssuer(byte[])`$_{154}$.

Note that the byte array returned is cloned to protect against subsequent modifications.

Returns: a byte array containing the required issuer distinguished name in ASN.1 DER format (or `null`)

Throws:

`IOException` - if an encoding error occurs

---

getIssuerAsString()

```
public java.lang.String getIssuerAsString()
```

Returns the issuer criterion as a `String`. This distinguished name must match the issuer distinguished name in the `X509Certificate`. If `null`, the issuer criterion is disabled and any issuer distinguished name will do.

If the value returned is not `null`, it is a distinguished name, in RFC 1779 format.

Returns: the required issuer distinguished name in RFC 1779 format (or `null`)

---

getKeyUsage()

```
public boolean[] getKeyUsage()
```

Returns the keyUsage criterion. The `X509Certificate` must allow the specified keyUsage values. If null, no keyUsage check will be done.

Note that the boolean array returned is cloned to protect against subsequent modifications.

Returns: a boolean array in the same format as the boolean array returned by `getKeyUsage()`$_{137}$. Or `null`.

---

getMatchAllSubjectAltNames()

```
public boolean getMatchAllSubjectAltNames()
```

Indicates if the `X509Certificate` must contain all or at least one of the subjectAlternativeNames specified in the `setSubjectAlternativeNames(Collection)`$_{158}$ or `addSubjectAlternativeName(int, String)`$_{146}$ methods. If `true`, the `X509Certificate` must contain all of the specified subject alternative names. If `false`, the `X509Certificate` must contain at least one of the specified subject alternative names.

Returns: `true` if the flag is enabled; `false` if the flag is disabled. The flag is `true` by default.

---

getNameConstraints()

```
public byte[] getNameConstraints()
```

Returns the name constraints criterion. The `X509Certificate` must have subject and subject alternative names that meet the specified name constraints.

The name constraints are returned as a byte array. This byte array contains the DER encoded form of the name constraints, as they would appear in the NameConstraints structure defined in RFC 2459 and X.509. The ASN.1 notation for this structure is supplied in the documentation for `setNameConstraints(byte[])`$_{155}$.

Note that the byte array returned is cloned to protect against subsequent modifications.

Returns: a byte array containing the ASN.1 DER encoding of a NameConstraints extension used for checking name constraints. `null` if no name constraints check will be performed.

---

getPathToNames()

```
public java.util.Collection getPathToNames()
```

Returns a copy of the pathToNames criterion. The `X509Certificate` must not include name constraints that would prohibit building a path to the specified names. If the value returned is `null`, no pathToNames check will be performed.

If the value returned is not `null`, it is a `Collection` with one entry for each name to be included in the pathToNames criterion. 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). There can be multiple names of the same type. Note that the `Collection` returned may contain duplicate names (same name and name type).

Each name in the `Collection` may be specified either as a `String` or as an ASN.1 encoded byte array. For more details about the formats used, see `addPathToName(int, String)`[145] and `addPathToName(int, byte[])`[145].

Note that a deep copy is performed on the `Collection` to protect against subsequent modifications.

Returns: a `Collection` of names (or `null`)

---

getPolicy()

```
public java.util.Set getPolicy()
```

Returns the policy criterion. The `X509Certificate` must include at least one of the specified policies in its certificate policies extension. If the `Set` returned is empty, then the `X509Certificate` must include at least some specified policy in its certificate policies extension. If the `Set` returned is `null`, no policy check will be performed.

Returns: an immutable `Set` of certificate policy OIDs in string format (or `null`)

---

getPrivateKeyValid()

```
public java.util.Date getPrivateKeyValid()
```

Returns the privateKeyValid criterion. The specified date must fall within the private key validity period for the `X509Certificate`. If `null`, no privateKeyValid check will be done.

Note that the `Date` returned is cloned to protect against subsequent modifications.

Returns: the `Date` to check (or `null`)

---

getSerialNumber()

```
public java.math.BigInteger getSerialNumber()
```

Returns the serialNumber criterion. The specified serial number must match the certificate serial number in the `X509Certificate`. If `null`, any certificate serial number will do.

Returns: the certificate serial number to match (or `null`)

---

getSubjectAlternativeNames()

```
public java.util.Collection getSubjectAlternativeNames()
```

Returns a copy of the subjectAlternativeNames criterion. The `X509Certificate` must contain all or at least one of the specified subjectAlternativeNames, depending on the value of the matchAllNames flag (see

`getMatchAllSubjectAltNames()`$_{149}$). If the value returned is `null`, no subjectAlternativeNames check will be performed.

If the value returned is not `null`, it is a `Collection` with one entry for each name to be included in the subject alternative name criterion. 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). There can be multiple names of the same type. Note that the `Collection` returned may contain duplicate names (same name and name type).

Each subject alternative name in the `Collection` may be specified either as a `String` or as an ASN.1 encoded byte array. For more details about the formats used, see `addSubjectAlternative-Name(int, String)`$_{146}$ and `addSubjectAlternativeName(int, byte[])`$_{146}$.

Note that a deep copy is performed on the `Collection` to protect against subsequent modifications.

Returns: a `Collection` of names (or `null`)

---

getSubjectAsBytes()

```
public byte[] getSubjectAsBytes()
```

Returns the subject criterion as a byte array. This distinguished name must match the subject distinguished name in the `X509Certificate`. If `null`, the subject criterion is disabled and any subject distinguished name will do.

If the value returned is not `null`, it is a byte array containing a single DER encoded distinguished name, as defined in X.501. The ASN.1 notation for this structure is supplied in the documentation for `setSubject(byte[])`$_{157}$.

Note that the byte array returned is cloned to protect against subsequent modifications.

Returns: a byte array containing the required subject distinguished name in ASN.1 DER format (or `null`)

Throws:

`IOException` - if an encoding error occurs

---

getSubjectAsString()

```
public java.lang.String getSubjectAsString()
```

Returns the subject criterion as a `String`. This distinguished name must match the subject distinguished name in the `X509Certificate`. If `null`, the subject criterion is disabled and any subject distinguished name will do.

If the value returned is not `null`, it is a distinguished name, in RFC 1779 format.

Returns: the required subject distinguished name in RFC 1779 format (or `null`)

---

getSubjectKeyIdentifier()

```
public byte[] getSubjectKeyIdentifier()
```

Returns the subjectKeyIdentifier criterion. The `X509Certificate` must contain a SubjectKeyIdentifier extension with the specified value. If `null`, no subjectKeyIdentifier check will be done.

Note that the byte array returned is cloned to protect against subsequent modifications.

Returns: the key identifier (or `null`)

getSubjectPublicKey()

```
public java.security.PublicKey getSubjectPublicKey()
```

Returns the subjectPublicKey criterion. The `X509Certificate` must contain the specified subject public key. If `null`, no subjectPublicKey check will be done.

Returns: the subject public key to check for (or `null`)

getSubjectPublicKeyAlgID()

```
public java.lang.String getSubjectPublicKeyAlgID()
```

Returns the subjectPublicKeyAlgID criterion. The `X509Certificate` must contain a subject public key with the specified algorithm. If `null`, no subjectPublicKeyAlgID check will be done.

Returns: the object identifier (OID) of the signature algorithm to check for (or `null`). An OID is represented by a set of nonnegative integers separated by periods.

hashCode()

```
public int hashCode()
```

Returns a hash code value for this object.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: a hash code value

match(Certificate)

```
public boolean match(Certificate[8] cert)
```

Decides whether a `Certificate` should be selected.

Specified By: `match(Certificate)`[152] in interface `X509CertSelector`[141]

Parameters:
`cert` - the `Certificate` to be checked

Returns: `true` if the `Certificate` should be selected, `false` otherwise

setAuthorityKeyIdentifier(byte[])

```
public void setAuthorityKeyIdentifier(byte[] authorityKeyID)
```

Sets the authorityKeyIdentifier criterion. The `X509Certificate` must contain an AuthorityKeyIdentifier extension for which the contents of the extension value matches the specified criterion value. If the criterion value is `null`, no authorityKeyIdentifier check will be done.

If `authorityKeyID` is not `null`, it should contain a single DER encoded value corresponding to the contents of the extension value (not including the object identifier, criticality setting, and encapsulating OCTET STRING) for an AuthorityKeyIdentifier extension. The ASN.1 notation for this structure follows.

```
AuthorityKeyIdentifier ::= SEQUENCE {
   keyIdentifier             [0] KeyIdentifier           OPTIONAL,
   authorityCertIssuer       [1] GeneralNames            OPTIONAL,
   authorityCertSerialNumber [2] CertificateSerialNumber OPTIONAL  }
KeyIdentifier ::= OCTET STRING
```

Authority key identifiers are not parsed by the `X509CertSelector`. Instead, the values are compared using a byte-by-byte comparison.

When the `keyIdentifier` field of `AuthorityKeyIdentifier` is populated, the value is usually taken from the `SubjectKeyIdentifier` extension in the issuer's certificate. Note, however, that the result of `X509Certificate.getExtensionValue(<SubjectKeyIdentifier Object Identifier>)` on the issuer's certificate may NOT be used directly as the input to `setAuthorityKeyIdentifier`. This is because the SubjectKeyIdentifier contains only a KeyIdentifier OCTET STRING, and not a SEQUENCE of KeyIdentifier, GeneralNames, and CertificateSerialNumber. In order to use the extension value of the issuer certificate's `SubjectKeyIdentifier` extension, it will be necessary to extract the value of the embedded `KeyIdentifier` OCTET STRING, then DER encode this OCTET STRING inside a SEQUENCE. For more details on SubjectKeyIdentifier, see `setSubjectKeyIdentifier(byte[])`[158] .

Note also that the byte array supplied here is cloned to protect against subsequent modifications.

Parameters:

`authorityKeyID` - the authority key identifier (or `null`)

---

setBasicConstraints(int)

`public void` **`setBasicConstraints`**`(int minMaxPathLen)`

Sets the basic constraints constraint. If the value is greater than or equal to zero, `X509Certificates` must include a basicConstraints extension with a pathLen of at least this value. If the value is -2, only end-entity certificates are accepted. If the value is -1, no check is done.

This constraint is useful when building a certification path forward (from the target toward the trust anchor. If a partial path has been built, any candidate certificate must have a maxPathLen value greater than or equal to the number of certificates in the partial path.

Parameters:

`minMaxPathLen` - the value for the basic constraints constraint

Throws:

`IllegalArgumentException` - if the value is less than -2

---

setCertificate(X509Certificate)

`public void` **`setCertificate`**`(``X509Certificate`[132] `cert)`

Sets the certificateEquals criterion. The specified `X509Certificate` must be equal to the `X509Certificate` passed to the `match` method. If `null`, then this check is not applied.

This method is particularly useful when it is necessary to match a single certificate. Although other criteria can be specified in conjunction with the certificateEquals criterion, it is usually not practical or necessary.

Parameters:

`cert` - the `X509Certificate` to match (or `null`)

setCertificateValid(Date)

```
public void setCertificateValid(java.util.Date certValid)
```

Sets the certificateValid criterion. The specified date must fall within the certificate validity period for the `X509Certificate`. If `null`, no certificateValid check will be done.

Note that the `Date` supplied here is cloned to protect against subsequent modifications.

Parameters:

`certValid` - the `Date` to check (or `null`)

setExtendedKeyUsage(Set)

```
public void setExtendedKeyUsage(java.util.Set keyPurposeSet)
```

Sets the extendedKeyUsage criterion. The `X509Certificate` must allow the specified key purposes in its extended key usage extension. If `keyPurposeSet` is empty or `null`, no extendedKeyUsage check will be done. Note that an `X509Certificate` that has no extendedKeyUsage extension implicitly allows all key purposes.

Note that the `Set` is cloned to protect against subsequent modifications.

Parameters:

`keyPurposeSet` - a `Set` of key purpose OIDs in string format (or `null`)

setIssuer(byte[])

```
public void setIssuer(byte[] issuerDN)
```

Sets the issuer criterion. The specified distinguished name must match the issuer distinguished name in the `X509Certificate`. If `null` is specified, the issuer criterion is disabled and any issuer distinguished name will do.

If `issuerDN` is not `null`, it should contain a single DER encoded distinguished name, as defined in X.501. The ASN.1 notation for this structure is as follows.

```
Name ::= CHOICE {
  RDNSequence }
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::=
  SET SIZE (1 .. MAX) OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
  type     AttributeType,
  value    AttributeValue }
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY AttributeType
....
DirectoryString ::= CHOICE {
      teletexString           TeletexString (SIZE (1..MAX)),
      printableString         PrintableString (SIZE (1..MAX)),
      universalString         UniversalString (SIZE (1..MAX)),
      utf8String              UTF8String (SIZE (1.. MAX)),
      bmpString               BMPString (SIZE (1..MAX)) }
```

Note that the byte array specified here is cloned to protect against subsequent modifications.

Parameters:

`issuerDN` - a byte array containing the distinguished name in ASN.1 DER encoded form (or `null`)

Throws:

`IOException` - if an encoding error occurs (incorrect form for DN)

---

setIssuer(String)

```
public void setIssuer(java.lang.String issuerDN)
```

Sets the issuer criterion. The specified distinguished name must match the issuer distinguished name in the `X509Certificate`. If `null`, any issuer distinguished name will do.

If `issuerDN` is not `null`, it should contain a distinguished name, in RFC 1779 format.

Parameters:

`issuerDN` - a distinguished name in RFC 1779 format (or `null`)

Throws:

`IOException` - if a parsing error occurs (incorrect form for DN)

---

setKeyUsage(boolean[])

```
public void setKeyUsage(boolean[] keyUsage)
```

Sets the keyUsage criterion. The `X509Certificate` must allow the specified keyUsage values. If `null`, no keyUsage check will be done. Note that an `X509Certificate` that has no keyUsage extension implicitly allows all keyUsage values.

Note that the boolean array supplied here is cloned to protect against subsequent modifications.

Parameters:

`keyUsage` - a boolean array in the same format as the boolean array returned by `getKeyUsage()`[137] . Or `null`.

---

setMatchAllSubjectAltNames(boolean)

```
public void setMatchAllSubjectAltNames(boolean matchAllNames)
```

Enables/disables matching all of the subjectAlternativeNames specified in the `setSubjectAlternativeNames(Collection)`[158] or `addSubjectAlternativeName(int, String)`[146] methods. If enabled, the `X509Certificate` must contain all of the specified subject alternative names. If disabled, the `X509Certificate` must contain at least one of the specified subject alternative names.

The matchAllNames flag is `true` by default.

Parameters:

`matchAll` - if `true`, the flag is enabled; if `false`, the flag is disabled.

---

setNameConstraints(byte[])

```
public void setNameConstraints(byte[] bytes)
```

Sets the name constraints criterion. The `X509Certificate` must have subject and subject alternative names that meet the specified name constraints.

The name constraints are specified as a byte array. This byte array should contain the DER encoded form of the name constraints, as they would appear in the NameConstraints structure defined in RFC 2459 and X.509. The ASN.1 definition of this structure appears below.

```
NameConstraints ::= SEQUENCE {
     permittedSubtrees       [0]     GeneralSubtrees OPTIONAL,
     excludedSubtrees        [1]     GeneralSubtrees OPTIONAL }
GeneralSubtrees ::= SEQUENCE SIZE (1..MAX) OF GeneralSubtree
GeneralSubtree ::= SEQUENCE {
     base                    GeneralName,
     minimum         [0]     BaseDistance DEFAULT 0,
     maximum         [1]     BaseDistance OPTIONAL }
BaseDistance ::= INTEGER (0..MAX)
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}
```

Note that the byte array supplied here is cloned to protect against subsequent modifications.

Parameters:

`bytes` - a byte array containing the ASN.1 DER encoding of a NameConstraints extension to be used for checking name constraints. Only the value of the extension is included, not the OID or criticality flag. Can be `null`, in which case no name constraints check will be performed.

Throws:

`IOException` - if a parsing error occurs

---

setPathToNames(Collection)

```
public void setPathToNames(java.util.Collection names)
```

Sets the pathToNames criterion. The `X509Certificate` must not include name constraints that would prohibit building a path to the specified names.

This method allows the caller to specify, with a single method call, the complete set of names which the `X509Certificates`'s name constraints must permit. The specified value replaces the previous value for the pathToNames criterion.

This constraint is useful when building a certification path forward (from the target toward the trust anchor. If a partial path has been built, any candidate certificate must not include name constraints that would prohibit building a path to any of the names in the partial path.

The `names` parameter (if not `null`) is a `Collection` with one entry for each name to be included in the pathToNames criterion. 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). There can be multiple names of the same type. If `null` is supplied as the value for this argument, no pathToNames check will be performed.

Each name in the `Collection` may be specified either as a `String` or as an ASN.1 encoded byte array. For more details about the formats used, see `addPathToName(int, String)`$_{145}$ and `addPathToName(int, byte[])`$_{145}$.

Note that the `names` parameter can contain duplicate names (same name and name type), but they may be removed from the `Collection` of names returned by the `getPathToNames()`$_{149}$ method.

Note that a deep copy is performed on the `Collection` to protect against subsequent modifications.

Parameters:

`names` - a `Collection` with one entry per name (or `null`)

Throws:

`IOException` - if a parsing error occurs

---

setPolicy(Set)

```
public void setPolicy(java.util.Set certPolicySet)
```

Sets the policy constraint. The `X509Certificate` must include at least one of the specified policies in its certificate policies extension. If `certPolicySet` is empty, then the `X509Certificate` must include at least some specified policy in its certificate policies extension. If `certPolicySet` is `null`, no policy check will be performed.

Note that the `Set` is cloned to protect against subsequent modifications.

Parameters:

`certPolicySet` - a `Set` of certificate policy OIDs in string format (or `null`)

Throws:

`IOException` - if a parsing error occurs

---

setPrivateKeyValid(Date)

```
public void setPrivateKeyValid(java.util.Date privateKeyValid)
```

Sets the privateKeyValid criterion. The specified date must fall within the private key validity period for the `X509Certificate`. If `null`, no privateKeyValid check will be done.

Note that the `Date` supplied here is cloned to protect against subsequent modifications.

Parameters:

`privateKeyValid` - the `Date` to check (or `null`)

---

setSerialNumber(BigInteger)

```
public void setSerialNumber(java.math.BigInteger serial)
```

Sets the serialNumber criterion. The specified serial number must match the certificate serial number in the `X509Certificate`. If `null`, any certificate serial number will do.

Parameters:

`serial` - the certificate serial number to match (or `null`)

---

setSubject(byte[])

```
public void setSubject(byte[] subjectDN)
```

Sets the subject criterion. The specified distinguished name must match the subject distinguished name in the `X509Certificate`. If `null`, any subject distinguished name will do.

If `subjectDN` is not `null`, it should contain a single DER encoded distinguished name, as defined in X.501. For the ASN.1 notation for this structure, see `setIssuer(byte[])`$_{154}$.

Parameters:

`subjectDN` - a byte array containing the distinguished name in ASN.1 DER format (or `null`)

Throws:
`IOException` - if an encoding error occurs (incorrect form for DN)

---

setSubject(String)

```
public void setSubject(java.lang.String subjectDN)
```

Sets the subject criterion. The specified distinguished name must match the subject distinguished name in the `X509Certificate`. If `null`, any subject distinguished name will do.

If `subjectDN` is not `null`, it should contain a distinguished name, in RFC 1779 format.

Parameters:
`subjectDN` - a distinguished name in RFC 1779 format (or `null`)

Throws:
`IOException` - if a parsing error occurs (incorrect form for DN)

---

setSubjectAlternativeNames(Collection)

```
public void setSubjectAlternativeNames(java.util.Collection names)
```

Sets the subjectAlternativeNames criterion. The `X509Certificate` must contain all or at least one of the specified subjectAlternativeNames, depending on the value of the matchAllNames flag (see `setMatchAllSubjectAltNames(boolean)`[155]).

This method allows the caller to specify, with a single method call, the complete set of subject alternative names for the subjectAlternativeNames criterion. The specified value replaces the previous value for the subjectAlternativeNames criterion.

The `names` parameter (if not `null`) is a `Collection` with one entry for each name to be included in the subject alternative name criterion. 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). There can be multiple names of the same type. If `null` is supplied as the value for this argument, no subjectAlternativeNames check will be performed.

Each subject alternative name in the `Collection` may be specified either as a `String` or as an ASN.1 encoded byte array. For more details about the formats used, see `addSubjectAlternativeName(int, String)`[146] and `addSubjectAlternativeName(int, byte[])`[146] .

Note that the `names` parameter can contain duplicate names (same name and name type), but they may be removed from the `Collection` of names returned by the `getSubjectAlternativeNames()`[150] method.

Note that a deep copy is performed on the `Collection` to protect against subsequent modifications.

Parameters:
`names` - a `Collection` of names (or `null`)

Throws:
`IOException` - if a parsing error occurs

---

setSubjectKeyIdentifier(byte[])

```
public void setSubjectKeyIdentifier(byte[] subjectKeyID)
```

Sets the subjectKeyIdentifier criterion. The `X509Certificate` must contain a SubjectKeyIdentifier extension for which the contents of the extension matches the specified criterion value. If the criterion value is `null`, no subjectKeyIdentifier check will be done.

If `subjectKeyID` is not `null`, it should contain a single DER encoded value corresponding to the contents of the extension value (not including the object identifier, criticality setting, and encapsulating OCTET STRING) for a SubjectKeyIdentifier extension. The ASN.1 notation for this structure follows.

```
SubjectKeyIdentifier ::= KeyIdentifier
KeyIdentifier ::= OCTET STRING
```

Since the format of subject key identifiers is not mandated by any standard, subject key identifiers are not parsed by the `X509CertSelector`. Instead, the values are compared using a byte-by-byte comparison.

Note that the byte array supplied here is cloned to protect against subsequent modifications.

Parameters:

`subjectKeyID` - the subject key identifier (or `null`)

---

setSubjectPublicKey(byte[])

```
public void setSubjectPublicKey(byte[] key)
```

Sets the subjectPublicKey criterion. The `X509Certificate` must contain the specified subject public key. If `null`, no subjectPublicKey check will be done.

Because this method allows the public key to be specified as a byte array, it may be used for unknown key types.

If `key` is not `null`, it should contain a single DER encoded SubjectPublicKeyInfo structure, as defined in X.509. The ASN.1 notation for this structure is as follows.

```
SubjectPublicKeyInfo  ::=  SEQUENCE  {
  algorithm            AlgorithmIdentifier,
  subjectPublicKey     BIT STRING  }
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
```

Note that the byte array supplied here is cloned to protect against subsequent modifications.

Parameters:

`key` - a byte array containing the subject public key in ASN.1 DER form (or `null`)

Throws:

`IOException` - if an encoding error occurs (incorrect form for DN)

---

setSubjectPublicKey(PublicKey)

```
public void setSubjectPublicKey(java.security.PublicKey key)
```

Sets the subjectPublicKey criterion. The `X509Certificate` must contain the specified subject public key. If `null`, no subjectPublicKey check will be done.

Parameters:

`key` - the subject public key to check for (or `null`)

setSubjectPublicKeyAlgID(String)

```
public void setSubjectPublicKeyAlgID(java.lang.String oid)
```

Sets the subjectPublicKeyAlgID criterion. The `X509Certificate` must contain a subject public key with the specified algorithm. If `null`, no subjectPublicKeyAlgID check will be done.

Parameters:

`oid` - The object identifier (OID) of the algorithm to check for (or `null`). An OID is represented by a set of nonnegative integers separated by periods.

toString()

```
public java.lang.String toString()
```

Return a printable representation of the `CertSelector`.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a `String` describing the contents of the `CertSelector`

java.security.cert
X509CRL

Syntax

```
public abstract class X509CRL extends CRL85 implements X509Extension178

java.lang.Object
  |
  +--CRL85
        |
        +--java.security.cert.X509CRL
```

All Implemented Interfaces: X509Extension178

Description

Abstract class for an X.509 Certificate Revocation List (CRL). A CRL is a time-stamped list identifying revoked certificates. It is signed by a Certificate Authority (CA) and made freely available in a public repository.

Each revoked certificate is identified in a CRL by its certificate serial number. When a certificate-using system uses a certificate (e.g., for verifying a remote user's digital signature), that system not only checks the certificate signature and validity but also acquires a suitably- recent CRL and checks that the certificate serial number is not on that CRL. The meaning of "suitably-recent" may vary with local policy, but it usually means the most recently-issued CRL. A CA issues a new CRL on a regular periodic basis (e.g., hourly, daily, or weekly). Entries are added to CRLs as revocations occur, and an entry may be removed when the certificate expiration date is reached.

The X.509 v2 CRL format is described below in ASN.1:

```
CertificateList  ::=  SEQUENCE  {
    tbsCertList          TBSCertList,
    signatureAlgorithm   AlgorithmIdentifier,
    signature            BIT STRING  }
```

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 `tbsCertList` is:

```
TBSCertList  ::=  SEQUENCE  {
    version                 Version OPTIONAL,
                            -- if present, must be v2
    signature               AlgorithmIdentifier,
    issuer                  Name,
    thisUpdate              ChoiceOfTime,
    nextUpdate              ChoiceOfTime OPTIONAL,
    revokedCertificates     SEQUENCE OF SEQUENCE  {
        userCertificate         CertificateSerialNumber,
        revocationDate          ChoiceOfTime,
        crlEntryExtensions      Extensions OPTIONAL
                                -- if present, must be v2
        }  OPTIONAL,
    crlExtensions           [0]  EXPLICIT Extensions OPTIONAL
                                 -- if present, must be v2
    }
```

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

toString()

```
InputStream inStream = new FileInputStream("fileName-of-crl");
CertificateFactory cf = CertificateFactory.getInstance("X.509");
X509CRL crl = (X509CRL)cf.generateCRL(inStream);
inStream.close();
```

See Also: `CRL`[85], `CertificateFactory`[20], `X509Extension`[178]

Member Summary

Constructors

protected	`X509CRL()`[163]
	Constructor for X.509 CRLs.

Methods

public boolean	`equals(Object)`[163]
	Compares this CRL for equality with the given object.
public abstract byte	`getEncoded()`[164]
	Returns the ASN.1 DER-encoded form of this CRL.
public abstract Principal	`getIssuerDN()`[164]
	Gets the `issuer` (issuer distinguished name) value from the CRL.
public abstract Date	`getNextUpdate()`[164]
	Gets the `nextUpdate` date from the CRL.
public abstract X509CRLEntry	`getRevokedCertificate(BigInteger)`[164]
	Gets the CRL entry, if any, with the given certificate serialNumber.
public abstract Set	`getRevokedCertificates()`[165]
	Gets all the entries from this CRL.
public abstract String	`getSigAlgName()`[165]
	Gets the signature algorithm name for the CRL signature algorithm.
public abstract String	`getSigAlgOID()`[165]
	Gets the signature algorithm OID string from the CRL.
public abstract byte	`getSigAlgParams()`[165]
	Gets the DER-encoded signature algorithm parameters from this CRL's signature algorithm.
public abstract byte	`getSignature()`[165]
	Gets the `signature` value (the raw signature bits) from the CRL.
public abstract byte	`getTBSCertList()`[166]
	Gets the DER-encoded CRL information, the `tbsCertList` from this CRL.
public abstract Date	`getThisUpdate()`[166]
	Gets the `thisUpdate` date from the CRL.
public abstract int	`getVersion()`[166]
	Gets the `version` (version number) value from the CRL.

Member Summary	
public int	hashCode()[166] Returns a hashcode value for this CRL from its encoded form.
public abstract void	verify(PublicKey)[167] Verifies that this CRL was signed using the private key that corresponds to the given public key.
public abstract void	verify(PublicKey, String)[167] Verifies that this CRL was signed using the private key that corresponds to the given public key.

Inherited Member Summary
Methods inherited from class CRL[85] getType()[86], toString()[86], isRevoked(Certificate)[86]
Methods inherited from class java.lang.Object getClass, clone, notify, notifyAll, wait, wait, wait, finalize
Methods inherited from interface X509Extension[178] hasUnsupportedCriticalExtension()[180], getCriticalExtensionOIDs()[179], getNonCriticalExtensionOIDs()[180], getExtensionValue(String)[179]

Constructors

X509CRL()

```
protected X509CRL()
```

Constructor for X.509 CRLs.

Methods

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this CRL for equality with the given object. If the `other` object is an `instanceof` `X509CRL`, then its encoded form is retrieved and compared with the encoded form of this CRL.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:

`other` - the object to test for equality with this CRL.

Returns: true iff the encoded forms of the two CRLs match, false otherwise.

---

getEncoded()

```
public abstract byte[] getEncoded()
```

Returns the ASN.1 DER-encoded form of this CRL.

Returns: the encoded form of this certificate

Throws:
`CRLException`[87] - if an encoding error occurs.

---

getIssuerDN()

```
public abstract java.security.Principal getIssuerDN()
```

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

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.

---

getNextUpdate()

```
public abstract java.util.Date getNextUpdate()
```

Gets the `nextUpdate` date from the CRL.

Returns: the `nextUpdate` date from the CRL, or null if not present.

---

getRevokedCertificate(BigInteger)

```
public abstract X509CRLEntry[168] getRevokedCertificate(java.math.BigInteger serialNumber)
```

Gets the CRL entry, if any, with the given certificate serialNumber.

Parameters:
`serialNumber` - the serial number of the certificate for which a CRL entry is to be looked up

Returns: the entry with the given serial number, or null if no such entry exists in this CRL.

See Also: X509CRLEntry[168]

---

getRevokedCertificates()

```
public abstract java.util.Set getRevokedCertificates()
```

Gets all the entries from this CRL. This returns a Set of X509CRLEntry objects.

Returns: all the entries or null if there are none present.

See Also: X509CRLEntry[168]

---

getSigAlgName()

```
public abstract java.lang.String getSigAlgName()
```

Gets the signature algorithm name for the CRL 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 java.lang.String getSigAlgOID()
```

Gets the signature algorithm OID string from the CRL. An OID is represented by a set of positive 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()` 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 CRL's signature algorithm. In most cases, the signature algorithm parameters are null; the parameters are usually supplied with the public key. If access to individual parameter values is needed then use `AlgorithmParameters` and instantiate with the name returned by `getSigAlgName()`.

See `getSigAlgName()` for relevant ASN.1 definitions.

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

---

getSignature()

getTBSCertList()

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

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

```
signature     BIT STRING
```

Returns: the signature.

---

getTBSCertList()

```
public abstract byte[] getTBSCertList()
```

Gets the DER-encoded CRL information, the `tbsCertList` from this CRL. This can be used to verify the signature independently.

Returns: the DER-encoded CRL information.

Throws:

`CRLException`[87] - if an encoding error occurs.

---

getThisUpdate()

```
public abstract java.util.Date getThisUpdate()
```

Gets the `thisUpdate` date from the CRL. The ASN.1 definition for this is:

```
thisUpdate   ChoiceOfTime
ChoiceOfTime ::= CHOICE {
    utcTime        UTCTime,
    generalTime    GeneralizedTime }
```

Returns: the `thisUpdate` date from the CRL.

---

getVersion()

```
public abstract int getVersion()
```

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

```
version    Version OPTIONAL,
            -- if present, must be v2
Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }
            -- v3 does not apply to CRLs but appears for consistency
            -- with definition of Version for certs
```

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

---

hashCode()

```
public int hashCode()
```

Returns a hashcode value for this CRL from its encoded form.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: the hashcode value.

verify(PublicKey)

```
public abstract void verify(java.security.PublicKey key)
```

Verifies that this CRL was signed using the private key that corresponds to the given public key.

Parameters:

`key` - the PublicKey used to carry out the verification.

Throws:

`NoSuchAlgorithmException` - on unsupported signature algorithms.

`InvalidKeyException` - on incorrect key.

`NoSuchProviderException` - if there's no default provider.

`SignatureException` - on signature errors.

`CRLException`[87] - on encoding errors.

verify(PublicKey, String)

```
public abstract void verify(java.security.PublicKey key, java.lang.String sigProvider)
```

Verifies that this CRL was signed using the private key that corresponds to the given public key. This method uses the signature verification engine supplied by the given provider.

Parameters:

`key` - the PublicKey used to carry out the verification.

`sigProvider` - the name of the signature provider.

Throws:

`NoSuchAlgorithmException` - on unsupported signature algorithms.

`InvalidKeyException` - on incorrect key.

`NoSuchProviderException` - on incorrect provider.

`SignatureException` - on signature errors.

`CRLException`[87] - on encoding errors.

java.security.cert
X509CRLEntry

Syntax

```
public abstract class X509CRLEntry implements X509Extension[178]

java.lang.Object
  |
  +--java.security.cert.X509CRLEntry
```

All Implemented Interfaces: X509Extension[178]

Description

Abstract class for a revoked certificate in a CRL (Certificate Revocation List). The ASN.1 definition for *revokedCertificates* is:

```
    revokedCertificates    SEQUENCE OF SEQUENCE  {
        userCertificate    CertificateSerialNumber,
        revocationDate     ChoiceOfTime,
        crlEntryExtensions Extensions OPTIONAL
                           -- if present, must be v2
    }  OPTIONAL
    CertificateSerialNumber  ::=  INTEGER
    Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension
    Extension  ::=  SEQUENCE  {
        extnId        OBJECT IDENTIFIER,
        critical      BOOLEAN DEFAULT FALSE,
        extnValue     OCTET STRING
                      -- contains a DER encoding of a value
                      -- of the type registered for use with
                      -- the extnId object identifier value
    }
```

See Also: X509CRL[161], X509Extension[178]

Member Summary		
Constructors		
	public	X509CRLEntry()[169]
Methods		
	public boolean	equals(Object)[169]
		Compares this CRL entry for equality with the given object.
	public abstract byte	getEncoded()[170]
		Returns the ASN.1 DER-encoded form of this CRL Entry, that is the inner SEQUENCE.
	public abstract Date	getRevocationDate()[170]
		Gets the revocation date from this X509CRLEntry, the *revocationDate*.

Member Summary	
public abstract BigInteger	getSerialNumber()[170] Gets the serial number from this X509CRLEntry, the *userCertificate*.
public abstract boolean	hasExtensions()[170] Returns true if this CRL entry has extensions.
public int	hashCode()[170] Returns a hashcode value for this CRL entry from its encoded form.
public abstract String	toString()[170] Returns a string representation of this CRL entry.

Inherited Member Summary
Methods inherited from class java.lang.Object getClass, clone, notify, notifyAll, wait, wait, wait, finalize
Methods inherited from interface X509Extension[178] hasUnsupportedCriticalExtension()[180], getCriticalExtensionOIDs()[179], getNonCriticalExtensionOIDs()[180], getExtensionValue(String)[179]

Constructors

X509CRLEntry()

```
public X509CRLEntry()
```

Methods

equals(Object)

```
public boolean equals(java.lang.Object other)
```

Compares this CRL entry for equality with the given object. If the `other` object is an `instanceof X509CRLEntry`, then its encoded form (the inner SEQUENCE) is retrieved and compared with the encoded form of this CRL entry.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:
`other` - the object to test for equality with this CRL entry.

Returns: true iff the encoded forms of the two CRL entries match, false otherwise.

getEncoded()

```
public abstract byte[] getEncoded()
```

Returns the ASN.1 DER-encoded form of this CRL Entry, that is the inner SEQUENCE.

Returns: the encoded form of this certificate

Throws:
CRLException[87] - if an encoding error occurs.

getRevocationDate()

```
public abstract java.util.Date getRevocationDate()
```

Gets the revocation date from this X509CRLEntry, the *revocationDate*.

Returns: the revocation date.

getSerialNumber()

```
public abstract java.math.BigInteger getSerialNumber()
```

Gets the serial number from this X509CRLEntry, the *userCertificate*.

Returns: the serial number.

hasExtensions()

```
public abstract boolean hasExtensions()
```

Returns true if this CRL entry has extensions.

Returns: true if this entry has extensions, false otherwise.

hashCode()

```
public int hashCode()
```

Returns a hashcode value for this CRL entry from its encoded form.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: the hashcode value.

toString()

```
public abstract java.lang.String toString()
```

Returns a string representation of this CRL entry.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a string representation of this CRL entry.

java.security.cert
X509CRLSelector

Syntax

```
public class X509CRLSelector implements CRLSelector89

java.lang.Object
  |
  +--java.security.cert.X509CRLSelector
```

All Implemented Interfaces: java.lang.Cloneable,CRLSelector89

Description

A `CRLSelector` that selects `X509CRLs` that match all specified criteria. This class is particularly useful when selecting CRLs from a `CertStore` to check revocation status of a particular certificate.

When first constructed, an `X509CRLSelector` has no criteria enabled and each of the `get` methods return a default value (`null`). Therefore, the `match(CRL)`175 method would return `true` for any `X509CRL`. Typically, several criteria are enabled (by calling `setIssuerNames(Collection)`176 or `setDateAndTime(Date)`176 , for instance) and then the `X509CRLSelector` is passed to `getCRLs(CRLSelector)`70 or some similar method.

Please refer to RFC 2459 for definitions of the X.509 CRL fields and extensions mentioned below.

Concurrent Access

Unless otherwise specified, the methods defined in this class are not thread-safe. Multiple threads that need to access a single object concurrently should synchronize amongst themselves and provide the necessary locking. Multiple threads each manipulating separate objects need not synchronize.

Since: 1.4

See Also: CRLSelector89, X509CRL161

Member Summary

Constructors

public	`X509CRLSelector()`173
	Creates an `X509CRLSelector`.

Methods

public void	`addIssuerName(byte[])`173
	Adds a name to the issuerNames criterion.
public void	`addIssuerName(String)`173
	Adds a name to the issuerNames criterion.

Member Summary

public Object	clone()[173] Returns a copy of this object.
public boolean	equals(Object)[174] Compares this object for equality with the specified object.
public X509Certificate	getCertificateChecking()[174] Returns the certificate being checked.
public Date	getDateAndTime()[174] Returns the dateAndTime criterion.
public Collection	getIssuerNames()[174] Returns a copy of the issuerNames criterion.
public BigInteger	getMaxCRL()[175] Returns the maxCRLNumber criterion.
public BigInteger	getMinCRL()[175] Returns the minCRLNumber criterion.
public int	hashCode()[175] Returns a hash code value for this object.
public boolean	match(CRL)[175] Decides whether a CRL should be selected.
public void	setCertificateChecking(X509Certificate)[175] Sets the certificate being checked.
public void	setDateAndTime(Date)[176] Sets the dateAndTime criterion.
public void	setIssuerNames(Collection)[176] Sets the issuerNames criterion.
public void	setMaxCRLNumber(BigInteger)[177] Sets the maxCRLNumber criterion.
public void	setMinCRLNumber(BigInteger)[177] Sets the minCRLNumber criterion.
public String	toString()[177] Returns a printable representation of the X509CRLSelector.

Inherited Member Summary

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait, finalize

Constructors

X509CRLSelector()

```
public X509CRLSelector()
```

Creates an `X509CRLSelector`. Initially, no criteria are set so any `X509CRL` will match.

Methods

addIssuerName(byte[])

```
public void addIssuerName(byte[] name)
```

Adds a name to the issuerNames criterion. The issuer distinguished name in the `X509CRL` must match at least one of the specified distinguished names.

This method allows the caller to add a name to the set of issuer names which `X509CRLs` may contain. The specified name is added to any previous value for the issuerNames criterion. If the specified name is a duplicate, it may be ignored. If a name is specified as a byte array, it should contain a single DER encoded distinguished name, as defined in X.501. The ASN.1 notation for this structure is as follows.

The name is provided as a byte array. This byte array should contain a single DER encoded distinguished name, as defined in X.501. The ASN.1 notation for this structure appears in the documentation for `setIssuerNames(Collection)`[176] .

Note that the byte array supplied here is cloned to protect against subsequent modifications.

Parameters:
`name` - a byte array containing the name in ASN.1 DER encoded form

Throws:
`IOException` - if a parsing error occurs

addIssuerName(String)

```
public void addIssuerName(java.lang.String name)
```

Adds a name to the issuerNames criterion. The issuer distinguished name in the `X509CRL` must match at least one of the specified distinguished names.

This method allows the caller to add a name to the set of issuer names which `X509CRLs` may contain. The specified name is added to any previous value for the issuerNames criterion. If the specified name is a duplicate, it may be ignored.

Parameters:
`name` - the name in RFC 1779 form

Throws:
`IOException` - if a parsing error occurs

clone()

`public java.lang.Object clone()`

Returns a copy of this object.

Specified By: `clone()`$_{173}$ in interface `X509CRLSelector`$_{171}$

Overrides: java.lang.Object.clone() in class java.lang.Object

Returns: the copy

---

equals(Object)

`public boolean equals(java.lang.Object other)`

Compares this object for equality with the specified object. If the other object is not an `X509CRLSelector`, return `false`. Otherwise, return `true` if the parameters of the objects are equal.

Overrides: java.lang.Object.equals(java.lang.Object) in class java.lang.Object

Parameters:
`other` - the object to test for equality

Returns: `true` if the objects are equal, `false` otherwise

---

getCertificateChecking()

`public X509Certificate`$_{132}$ `getCertificateChecking()`

Returns the certificate being checked. This is not a criterion. Rather, it is optional information that may help a `CertStore` find CRLs that would be relevant when checking revocation for the specified certificate. If the value returned is `null`, then no such optional information is provided.

Returns: the certificate being checked (or `null`)

---

getDateAndTime()

`public java.util.Date getDateAndTime()`

Returns the dateAndTime criterion. The specified date must be equal to or later than the value of the thisUpdate component of the `X509CRL` and earlier than the value of the nextUpdate component. There is no match if the `X509CRL` does not contain a nextUpdate component. If `null`, no dateAndTime check will be done.

Note that the `Date` returned is cloned to protect against subsequent modifications.

Returns: the `Date` to match against (or `null`)

---

getIssuerNames()

`public java.util.Collection getIssuerNames()`

Returns a copy of the issuerNames criterion. The issuer distinguished name in the `X509CRL` must match at least one of the specified distinguished names. If the value returned is `null`, any issuer distinguished name will do.

If the value returned is not `null`, it is a `Collection` of names. Each name is a `String` or a byte array representing a distinguished name (in RFC 1779 or ASN.1 DER encoded form, respectively). Note that the `Collection` returned may contain duplicate names.

If a name is specified as a byte array, it should contain a single DER encoded distinguished name, as defined in X.501. The ASN.1 notation for this structure is given in the documentation for `setIssuerNames(Collection)`[176].

Note that a deep copy is performed on the `Collection` to protect against subsequent modifications.

Returns: a `Collection` of names (or `null`)

---

getMaxCRL()

```
public java.math.BigInteger getMaxCRL()
```

Returns the maxCRLNumber criterion. The `X509CRL` must have a CRL number extension whose value is less than or equal to the specified value. If `null`, no maxCRLNumber check will be done.

Returns: the maximum CRL number accepted (or `null`)

---

getMinCRL()

```
public java.math.BigInteger getMinCRL()
```

Returns the minCRLNumber criterion. The `X509CRL` must have a CRL number extension whose value is greater than or equal to the specified value. If `null`, no minCRLNumber check will be done.

Returns: the minimum CRL number accepted (or `null`)

---

hashCode()

```
public int hashCode()
```

Returns a hash code value for this object.

Overrides: java.lang.Object.hashCode() in class java.lang.Object

Returns: a hash code value

---

match(CRL)

```
public boolean match(CRL[85] crl)
```

Decides whether a `CRL` should be selected.

Specified By: `match(CRL)`[175] in interface `X509CRLSelector`[171]

Parameters:
`crl` - the `CRL` to be checked

Returns: `true` if the `CRL` should be selected, `false` otherwise

---

setCertificateChecking(X509Certificate)

```
public void setCertificateChecking(X509Certificate[132] cert)
```

Sets the certificate being checked. This is not a criterion. Rather, it is optional information that may help a `CertStore` find CRLs that would be relevant when checking revocation for the specified certificate. If `null` is specified, then no such optional information is provided.

Parameters:

`cert` - the `X509Certificate` being checked (or `null`)

---

setDateAndTime(Date)

```
public void setDateAndTime(java.util.Date dateAndTime)
```

Sets the dateAndTime criterion. The specified date must be equal to or later than the value of the thisUpdate component of the `X509CRL` and earlier than the value of the nextUpdate component. There is no match if the `X509CRL` does not contain a nextUpdate component. If `null`, no dateAndTime check will be done.

Note that the `Date` supplied here is cloned to protect against subsequent modifications.

Parameters:

`dateAndTime` - the `Date` to match against (or `null`)

---

setIssuerNames(Collection)

```
public void setIssuerNames(java.util.Collection names)
```

Sets the issuerNames criterion. The issuer distinguished name in the `X509CRL` must match at least one of the specified distinguished names. If `null`, any issuer distinguished name will do.

This method allows the caller to specify, with a single method call, the complete set of issuer names which `X509CRLs` may contain. The specified value replaces the previous value for the issuerNames criterion.

The `names` parameter (if not `null`) is a `Collection` of names. Each name is a `String` or a byte array representing a distinguished name (in RFC 1779 or ASN.1 DER encoded form, respectively). If `null` is supplied as the value for this argument, no issuerNames check will be performed.

Note that the `names` parameter can contain duplicate distinguished names, but they may be removed from the `Collection` of names returned by the `getIssuerNames()`[174] method.

If a name is specified as a byte array, it should contain a single DER encoded distinguished name, as defined in X.501. The ASN.1 notation for this structure is as follows.

```
Name ::= CHOICE {
  RDNSequence }
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::=
  SET SIZE (1 .. MAX) OF AttributeTypeAndValue
AttributeTypeAndValue ::= SEQUENCE {
  type     AttributeType,
  value    AttributeValue }
AttributeType ::= OBJECT IDENTIFIER
AttributeValue ::= ANY DEFINED BY AttributeType
....
DirectoryString ::= CHOICE {
      teletexString           TeletexString (SIZE (1..MAX)),
      printableString         PrintableString (SIZE (1..MAX)),
      universalString         UniversalString (SIZE (1..MAX)),
      utf8String              UTF8String (SIZE (1.. MAX)),
      bmpString               BMPString (SIZE (1..MAX)) }
```

Note that a deep copy is performed on the `Collection` to protect against subsequent modifications.

Parameters:

`names` - a `Collection` of names (or `null`)

Throws:

`IOException` - if a parsing error occurs

setMaxCRLNumber(BigInteger)

```
public void setMaxCRLNumber(java.math.BigInteger maxCRL)
```

Sets the maxCRLNumber criterion. The `X509CRL` must have a CRL number extension whose value is less than or equal to the specified value. If `null`, no maxCRLNumber check will be done.

Parameters:

`maxCRL` - the maximum CRL number accepted (or `null`)

setMinCRLNumber(BigInteger)

```
public void setMinCRLNumber(java.math.BigInteger minCRL)
```

Sets the minCRLNumber criterion. The `X509CRL` must have a CRL number extension whose value is greater than or equal to the specified value. If `null`, no minCRLNumber check will be done.

Parameters:

`minCRL` - the minimum CRL number accepted (or `null`)

toString()

```
public java.lang.String toString()
```

Returns a printable representation of the `X509CRLSelector`.

Overrides: java.lang.Object.toString() in class java.lang.Object

Returns: a `String` describing the contents of the `X509CRLSelector`.

java.security.cert
X509Extension

Syntax
```
public interface X509Extension
```

All Known Implementing Classes: `X509Certificate`[132], `X509CRL`[161], `X509CRLEntry`[168]

Description
Interface for an X.509 extension.

The extensions defined for X.509 v3 `X509Certificate`[132] and v2 `X509CRL`[161] (Certificate Revocation Lists) provide methods for associating additional attributes with users or public keys, for managing the certification hierarchy, and for managing CRL distribution. The X.509 extensions format also allows communities to define private extensions to carry information unique to those communities.

Each extension in a certificate/CRL may be designated as critical or non-critical. A certificate/CRL-using system (an application validating a certificate/CRL) must reject the certificate/CRL if it encounters a critical extension it does not recognize. A non-critical extension may be ignored if it is not recognized.

The ASN.1 definition for this is:

```
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension
Extension  ::=  SEQUENCE  {
    extnId        OBJECT IDENTIFIER,
    critical      BOOLEAN DEFAULT FALSE,
    extnValue     OCTET STRING
                  -- contains a DER encoding of a value
                  -- of the type registered for use with
                  -- the extnId object identifier value
}
```

Since not all extensions are known, the `getExtensionValue` method returns the DER-encoded OCTET STRING of the extension value (i.e., the `extnValue`). This can then be handled by a *Class* that understands the extension.

Member Summary

Methods

public Set	`getCriticalExtensionOIDs()`[179] Gets a Set of the OID strings for the extension(s) marked CRITICAL in the certificate/CRL managed by the object implementing this interface.
public byte	`getExtensionValue(String)`[179] Gets the DER-encoded OCTET string for the extension value (*extnValue*) identified by the passed-in `oid` String.
public Set	`getNonCriticalExtensionOIDs()`[180] Gets a Set of the OID strings for the extension(s) marked NON-CRITICAL in the certificate/CRL managed by the object implementing this interface.

Member Summary	
public boolean	hasUnsupportedCriticalExtension()[180] Check if there is a critical extension that is not supported.

Methods

getCriticalExtensionOIDs()

```
public java.util.Set getCriticalExtensionOIDs()
```

Gets a Set of the OID strings for the extension(s) marked CRITICAL in the certificate/CRL managed by the object implementing this interface. Here is sample code to get a Set of critical extensions from an X509Certificate and print the OIDs:

```
InputStream inStrm = new FileInputStream("DER-encoded-Cert");
CertificateFactory cf = CertificateFactory.getInstance("X.509");
X509Certificate cert = (X509Certificate)cf.generateCertificate(inStrm);
inStrm.close();
Set critSet = cert.getCriticalExtensionOIDs();
if (critSet != null && !critSet.isEmpty()) {
    System.out.println("Set of critical extensions:");
    for (Iterator i = critSet.iterator(); i.hasNext();) {
        String oid = (String)i.next();
        System.out.println(oid);
    }
}
```

Returns: a Set (or an empty Set if none are marked critical) of the extension OID strings for extensions that are marked critical. If there are no extensions present at all, then this method returns null.

getExtensionValue(String)

```
public byte[] getExtensionValue(java.lang.String oid)
```

Gets the DER-encoded OCTET string for the extension value (*extnValue*) identified by the passed-in `oid` String. The `oid` string is represented by a set of positive whole numbers separated by periods.

For example:

OID *(Object Identifier)*	Extension Name
2.5.29.14	SubjectKeyIdentifier
2.5.29.15	KeyUsage
2.5.29.16	PrivateKeyUsage
2.5.29.17	SubjectAlternativeName
2.5.29.18	IssuerAlternativeName

2.5.29.19	BasicConstraints
2.5.29.30	NameConstraints
2.5.29.33	PolicyMappings
2.5.29.35	AuthorityKeyIdentifier
2.5.29.36	PolicyConstraints

Parameters:
`oid` - the Object Identifier value for the extension.

Returns: the DER-encoded octet string of the extension value or null if it is not present.

---

getNonCriticalExtensionOIDs()

```
public java.util.Set getNonCriticalExtensionOIDs()
```

Gets a Set of the OID strings for the extension(s) marked NON-CRITICAL in the certificate/CRL managed by the object implementing this interface. Here is sample code to get a Set of non-critical extensions from an X509CRL revoked certificate entry and print the OIDs:

```
InputStream inStrm = new FileInputStream("DER-encoded-CRL");
CertificateFactory cf = CertificateFactory.getInstance("X.509");
X509CRL crl = (X509CRL)cf.generateCRL(inStrm);
inStrm.close();
byte[] certData = <DER-encoded certificate data>
ByteArrayInputStream bais = new ByteArrayInputStream(certData);
X509Certificate cert = (X509Certificate)cf.generateCertificate(bais);
bais.close();
X509CRLEntry badCert =
             crl.getRevokedCertificate(cert.getSerialNumber());
if (badCert != null) {
    Set nonCritSet = badCert.getNonCriticalExtensionOIDs();
    if (nonCritSet != null)
        for (Iterator i = nonCritSet.iterator(); i.hasNext();) {
            String oid = (String)i.next();
            System.out.println(oid);
        }
}
```

Returns: a Set (or an empty Set if none are marked non-critical) of the extension OID strings for extensions that are marked non-critical. If there are no extensions present at all, then this method returns null.

---

hasUnsupportedCriticalExtension()

```
public boolean hasUnsupportedCriticalExtension()
```

Check if there is a critical extension that is not supported.

Returns: `true` if a critical extension is found that is not supported, otherwise `false`.

Index

A

B

C

E

G

H

I

J

L

M

P

R

S

T

V

W

X