| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/X500Principal.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PREV CLASS   [**NEXT CLASS**](http://docs.google.com/javax/security/auth/x500/X500PrivateCredential.html) | [**FRAMES**](http://docs.google.com/index.html?javax/security/auth/x500/X500Principal.html)    [**NO FRAMES**](http://docs.google.com/X500Principal.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#3rdcrjn) | [METHOD](#44sinio) |

## **javax.security.auth.x500**

Class X500Principal

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.security.auth.x500.X500Principal**

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

public final class **X500Principal**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Principal](http://docs.google.com/java/security/Principal.html), [Serializable](http://docs.google.com/java/io/Serializable.html)

This class represents an X.500 Principal. X500Principals are represented by distinguished names such as "CN=Duke, OU=JavaSoft, O=Sun Microsystems, C=US".

This class can be instantiated by using a string representation of the distinguished name, or by using the ASN.1 DER encoded byte representation of the distinguished name. The current specification for the string representation of a distinguished name is defined in [RFC 2253](http://www.ietf.org/rfc/rfc2253.txt). This class, however, accepts string formats from both RFC 2253 and [RFC 1779](http://www.ietf.org/rfc/rfc1779.txt), and also recognizes attribute type keywords whose OIDs (Object Identifiers) are defined in [RFC 2459](http://www.ietf.org/rfc/rfc2459.txt).

The string representation for this X500Principal can be obtained by calling the getName methods.

Note that the getSubjectX500Principal and getIssuerX500Principal methods of X509Certificate return X500Principals representing the issuer and subject fields of the certificate.

**Since:** 1.4 **See Also:**[X509Certificate](http://docs.google.com/java/security/cert/X509Certificate.html), [Serialized Form](http://docs.google.com/serialized-form.html#javax.security.auth.x500.X500Principal)

| **Field Summary** | |
| --- | --- |
| static [String](http://docs.google.com/java/lang/String.html) | [**CANONICAL**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#CANONICAL)            Canonical String format of Distinguished Names. |
| static [String](http://docs.google.com/java/lang/String.html) | [**RFC1779**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#RFC1779)            RFC 1779 String format of Distinguished Names. |
| static [String](http://docs.google.com/java/lang/String.html) | [**RFC2253**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#RFC2253)            RFC 2253 String format of Distinguished Names. |

| **Constructor Summary** | |
| --- | --- |
| [**X500Principal**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#X500Principal(byte%5B%5D))(byte[] name)            Creates an X500Principal from a distinguished name in ASN.1 DER encoded form. |
| [**X500Principal**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#X500Principal(java.io.InputStream))([InputStream](http://docs.google.com/java/io/InputStream.html) is)            Creates an X500Principal from an InputStream containing the distinguished name in ASN.1 DER encoded form. |
| [**X500Principal**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#X500Principal(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Creates an X500Principal from a string representation of an X.500 distinguished name (ex: "CN=Duke, OU=JavaSoft, O=Sun Microsystems, C=US"). |
| [**X500Principal**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#X500Principal(java.lang.String,%20java.util.Map))([String](http://docs.google.com/java/lang/String.html) name, [Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[String](http://docs.google.com/java/lang/String.html)> keywordMap)            Creates an X500Principal from a string representation of an X.500 distinguished name (ex: "CN=Duke, OU=JavaSoft, O=Sun Microsystems, C=US"). |

| **Method Summary** | |
| --- | --- |
| boolean | [**equals**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) o)            Compares the specified Object with this X500Principal for equality. |
| byte[] | [**getEncoded**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#getEncoded())()            Returns the distinguished name in ASN.1 DER encoded form. |
| [String](http://docs.google.com/java/lang/String.html) | [**getName**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#getName())()            Returns a string representation of the X.500 distinguished name using the format defined in RFC 2253. |
| [String](http://docs.google.com/java/lang/String.html) | [**getName**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#getName(java.lang.String))([String](http://docs.google.com/java/lang/String.html) format)            Returns a string representation of the X.500 distinguished name using the specified format. |
| [String](http://docs.google.com/java/lang/String.html) | [**getName**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#getName(java.lang.String,%20java.util.Map))([String](http://docs.google.com/java/lang/String.html) format, [Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[String](http://docs.google.com/java/lang/String.html)> oidMap)            Returns a string representation of the X.500 distinguished name using the specified format. |
| int | [**hashCode**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#hashCode())()            Return a hash code for this X500Principal. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/javax/security/auth/x500/X500Principal.html#toString())()            Return a user-friendly string representation of this X500Principal. |

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

| **Field Detail** |
| --- |

### RFC1779

public static final [String](http://docs.google.com/java/lang/String.html) **RFC1779**

RFC 1779 String format of Distinguished Names.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.security.auth.x500.X500Principal.RFC1779)

### RFC2253

public static final [String](http://docs.google.com/java/lang/String.html) **RFC2253**

RFC 2253 String format of Distinguished Names.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.security.auth.x500.X500Principal.RFC2253)

### CANONICAL

public static final [String](http://docs.google.com/java/lang/String.html) **CANONICAL**

Canonical String format of Distinguished Names.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.security.auth.x500.X500Principal.CANONICAL)

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

### X500Principal

public **X500Principal**([String](http://docs.google.com/java/lang/String.html) name)

Creates an X500Principal from a string representation of an X.500 distinguished name (ex: "CN=Duke, OU=JavaSoft, O=Sun Microsystems, C=US"). The distinguished name must be specified using the grammar defined in RFC 1779 or RFC 2253 (either format is acceptable).

This constructor recognizes the attribute type keywords defined in RFC 1779 and RFC 2253 (and listed in [getName(String format)](http://docs.google.com/javax/security/auth/x500/X500Principal.html#getName(java.lang.String))), as well as the T, DNQ or DNQUALIFIER, SURNAME, GIVENNAME, INITIALS, GENERATION, EMAILADDRESS, and SERIALNUMBER keywords whose OIDs are defined in RFC 2459 and its successor. Any other attribute type must be specified as an OID.

**Parameters:**name - an X.500 distinguished name in RFC 1779 or RFC 2253 format **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the name is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the name is improperly specified

### X500Principal

public **X500Principal**([String](http://docs.google.com/java/lang/String.html) name,  
 [Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[String](http://docs.google.com/java/lang/String.html)> keywordMap)

Creates an X500Principal from a string representation of an X.500 distinguished name (ex: "CN=Duke, OU=JavaSoft, O=Sun Microsystems, C=US"). The distinguished name must be specified using the grammar defined in RFC 1779 or RFC 2253 (either format is acceptable).

This constructor recognizes the attribute type keywords specified in [X500Principal(String)](http://docs.google.com/javax/security/auth/x500/X500Principal.html#X500Principal(java.lang.String)) and also recognizes additional keywords that have entries in the keywordMap parameter. Keyword entries in the keywordMap take precedence over the default keywords recognized by X500Principal(String). Keywords MUST be specified in all upper-case, otherwise they will be ignored. Improperly specified keywords are ignored; however if a keyword in the name maps to an improperly specified OID, an IllegalArgumentException is thrown. It is permissible to have 2 different keywords that map to the same OID.

**Parameters:**name - an X.500 distinguished name in RFC 1779 or RFC 2253 formatkeywordMap - an attribute type keyword map, where each key is a keyword String that maps to a corresponding object identifier in String form (a sequence of nonnegative integers separated by periods). The map may be empty but never null. **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if name or keywordMap is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the name is improperly specified or a keyword in the name maps to an OID that is not in the correct form**Since:** 1.6

### X500Principal

public **X500Principal**(byte[] name)

Creates an X500Principal from a distinguished name in ASN.1 DER encoded form. 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)) }

**Parameters:**name - a byte array containing the distinguished name in ASN.1 DER encoded form **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if an encoding error occurs (incorrect form for DN)

### X500Principal

public **X500Principal**([InputStream](http://docs.google.com/java/io/InputStream.html) is)

Creates an X500Principal from an InputStream containing the distinguished name in ASN.1 DER encoded form. The ASN.1 notation for this structure is supplied in the documentation for [X500Principal(byte[] name)](http://docs.google.com/javax/security/auth/x500/X500Principal.html#X500Principal(byte%5B%5D)).

The read position of the input stream is positioned to the next available byte after the encoded distinguished name.

**Parameters:**is - an InputStream containing the distinguished name in ASN.1 DER encoded form **Throws:** [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the InputStream is null [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if an encoding error occurs (incorrect form for DN)

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

### getName

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

Returns a string representation of the X.500 distinguished name using the format defined in RFC 2253.

This method is equivalent to calling getName(X500Principal.RFC2253).

**Specified by:**[getName](http://docs.google.com/java/security/Principal.html#getName()) in interface [Principal](http://docs.google.com/java/security/Principal.html) **Returns:**the distinguished name of this X500Principal

### getName

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

Returns a string representation of the X.500 distinguished name using the specified format. Valid values for the format are "RFC1779", "RFC2253", and "CANONICAL" (case insensitive).

If "RFC1779" is specified as the format, this method emits the attribute type keywords defined in RFC 1779 (CN, L, ST, O, OU, C, STREET). Any other attribute type is emitted as an OID.

If "RFC2253" is specified as the format, this method emits the attribute type keywords defined in RFC 2253 (CN, L, ST, O, OU, C, STREET, DC, UID). Any other attribute type is emitted as an OID. Under a strict reading, RFC 2253 only specifies a UTF-8 string representation. The String returned by this method is the Unicode string achieved by decoding this UTF-8 representation.

If "CANONICAL" is specified as the format, this method returns an RFC 2253 conformant string representation with the following additional canonicalizations:

1. Leading zeros are removed from attribute types that are encoded as dotted decimal OIDs
2. DirectoryString attribute values of type PrintableString and UTF8String are not output in hexadecimal format
3. DirectoryString attribute values of types other than PrintableString and UTF8String are output in hexadecimal format
4. Leading and trailing white space characters are removed from non-hexadecimal attribute values (unless the value consists entirely of white space characters)
5. Internal substrings of one or more white space characters are converted to a single space in non-hexadecimal attribute values
6. Relative Distinguished Names containing more than one Attribute Value Assertion (AVA) are output in the following order: an alphabetical ordering of AVAs containing standard keywords, followed by a numeric ordering of AVAs containing OID keywords.
7. The only characters in attribute values that are escaped are those which section 2.4 of RFC 2253 states must be escaped (they are escaped using a preceding backslash character)
8. The entire name is converted to upper case using String.toUpperCase(Locale.US)
9. The entire name is converted to lower case using String.toLowerCase(Locale.US)
10. The name is finally normalized using normalization form KD, as described in the Unicode Standard and UAX #15

Additional standard formats may be introduced in the future.

**Parameters:**format - the format to use **Returns:**a string representation of this X500Principal using the specified format **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified format is invalid or null

### getName

public [String](http://docs.google.com/java/lang/String.html) **getName**([String](http://docs.google.com/java/lang/String.html) format,  
 [Map](http://docs.google.com/java/util/Map.html)<[String](http://docs.google.com/java/lang/String.html),[String](http://docs.google.com/java/lang/String.html)> oidMap)

Returns a string representation of the X.500 distinguished name using the specified format. Valid values for the format are "RFC1779" and "RFC2253" (case insensitive). "CANONICAL" is not permitted and an IllegalArgumentException will be thrown.

This method returns Strings in the format as specified in [getName(String)](http://docs.google.com/javax/security/auth/x500/X500Principal.html#getName(java.lang.String)) and also emits additional attribute type keywords for OIDs that have entries in the oidMap parameter. OID entries in the oidMap take precedence over the default OIDs recognized by getName(String). Improperly specified OIDs are ignored; however if an OID in the name maps to an improperly specified keyword, an IllegalArgumentException is thrown.

Additional standard formats may be introduced in the future.

Warning: additional attribute type keywords may not be recognized by other implementations; therefore do not use this method if you are unsure if these keywords will be recognized by other implementations.

**Parameters:**format - the format to useoidMap - an OID map, where each key is an object identifier in String form (a sequence of nonnegative integers separated by periods) that maps to a corresponding attribute type keyword String. The map may be empty but never null. **Returns:**a string representation of this X500Principal using the specified format **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if the specified format is invalid, null, or an OID in the name maps to an improperly specified keyword [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if oidMap is null**Since:** 1.6

### getEncoded

public byte[] **getEncoded**()

Returns the distinguished name in ASN.1 DER encoded form. The ASN.1 notation for this structure is supplied in the documentation for [X500Principal(byte[] name)](http://docs.google.com/javax/security/auth/x500/X500Principal.html#X500Principal(byte%5B%5D)).

Note that the byte array returned is cloned to protect against subsequent modifications.

**Returns:**a byte array containing the distinguished name in ASN.1 DER encoded form

### toString

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

Return a user-friendly string representation of this X500Principal.

**Specified by:**[toString](http://docs.google.com/java/security/Principal.html#toString()) in interface [Principal](http://docs.google.com/java/security/Principal.html)**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of this X500Principal

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) o)

Compares the specified Object with this X500Principal for equality.

Specifically, this method returns true if the Object *o* is an X500Principal and if the respective canonical string representations (obtained via the getName(X500Principal.CANONICAL) method) of this object and *o* are equal.

This implementation is compliant with the requirements of RFC 2459.

**Specified by:**[equals](http://docs.google.com/java/security/Principal.html#equals(java.lang.Object)) in interface [Principal](http://docs.google.com/java/security/Principal.html)**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**o - Object to be compared for equality with this X500Principal **Returns:**true if the specified Object is equal to this X500Principal, false otherwise**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### hashCode

public int **hashCode**()

Return a hash code for this X500Principal.

The hash code is calculated via: getName(X500Principal.CANONICAL).hashCode()

**Specified by:**[hashCode](http://docs.google.com/java/security/Principal.html#hashCode()) in interface [Principal](http://docs.google.com/java/security/Principal.html)**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a hash code for this X500Principal**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/X500Principal.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PREV CLASS   [**NEXT CLASS**](http://docs.google.com/javax/security/auth/x500/X500PrivateCredential.html) | [**FRAMES**](http://docs.google.com/index.html?javax/security/auth/x500/X500Principal.html)    [**NO FRAMES**](http://docs.google.com/X500Principal.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#3rdcrjn) | [METHOD](#44sinio) |

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

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

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