IBM Java JSSE FIPS 140-2 Cryptographic Module

Security Policy

IBM JAVA JSSE FIPS 140-2
Cryptographic module
Revision: 1.1

March 2004

Status: Final

First Edition (March 2004)
This edition applies to the First Edition of the IBMJSSEFIPS – Security Policy and to all subsequent versions until otherwise indicated in new editions.

Table of Contents

Introduction

The IBM® Java® JSSE (Java Secure Sockets Extension) FIPS 140-2 Cryptographic Module (IBMJSSEFIPS) for Multi-platforms is a scalable, multi-purpose Secure Sockets provider that supports only FIPS approved TLS cipher suites via the Java2 Application Programming Interfaces (APIs). The IBM Java JSSE FIPS provider (hereafter referred to as IBMJSSEFIPS) comprises the following Federal Information Processing Standards (FIPS) 140-2 [1] compliant components:

- IBMJSSEFIPS for Solaris®, Windows®, AIX®, z/OS®, AS/400®, Linux® (Red Hat and SuSE®)

In order to meet the requirements set forth in the FIPS publication 140-2, the encryption algorithms utilized by the IBMJSSEFIPS provider are isolated into the IBMJSSEFIPS provider cryptographic module (hereafter referred to as cryptographic module), which is accessed by the product code via the Java JSSE framework APIs. As the IBMJSSEFIPS provider utilizes the cryptographic module in an approved manner, the product complies with the FIPS 140-2 requirements when properly configured.

This document focuses on the features and security policy provided by the cryptographic module, and describes how the module is designed to meet FIPS 140-2 compliance.

Operation of the Cryptographic Module

The cryptographic module must be utilized in a compliant manner, as described herein, to maintain FIPS 140-2 validation. It is the application and application administrator's responsibility to understand and deploy the proper configuration for compliance.

The module is available as a software module on multiple platforms. The platforms certified are outlined in the *Cryptographic Module Specification* section of this document. The module must be used in one of the specified environments.

An application utilizes the module through the interfaces specified in the *Cryptographic Module Interfaces* section of this document. A list of all services provided through these interfaces may be found in the *Cryptographic Module Services* section of this document.

The roles govern which of the services are available for operator use. The *Cryptographic Module Roles* section of this document details the access control policy of the module.

The module can provide for protection of sensitive data, such as keys or contexts. Information on key protection is outlined in the C*ryptographic Module Key Management* section. When the module is initialized, it validates its own integrity, and verifies the algorithms are functioning correctly. The *Cryptographic Module Self-Tests* section details the internal tests performed by the module.

The modules physical security relies on the physical security of the computer. Steps to deploy and maintain this secure environment are outlined in the *Cryptographic Module Physical Security* section of this document.

Module Components

The following table lists the module components:

Type	Name	Release	Date	Delivery
Software	IBM JSSE FIPS 140-2	1.1		

	Cryptographic Module JAR file (ibmjssefips.jar)			
Documentation	IBM JSSE FIPS 140-2 Cryptographic Module Security Policy	1.1		

Table 1: Module Component List for all platforms

Module description

IBMJSSEFIPS is an SSL (Secure Sockets Layer) interface that provides API's to allow users to write SSL applications. IBMJSSEFIPS uses an embedded FIPS-Approved module. IBM SSLite in Java (Certificate #406) for the actual SSL interfaces.

IBM SSLite in Java is a TLS (Transport Layer Security) V1.0 protocol implementation including PKI (Public Key Infrastructure) functionality for the hand-shake, in Java. This implementation, in turn relies exclusively on the internal FIPS-Approved IBM CryptoLite in Java module for all cryptographic functionality. See the IBM SSLite in Java Security Policy for more information.

Cryptographic Module Specification

An application utilizes the module through the interfaces specified in the *Cryptographic Module Interfaces* section of this document. A list of the basic services provided through these interfaces may be found in the *Cryptographic Module Services* section of this document. A complete list of all services and details on their usage can be found in the *Java Secure Socket Extension (JSSE) API User's Guide* and the associated *IBMJSSEProvider Class Documentation*.

The module was tested and certified to the following FIPS 140-2 defined levels:

Overall	Security Level 1
Cryptographic Module	Security Level 1
Ports and Interfaces	Security Level 1
Roles, Services, and Authentication	Security Level 1
Finite State Model	Security Level 1
Physical Security	Security Level 1
Operational Environment	Security Level 1
Key Management	Security Level 1
EMI/EMC	Security Level 1
Self-Tests	Security Level 1
Design Assurance	Security Level 1
Mitigation of Other Attacks	N/A

Table 2: FIPS validation level

The IBMJSSEFIPS Module is packaged in a single Java Archive File, which contains all the code for the module. IBMJSSEFIPS runs upon many platforms including Microsoft Windows 95 ®, Microsoft Windows 98 ®, Microsoft Windows Me ® and Microsoft Windows NT ®, Solaris ®, HP-UX ®, Linux ®, z/OS®, AS/400® and AIX ®; however, the IBMJSSEFIPS Module was not tested or validated upon each of these platforms as part of this effort.

As outlined in G.5 of the Implementation Guidance for FIPS 140-2, the module maintains its compliance on other operating systems, provided:

- The GPC uses the specified single user operating system/mode specific on the validation certificate, or another compatible single user operating system, and
- The source code of the software cryptographic module does not require modification prior to the recompilation to allow porting to another compatible single user operating system.

Since the IBMJSSEFIPS module is a pure Java implementation it should be able to run unmodified on any system that supports a Java Runtime of at least Version 1.3.1. Testing and validating the IBMJSSEFIPS package on the

following representative Windows and Unix platforms will demonstrate the above requirements:

IBMJSSEFIPS **was** tested and validated on a machine running the Microsoft Windows 2000® Advanced Server SP4 with JVM 1.4.1 and Microsoft Windows 2000® Professional with Service Pack 3 with JVM 1.3.1 03 and JVM 1.4.1 04. The software module maintains compliance when running on the Microsoft Windows 95 ®, Microsoft Windows 98®, Microsoft Windows Me®, Microsoft Windows NT®, Microsoft Windows 2000®, and Microsoft Windows XP® operating systems, as well as, JVMs at the 1.4.x level of those operating systems.

IBMJSSEFIPS **was** tested and validated on a machine running the AIX ® 5.2 operating system with JVM 1.3.1 and JVM 1.4.1. The software module maintains compliance when running on other versions of AIX®, as well as, JVMs at the 1.4.x level of those AIX® versions.

IBMJSSEFIPS **was** tested and validated on a machine running the Solaris® 5.8 operating system with JVM 1.3.1 and JVM 1.4.1. The software module maintains compliance when running on other UNIX based operating systems, such as HP-UX, as well as, JVMs at the 1.4.x level of those operating systems.

IBMJSSEFIPS **was** tested and validated on a machine running the Red Hat Linux Advanced Server 2.1 operating system with JVM 1.4.1 05. The software module maintains compliance when running on other Linux based operating systems, as well as, JVMs at the 1.3.1 or 1.4.x level of those operating systems.

IBMJSSEFIPS **was** tested and validated on a machine running the SuSE® Linux Enterprise Server 8.0 operating system with JVM 1.4.1 05. The software module maintains compliance when running on other Linux based operating systems, as well as, JVMs at the 1.3.1 or 1.4.x level of those operating systems.

IBMJSSEFIPS **was** tested and validated on a machine running the z/OS® V1R4 operating system with JVM 1.4.1. The software module maintains compliance when running on other z/OS operating system releases, as well as, JVMs at the 1.3.1 or 1.4.x level on those operating system releases.

IBMJSSEFIPS **was** tested and validated on a machine running the IBM Operating System/400 V5R2M0 operating system with JVM 1.4.1. The software module maintains compliance when running on other IBM Operating System/400

operating system releases, as well as, JVMs at the 1.3.1 or 1.4.x level on those operating system releases.

The module supports the following Approved algorithms:

This module uses the encryption algorithms implemented in the internal IBM SSLite in Java source code base. Please see section 5.3 of the SSLite Security policy document for all the supported algorithms. The list below is the list of algorithms that IBMJSSEFIPS uses when it implements the SSL interfaces.

HASH Functions

Algorithm	Specification	FIPS Approved
SHA	**FIPS180-1** Hash algorithm; hash size: 20 bytes; block size: 64 bytes.	Yes

Table 3: Supported Hash Functions

CIPHER Functions

Algorithm	Specification & Description	FIPS Approved
DES, **DES-CBC**	**FIPS 46-3** Symmetric block cipher; block size: 8 bytes; key size: 56 bits. For use in legacy systems only.	Yes
3DES, **3DES-CBC**	**FIPS 46-3** Triple DES has 112/168 bits key length depending on type of key.	Yes
AES, **AES-CBC**	**FIPS 197** AES has 128 to 256 bits key length.	Yes

Table 4: Supported Cipher Functions

Public Key

Algorithm	Specification	FIPS Approved
RSA **Sign/Verify**	Public key encryption/signature scheme. Typical key/data sizes: 512, 768, 1024 (typical), 2048 bits. PKCS#1	Yes
RSA Encrypt /	RSA specification and padding scheme:	No

Decrypt	PKCS#1	
Diffie-Hellman	Public key crypto system. Typical key/data sizes: 512, 768, 1024 (typical), 2048 bits. Used for key agreement.	No

Table 5: Supported Public Key Functions

Random Number Generators

Algorithm	**Specification**	**FIPS Approved**
PSEUDO Random Number Generator	**FIPS 186-2** **ANSI X9.31 1998** .	Yes
Universal Software Based Random Number Generator	Patented by IBM, EC Pat.No. EP1081591A2, The random number generator works reliably on variety of platforms without exploiting platform specific features. Entropy evaluation through statistical analysis. Performance: 20-1000 bits/seconds (Used to seed the Approved PRNG in FIPS mode)	No

Table 6: Supported Random Number Generators

The other Approved and non-Approved algorithms implemented by IBM CryptoLite in Java are not accessible directly from IBMJSSEFIPS. The IBMJSSEFIPS jar file has been obfuscated to not allow this access.

Cryptographic Module Interfaces

The cryptographic module is classified as a “multi-chip standalone unit” for FIPS 140-2 purposes. Thus, the module’s physical interfaces consist of those found as part of the computer’s hardware, such as the keyboard, mouse, disk drive, CD drive, network adapters, serial and USB ports, monitor, speakers, etc. The module’s logical interface is provided through the documented API.

Cryptographic Module Services

The module services are accessible from Java language programs through an Application Program Interface (API). The interface is defined in the *Java Secure Socket Extension (JSSE) API User's Guide* and associated *IBMJSSEProvider Class Documentation* with the restrictions outlined in the User Guidance section of this document.

This cryptographic module only supports the TLS protocol and the following cipher suites:

SSL_RSA_WITH_DES_CBC_SHA (hex number 0009),

SSL_RSA_FIPS_WITH_DES_CBC_SHA (hex number FEFE),

SSL_RSA_WITH_3DES_EDE_CBC_SHA (hex number 000A), and

SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (hex number FEFF)

SSL_RSA_WITH_AES_128_CBC_SHA (hex number 002F)

SSL_RSA_WITH_AES_256_CBC_SHA (hex number 0035)

SSL_DHE_RSA_WITH_AES_128_CBC_SHA (hex number 0033)

SSL_DHE_RSA_WITH_AES_256_CBC_SHA (hex number 0039)

SSL_DHE_RSA_WITH_DES_CBC_SHA (hex number 0015)

SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA (hex number 0016)

SSL_DHE_DSS_WITH_AES_128_CBC_SHA (hex number 0032)

SSL_DHE_DSS_WITH_AES_256_CBC_SHA (hex number 0038)

SSL_DHE_DSS_WITH_DES_CBC_SHA (hex number 0012)

SSL_RSA_EXPORT1024_WITH_DES_CBC_SHA (hex number 0062)

SSL_RSA_EXPORT_WITH_DES40_CBC_SHA (hex number 0008)

SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA (hex number 0014)

SSL_DHE_DSS_EXPORT1024_WITH_DES_CBC_SHA (hex number 0063)

SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA (hex number 0011)

SSL_DH_anon_WITH_AES_128_CBC_SHA (hex number 0034)

SSL_DH_anon_WITH_AES_256_CBC_SHA (hex number 003A)

SSL_DH_anon_WITH_DES_CBC_SHA (hex number 001A

SSL_DH_anon_WITH_3DES_EDE_CBC_SHA (hex number 001B

SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA (hex number 0019)

Cryptographic Module Roles

This module makes use of the functions in the SSLite module. The roles below are as defined in the SSLite Security Policy section 6.1.

Roles

The IBMJSSEFIPS module supports two roles, Crypto Officer role and a user role.

- **ROLE_CO**: The Crypto Officer Role is purely an administrative role and does not involve the use of any of the modules cryptographic services. The role is not explicitly authenticated but assumed implicitly on implementation of the modules installation and usage sections defined in the security rules section.

- **ROLE_USER**: The User Role has access to all of the modules services. The role is not explicitly authenticated but assumed implicitly on access of any of the modules services.

Role	Type of Authentication	Authentication Data
Cryptographic Officer Role	None	None
User Role	None	None

Table 7: Roles and Required Identification and Authentication

Authentication Mechanism	Strength of Mechanism
There are no role or user authentication mechanisms	Not Applicable

Table 8: Strengths of Authentication Mechanisms

Cryptographic Module Key Management

This module makes use of the functions in the SSLite module. The management of cryptographic keys, as defined below, is in the SSLite Security Policy section 6.1.

Key Storage

The IBMJSSEFIPS module does not provide long-term cryptographic key storage. If an application program makes use of an IBMJSSEFIPS service to implement cryptographic key storage functionality, it is a responsibility of the application program developers to ensure FIPS140-2 compliance of key storing techniques they implement.

Key generation

IBMJSSEFIPS provides protocol services for TLSv1.0. This protocol involves the generation of key material based on elements within the handshake protocol. TLSv1.0 depends on SHA-1 for the generation of key material.

Key Protection

The management and allocation of memory is the responsibility of the operating system. Each instance of the cryptographic module is self-contained within a process space. All keys are associated with the User role.

Key Generation

Key generation is handled using the IBM SSLite in Java module that in turn uses the IBM CryptoLite in Java subsystem that uses a FIPS approved RNG algorithm that is based on SHA-1. The RNG has a maximum number of internal states of 2^160, this being limited by the compression function in SHA-1. The RSA key generation algorithms use the RNG engine seeded with 20 bytes of true random data. This true random generator is based on IBM patented technology where statistical analysis used to estimate the entropy of the clock jitter. The internal RNG engine is enhanced using an automatic reseeding policy that insert a random byte every 128 bytes of output if more than 30 seconds passed since last being reseeded.

Key zeroization

Key objects are zeroed and any associated data discarded when the key object is garbage collected through the finalizer method. The IBM SSLite in Java uses the IBM CryptoLite in Java sub-module to provide an additional mechanisms which helps to ensure key zeroization through a dispose method.

Key Import/Export

The IBMJSSEFIPS module provides a series of services for applications to access cryptographic material contained within various long-term storage elements or tokens. These key repositories and tokens are outside of IBMJSSEFIPS's cryptographic boundary. The IBMJSSEFIPS module temporarily holds and uses key material on behalf of the calling applications and processes. Key material imported from long-term storage is stored internally in token key rings. This temporary internal storage of key material and its subsequent use is on behalf of the calling applications.

IBMJSSEFIPS supports the following a token type only.

Java KeyStore is a database of private keys and their associated certificates or certificate chains. The certificate chains are used in authenticating end entity certificates. The Java Cryptography Architecture (JCA) provides extensible architecture to manage keys. This architecture is embodied in java.security as a KeyStore. The Java KeyStore follows the existing JCA architecture, which provides a framework for implementations of a KeyStore.

Cryptographic Module Self Tests

IBMJSSEFIPS relies exclusively on the embedded SSLite module for all FIPS-required self-tests, which in turn, are performed by the FIPS-Approved IBM CryptoLite in Java module embedded within SSLite. See section 5.5 of the SSLite Security Policy for more information.

Cryptographic Module Security Rules

Operating System

The cryptographic module is dependant on the operating system environment being set up in accordance with FIPS 140-2 specifications. This includes that the host operating system be restricted to a single operator mode. An additional requirement for this cryptographic provider is the availability of a valid commercial grade installation of an IBM Java JRE 1.3.1 JVM or higher.

Java object model

The use of Java objects within the cryptographic module. In Java each cryptographic object is unique. Thus when an application generates a cryptographic object for use the object is unique to that instance of the application. In this regard other processes have no access to that object and can therefore not interrupt or gain access to the information or activities contained within that object. In this way the cryptographic module protects the single users control of the cryptographic activities and data.

As the Java architecture creates objects that are unique to the application, this allows for “single” user access to the cryptographic operations and data. It is recommended that an application not create static objects. Static objects are shared in the Java architecture and the creation of a static object would be counter to the unique object method of controlling access and data.

User Guidance Programming practices

This section contains guidance for application programmers to avoid practices that could potentially compromise the secure use of this cryptographic module.

- Statics – To ensure that each cryptographic object is unique and accessible only by the individual user, it is important not to use static objects, as all users of the JVM share these objects.

  As the Java architecture creates objects that are unique to the application, this allows for “single” user access to the cryptographic operations and data. It is recommended that an application not create static objects. Static objects are shared in the Java architecture and the creation of a static object would be counter to the unique object method of controlling access and data.

- The program must insure that only the IBM JSSE interfaces are used that documented in the *Java Secure Socket Extension (JSSE) API User's Guide* and associated *IBMJSSEProvider Class Documentation* based on the level of the JVM that the IBMJSSEFIPS provider is running under.

- The classes of this provider should not be called directly (i.e. the com.ibm.fips.* classes). Only use the standard extension API interfaces documented in the *Java Secure Socket Extension (JSSE) API User's Guide* and associated *IBMJSSEProvider Class Documentation* at the JVM level that the IBMJSSEFIPS provider is running on.

- Any tokens used for storing private cryptographic keys should be password protected. The password should follow generally accepted guidelines for password security. All soft tokens should be configured local to the computer.

- Only the TLS protocol is supported.

- If an application wishes to use the defaults for creating an SSL session, they must set the following in the JVMs **java.security** file:
  ssl.SocketFactory.provider=com.ibm.fips.jsse.JSSESocketFactory
  ssl.ServerSocketFactory.provider=
  com.ibm.fips.jsse.JSSEServerSocketFactory

- The provider name to be used with this JSSE provider is com.ibm.fips.jsse.IBMJSSEFIPSProvider.

References

[1] National Institute of Standards and Technology. May 2001. *Security Requirements for Cryptographic Modules.* Federal Information Processing Standards Publication 140-2.

[2] National Institute of Standards and Technology. November 2001. *AES Key Wrap Specification.* Internet. 22 April 2002. http://csrc.nist.gov/encryption/kms/key-wrap.pdf

[3] IBM CryptoLite in Java module. FIPS 140-2 Validated - Certificate #354.

[4] IBM SSLite in Java module. FIPS 140-2 Validate – Certifcagte #406.

Appendix A: Function List

The Cryptographic module has no direct user interfaces. This module is a JSSE provider and the interface into it is through the JSSE framework APIs only. The program must insure that only the IBM JSSE interfaces are used that are documented in the *Java Secure Socket Extension (JSSE) API User's Guide* and associated *IBMJSSEProvider Class Documentation* based on the level of the JVM that the IBMJSSEFIPS provider is running under.

In a J2SDK version 1.3.1 , the JSSE reference implementation classes and interfaces were provided in the com.ibm.net.ssl package.

Now that JSSE has been integrated into the J2SDK, v 1.4. The classes formerly in com.ibm.net.ssl have been promoted to the javax.net.ssl package and are now a part of the standard JSSE API.

If you wish to use the HTTPS protocol handler, you must set the property *java.protocol.handler.pkgs*. In 1.3.x, the https protocol handler is: **com.ibm.net.ssl.internal.www.protocol** and in 1.4.x, the https protocol handler is: **com.ibm.net.ssl.www.protocol**. For example to set the HTTPS handler for 1.4.x:

```
properties.put("java.protocol.handler.pkgs", "com.ibm.net.ssl.www.protocol");
```

For compatibility purposes the com.ibm.net.ssl classes and interfaces still exist, but have been deprecated. Applications written using them can run in the J2SDK, v 1.4 without being recompiled. This may change in a future release; these classes/interfaces may be removed. Thus, all new applications should be written using the javax classes/interfaces.

For now, applications written using the com.ibm.net.ssl API can utilize *either* JSSE 1.0.2 providers (ones using com.ibm.net.ssl) *or* JSSE providers written for the J2SDK, v 1.4 (ones using the javax API). However, applications written using the JSSE API in the J2SDK, v 1.4 can only utilize JSSE providers written for the J2SDK, v 1.4. This new release contains some new functionality and attempting to access such functionality on a provider that doesn't supply it wouldn't work. IBMJSSE, provided with the J2SDK from IBM, is a provider written using the javax API.

The following list are the Framework API's and are only provided for reference and are not part of the IBMJSSEFIPS module:

A

addHandshakeCompletedListener(HandshakeCompletedListener) - Method in class javax.net.ssl.SSLSocket this registers an event listener to receive notifications that an SSL handshake has completed on this connection.

C

canonizeString(String) - Method in class com.ibm.net.ssl.www.ParseUtil Returns a canonical version of the specified string.

checkClientTrusted(X509Certificate[], String) - Method in interface javax.net.ssl.X509TrustManager
Given the partial or complete certificate chain provided by the peer, build a certificate path to a trusted root and return if it can be validated and is trusted for client SSL authentication based on the authentication type.

checkServerTrusted(X509Certificate[], String) - Method in interface javax.net.ssl.X509TrustManager
Given the partial or complete certificate chain provided by the peer, build a certificate path to a trusted root and return if it can be validated and is trusted for server SSL authentication based on the authentication type.

chooseClientAlias(String[], Principal[], Socket) - Method in interface javax.net.ssl.X509KeyManager
Choose an alias to authenticate the client side of a secure socket given the public key type and the list of certificate issuer authorities recognized by the peer (if any).

chooseClientAlias(String, Principal[]) - Method in interface com.ibm.net.ssl.X509KeyManager
Deprecated. Choose an alias to authenticate the client side of a secure socket given the public key type and the list of certificate issuer authorities recognized by the peer (if any).

chooseServerAlias(String, Principal[]) - Method in interface com.ibm.net.ssl.X509KeyManager
Deprecated. Choose an alias to authenticate the server side of a secure socket given the public key type and the list of certificate issuer authorities recognized by the peer (if any).

chooseServerAlias(String, Principal[], Socket) - Method in interface javax.net.ssl.X509KeyManager
Choose an alias to authenticate the server side of a secure socket given the public key type and the list of certificate issuer authorities recognized by the peer (if any).

createServerSocket() - Method in class javax.net.ServerSocketFactory
Returns an unbound server socket.

createServerSocket(int) - Method in class javax.net.ServerSocketFactory
Returns a server socket bound to the specified port.

createServerSocket(int, int) - Method in class javax.net.ServerSocketFactory
Returns a server socket bound to the specified port, and uses the specified connection backlog.

createServerSocket(int, int, InetAddress) - Method in class javax.net.ServerSocketFactory
Returns a server socket bound to the specified port, with a specified listen backlog and local IP.

createSocket() - Method in class javax.net.SocketFactory
Creates an unconnected socket.

createSocket(InetAddress, int) - Method in class javax.net.SocketFactory
Creates a socket and connects it to the specified port number at the specified address.

createSocket(InetAddress, int, InetAddress, int) - Method in class javax.net.SocketFactory
Creates a socket and connect it to the specified remote address on the specified remote port.

createSocket(Socket, String, int, boolean) - Method in class javax.net.ssl.SSLSocketFactory
Returns a socket layered over an existing socket connected to the named host, at the given port.

createSocket(String, int) - Method in class javax.net.SocketFactory
Creates a socket and connects it to the specified remote host at the specified remote port.

createSocket(String, int, InetAddress, int) - Method in class javax.net.SocketFactory
Creates a socket and connects it to the specified remote host on the specified remote port.

D

decode(String) - Static method in class com.ibm.net.ssl.www.ParseUtil

Returns a new String constructed from the specified String by replacing the URL escape sequences and UTF8 encoding with the characters they represent.

E

encodePath(String) - Static method in class com.ibm.net.ssl.www.ParseUtil
Constructs an encoded version of the specified path string suitable for use in the construction of a URL.

G

getAcceptedIssuers() - Method in interface com.ibm.net.ssl.X509TrustManager
Deprecated. Return an array of certificate authority certificates which are trusted for authenticating peers.

getAcceptedIssuers() - Method in interface javax.net.ssl.X509TrustManager
Return an array of certificate authority certificates which are trusted for authenticating peers.

getAlgorithm() - Method in class com.ibm.net.ssl.TrustManagerFactory
Deprecated. Returns the algorithm name of this TrustManagerFactory object.

getAlgorithm() - Method in class com.ibm.net.ssl.KeyManagerFactory
Deprecated. Returns the algorithm name of this KeyManagerFactory object.

getAlgorithm() - Method in class javax.net.ssl.TrustManagerFactory
Returns the algorithm name of this TrustManagerFactory object.

getAlgorithm() - Method in class javax.net.ssl.KeyManagerFactory
Returns the algorithm name of this KeyManagerFactory object.

getCertificateChain(String) - Method in interface com.ibm.net.ssl.X509KeyManager
Deprecated. Returns the certificate chain associated with the given alias.

getCertificateChain(String) - Method in interface javax.net.ssl.X509KeyManager
Returns the certificate chain associated with the given alias.

getCipherSuite() - Method in interface javax.net.ssl.SSLSession
Returns the name of the SSL cipher suite which is used for all connections in the session.

getCipherSuite() - Method in class javax.net.ssl.HttpsURLConnection
Returns the cipher suite in use on this connection.

getCipherSuite() - Method in class javax.net.ssl.HandshakeCompletedEvent
Returns the cipher suite in use by the session which was produced by the handshake.

getClientAliases(String, Principal[]) - Method in interface com.ibm.net.ssl.X509KeyManager
Deprecated. Get the matching aliases for authenticating the client side of a secure socket given the public key type and the list of certificate issuer authorities recognized by the peer (if any).

getClientAliases(String, Principal[]) - Method in interface javax.net.ssl.X509KeyManager
Get the matching aliases for authenticating the client side of a secure socket given the public key type and the list of certificate issuer authorities recognized by the peer (if any).

getClientSessionContext() - Method in class javax.net.ssl.SSLContext
Returns the client session context, which represents the set of SSL sessions available for use during the handshake phase of client-side SSL sockets.

getCreationTime() - Method in interface javax.net.ssl.SSLSession
Returns the time at which this Session representation was created, in milliseconds since midnight, January 1, 1970 UTC.

getDefault() - Static method in class javax.net.SocketFactory
Returns a copy of the environment's default socket factory.

getDefault() - Static method in class javax.net.ServerSocketFactory
Returns a copy of the environment's default socket factory.

getDefault() - Static method in class javax.net.ssl.SSLSocketFactory
Returns the default SSL socket factory.

getDefault() - Static method in class javax.net.ssl.SSLServerSocketFactory
Returns the default SSL server socket factory.

getDefaultAlgorithm() - Static method in class com.ibm.net.ssl.TrustManagerFactory
Deprecated. The default TrustManager can be changed by setting the value of the "javax.net.ssl.TrustManager.type" security property (in the Java security properties file) to the desired name.

getDefaultAlgorithm() - Static method in class com.ibm.net.ssl.KeyManagerFactory
Deprecated. The default KeyManager can be changed by setting the value of the "javax.net.ssl.KeyManager.type" security property (in the Java security properties file) to the desired name.

getDefaultAlgorithm() - Static method in class javax.net.ssl.TrustManagerFactory
Obtains the default TrustManagerFactory algorithm name.

getDefaultAlgorithm() - Static method in class javax.net.ssl.KeyManagerFactory
The default KeyManager can be changed by setting the value of the "ssl.KeyManagerFactory.algorithm" security property (in the Java security properties file) to the desired name.

getDefaultCipherSuites() - Method in class javax.net.ssl.SSLSocketFactory
Returns the list of cipher suites which are enabled by default.

getDefaultCipherSuites() - Method in class javax.net.ssl.SSLServerSocketFactory
Returns the list of cipher suites which are enabled by default.

getDefaultHostnameVerifier() - Static method in class javax.net.ssl.HttpsURLConnection
Gets the default HostnameVerifier that it inherited when an instance of this class is created.

getDefaultSSLSocketFactory() - Static method in class com.ibm.net.ssl.HttpsURLConnection
Deprecated. Gets the default SSL socket factory.

getDefaultSSLSocketFactory() - Static method in class javax.net.ssl.HttpsURLConnection
Gets the default static SSL socket factory used when creating sockets for secure https URL connections.

getEnabledCipherSuites() - Method in class javax.net.ssl.SSLSocket
Returns the names of the SSL cipher suites which are currently enabled for use on this connection.

getEnabledCipherSuites() - Method in class javax.net.ssl.SSLServerSocket
Returns the list of cipher suites which are currently enabled for use by newly accepted connections.

getEnabledProtocols() - Method in class javax.net.ssl.SSLSocket

Returns the names of the protocol versions which are currently enabled for use on this connection.

getEnabledProtocols() - Method in class javax.net.ssl.SSLServerSocket
Returns the names of the protocols which are currently enabled for use by the newly accepted connections.

getEnableSessionCreation() - Method in class javax.net.ssl.SSLSocket
Returns true if new SSL sessions may be established by this socket.

getEnableSessionCreation() - Method in class javax.net.ssl.SSLServerSocket
Returns true if new SSL sessions may be established by the sockets which are created from this server socket.

getHostnameVerifier() - Method in class javax.net.ssl.HttpsURLConnection
Gets the HostnameVerifier in place on this instance.

getId() - Method in interface javax.net.ssl.SSLSession
Returns the identifier assigned to this Session.

getIds() - Method in interface javax.net.ssl.SSLSessionContext
Returns an Enumeration of all session id's grouped under this SSLSessionContext.

getInstance(String) - Static method in class com.ibm.net.ssl.TrustManagerFactory
Deprecated. Generates a TrustManagerFactory object that implements the specified trust management algorithm.

getInstance(String) - Static method in class com.ibm.net.ssl.SSLContext
Deprecated. Generates a SSLContext object that implements the specified secure socket protocol.

getInstance(String) - Static method in class com.ibm.net.ssl.KeyManagerFactory
Deprecated. Generates a KeyManagerFactory object that implements the specified management algorithm.

getInstance(String) - Static method in class javax.net.ssl.TrustManagerFactory
Generates a TrustManagerFactory object that implements the specified trust management algorithm.

getInstance(String) - Static method in class javax.net.ssl.SSLContext
Generates a SSLContext object that implements the specified secure socket protocol.

getInstance(String) - Static method in class javax.net.ssl.KeyManagerFactory
Generates a KeyManagerFactory object that implements the specified key management algorithm.

getInstance(String, Provider) - Static method in class javax.net.ssl.TrustManagerFactory
Generates a TrustManagerFactory object for the specified trust management algorithm from the specified provider.

getInstance(String, Provider) - Static method in class javax.net.ssl.SSLContext
Generates a SSLContext object that implements the specified secure socket protocol from the specified provider.

getInstance(String, Provider) - Static method in class javax.net.ssl.KeyManagerFactory
Generates a KeyManagerFactory object for the specified key management algorithm from the specified provider.

getInstance(String, String) - Static method in class com.ibm.net.ssl.TrustManagerFactory
Deprecated. Generates a TrustManagerFactory object for the specified trust management algorithm from the specified provider.

getInstance(String, String) - Static method in class com.ibm.net.ssl.SSLContext
Deprecated. Generates a SSLContext object that implements the specified secure socket protocol.

getInstance(String, String) - Static method in class com.ibm.net.ssl.KeyManagerFactory
Deprecated. Generates a KeyManagerFactory object for the specified key management algorithm from the specified provider.

getInstance(String, String) - Static method in class javax.net.ssl.TrustManagerFactory
Generates a TrustManagerFactory object for the specified trust management algorithm from the specified provider.

getInstance(String, String) - Static method in class javax.net.ssl.SSLContext
Generates a SSLContext object that implements the specified secure socket protocol from the specified provider.

getInstance(String, String) - Static method in class javax.net.ssl.KeyManagerFactory
Generates a KeyManagerFactory object for the specified key management algorithm from the specified provider.

getKeyManagers() - Method in class com.ibm.net.ssl.KeyManagerFactory
Deprecated. Returns one key manager for each type of key material.

getKeyManagers() - Method in class javax.net.ssl.KeyManagerFactory
Returns one key manager for each type of key material.

getLastAccessedTime() - Method in interface javax.net.ssl.SSLSession
Returns the last time this Session representation was accessed by the session level infrastructure, in milliseconds since midnight, January 1, 1970 UTC.

getLocalCertificates() - Method in interface javax.net.ssl.SSLSession
Returns the certificate(s) that were sent to the peer during handshaking.

getLocalCertificates() - Method in class javax.net.ssl.HttpsURLConnection
Returns the certificate(s) that were sent to the server during handshaking.

getLocalCertificates() - Method in class javax.net.ssl.HandshakeCompletedEvent
Returns the certificate(s) that were sent to the peer during handshaking.

getName() - Method in class javax.net.ssl.SSLSessionBindingEvent
Returns the name to which the object is being bound, or the name from which the object is being unbound.

getNeedClientAuth() - Method in class javax.net.ssl.SSLSocket
Returns true if the socket will require client authentication.

getNeedClientAuth() - Method in class javax.net.ssl.SSLServerSocket
Returns true if client authentication is required on newly accepted connections.

getPeerCertificateChain() - Method in interface javax.net.ssl.SSLSession
Returns the identity of the peer which was identified as part of defining the session.

getPeerCertificateChain() - Method in class javax.net.ssl.HandshakeCompletedEvent
Returns the identity of the peer which was identified as part of defining the session.

getPeerCertificates() - Method in interface javax.net.ssl.SSLSession
Returns the identity of the peer which was established as part of defining the session.

getPeerCertificates() - Method in class javax.net.ssl.HandshakeCompletedEvent
Returns the identity of the peer which was established as part of defining the session.

getPeerHost() - Method in interface javax.net.ssl.SSLSession
Returns the host name of the peer in this session.

getPrivateKey(String) - Method in interface com.ibm.net.ssl.X509KeyManager
Deprecated. Returns the key associated with the given alias.

getPrivateKey(String) - Method in interface javax.net.ssl.X509KeyManager
Returns the key associated with the given alias.

getProtocol() - Method in class com.ibm.net.ssl.SSLContext
Deprecated. Returns the protocol name of this SSLContext object.

getProtocol() - Method in interface javax.net.ssl.SSLSession
Returns the standard name of the protocol used for all connections in the session.

getProtocol() - Method in class javax.net.ssl.SSLContext
Returns the protocol name of this SSLContext object.

getProvider() - Method in class com.ibm.net.ssl.TrustManagerFactory
Deprecated. Returns the provider of this TrustManagerFactory object.

getProvider() - Method in class com.ibm.net.ssl.SSLContext
Deprecated. Returns the provider of this SSLContext object.

getProvider() - Method in class com.ibm.net.ssl.KeyManagerFactory
Deprecated. Returns the provider of this KeyManagerFactory object.

getProvider() - Method in class javax.net.ssl.TrustManagerFactory
Returns the provider of this TrustManagerFactory object.

getProvider() - Method in class javax.net.ssl.SSLContext
Returns the provider of this SSLContext object.

getProvider() - Method in class javax.net.ssl.KeyManagerFactory
Returns the provider of this KeyManagerFactory object.

getServerAliases(String, Principal[]) - Method in interface com.ibm.net.ssl.X509KeyManager

Deprecated. Get the matching aliases for authenticating the server side of a secure socket given the public key type and the list of certificate issuer authorities recognized by the peer (if any).

getServerAliases(String, Principal[]) - Method in interface javax.net.ssl.X509KeyManager
Get the matching aliases for authenticating the server side of a secure socket given the public key type and the list of certificate issuer authorities recognized by the peer (if any).

getServerCertificateChain() - Method in class com.ibm.net.ssl.HttpsURLConnection
Deprecated. Returns the server's X.509 certificate chain, or null if the server did not authenticate.

getServerCertificates() - Method in class javax.net.ssl.HttpsURLConnection
Returns the server's certificate chain which was established as part of defining the session.

getServerSessionContext() - Method in class javax.net.ssl.SSLContext
Returns the server session context, which represents the set of SSL sessions available for use during the handshake phase of server-side SSL sockets.

getServerSocketFactory() - Method in class com.ibm.net.ssl.SSLContext
Deprecated. Returns a ServerSocketFactory object for this context.

getServerSocketFactory() - Method in class javax.net.ssl.SSLContext
Returns a ServerSocketFactory object for this context.

getSession() - Method in class javax.net.ssl.SSLSocket
Returns the SSL Session in use by this connection.

getSession() - Method in class javax.net.ssl.SSLSessionBindingEvent
Returns the SSLSession into which the listener is being bound or from which the listener is being unbound.

getSession() - Method in class javax.net.ssl.HandshakeCompletedEvent
Returns the session that triggered this event.

getSession(byte[]) - Method in interface javax.net.ssl.SSLSessionContext
Returns the SSLSession bound to the specified session id.

getSessionCacheSize() - Method in interface javax.net.ssl.SSLSessionContext
Returns the size of the cache used for storing SSLSession objects grouped under this SSLSessionContext.

getSessionContext() - Method in interface javax.net.ssl.SSLSession
Returns the context in which this session is bound.

getSessionTimeout() - Method in interface javax.net.ssl.SSLSessionContext
Returns the timeout limit of SSLSession objects grouped under this SSLSessionContext.

getSocket() - Method in class javax.net.ssl.HandshakeCompletedEvent
Returns the socket which is the source of this event.

getSocketFactory() - Method in class com.ibm.net.ssl.SSLContext
Deprecated. Returns a SocketFactory object for this context.

getSocketFactory() - Method in class javax.net.ssl.SSLContext
Returns a SocketFactory object for this context.

getSSLSocketFactory() - Method in class com.ibm.net.ssl.HttpsURLConnection
Deprecated. Gets the SSL socket factory.

getSSLSocketFactory() - Method in class javax.net.ssl.HttpsURLConnection
Gets the SSL socket factory to be used when creating sockets for secure https URL connections.

getSupportedCipherSuites() - Method in class javax.net.ssl.SSLSocketFactory
Returns the names of the cipher suites which could be enabled for use on an SSL connection.

getSupportedCipherSuites() - Method in class javax.net.ssl.SSLSocket
Returns the names of the cipher suites which could be enabled for use on this connection.

getSupportedCipherSuites() - Method in class javax.net.ssl.SSLServerSocketFactory
Returns the names of the cipher suites which could be enabled for use on an SSL connection created by this factory.

getSupportedCipherSuites() - Method in class javax.net.ssl.SSLServerSocket
Returns the names of the cipher suites which could be enabled for use on an SSL connection.

getSupportedProtocols() - Method in class javax.net.ssl.SSLSocket
Returns the names of the protocols which could be enabled for use on an SSL connection.

getSupportedProtocols() - Method in class javax.net.ssl.SSLServerSocket
Returns the names of the protocols which could be enabled for use.

H

I

init(KeyStore) - Method in class com.ibm.net.ssl.TrustManagerFactory
Deprecated. Initializes this factory with a source of certificate authorities and related trust material.

init(KeyStore) - Method in class javax.net.ssl.TrustManagerFactory
Initializes this factory with a source of certificate authorities and related trust material.

init(KeyStore, char[]) - Method in class com.ibm.net.ssl.KeyManagerFactory
Deprecated. Initializes this factory with a source of key material.

init(KeyStore, char[]) - Method in class javax.net.ssl.KeyManagerFactory
Initializes this factory with a source of key material.

init(ManagerFactoryParameters) - Method in class javax.net.ssl.TrustManagerFactory
Initializes this factory with a source of provider-specific trust material.

init(ManagerFactoryParameters) - Method in class javax.net.ssl.KeyManagerFactory
Initializes this factory with a source of provider-specific key material.

init(String, String, String) - Method in class com.ibm.net.ssl.SSLContext
Deprecated. Initializes this context.

invalidate() - Method in interface javax.net.ssl.SSLSession
Invalidates the session.

isClientTrusted(X509Certificate[]) - Method in interface com.ibm.net.ssl.X509TrustManager
Deprecated. Given the partial or complete certificate chain provided by the peer, build a certificate path to a trusted root and return true if it can be validated and is trusted for client SSL authentication.

isServerTrusted(X509Certificate[]) - Method in interface com.ibm.net.ssl.X509TrustManager
Deprecated. Given the partial or complete certificate chain provided by the peer, build a certificate path to a trusted root and return true if it can be validated and is trusted for server SSL authentication.

J

javax.net - package javax.net

javax.net.ssl - package javax.net.ssl

K

KeyManager - interface com.ibm.net.ssl.KeyManager.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by KeyManager. of a secure socket.*

KeyManager - interface javax.net.ssl.KeyManager.
This is the base interface for JSSE key managers.

KeyManagerFactory - class com.ibm.net.ssl.KeyManagerFactory.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by KeyManagerFactory.*

KeyManagerFactory - class javax.net.ssl.KeyManagerFactory.
This class acts as a factory for key managers based on a source of key material.

KeyManagerFactorySpi - class com.ibm.net.ssl.KeyManagerFactorySpi.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by KeyManagerFactorySpi.*

KeyManagerFactorySpi - class javax.net.ssl.KeyManagerFactorySpi.
This class defines the *Service Provider Interface* (**SPI**) for the KeyManagerFactory class.

KeyManagerFactorySpi() - Constructor for class com.ibm.net.ssl.KeyManagerFactorySpi
Deprecated.

KeyManagerFactorySpi() - Constructor for class javax.net.ssl.KeyManagerFactorySpi

---

L

M

ManagerFactoryParameters - interface javax.net.ssl.ManagerFactoryParameters.

This class is the base interface for providing algorithm-specific information to a KeyManagerFactory or TrustManagerFactory.

P

putValue(String, Object) - Method in interface javax.net.ssl.SSLSession
Binds the specified value object into the session's application layer data with the given name.

R

removeHandshakeCompletedListener(HandshakeCompletedListener) - Method in class javax.net.ssl.SSLSocket
Removes a previously registered handshake completion listener.

removeValue(String) - Method in interface javax.net.ssl.SSLSession
Removes the object bound to the given name in the session's application layer data.

S

ServerSocketFactory - class javax.net.ServerSocketFactory.
This class creates server sockets.

setEnabledCipherSuites(String[]) - Method in class javax.net.ssl.SSLSocket
Controls which particular cipher suites are enabled for use on this connection.

setEnabledCipherSuites(String[]) - Method in class javax.net.ssl.SSLServerSocket
Controls which particular SSL cipher suites are enabled for use by accepted connections.

setEnabledProtocols(String[]) - Method in class javax.net.ssl.SSLSocket
Controls which particular protocol versions are enabled for use on this connection.

setEnabledProtocols(String[]) - Method in class javax.net.ssl.SSLServerSocket
Controls which particular protocols are enabled for use by accepted connections.

setEnableSessionCreation(boolean) - Method in class javax.net.ssl.SSLSocket
Controls whether new SSL sessions may be established by this socket.

setEnableSessionCreation(boolean) - Method in class javax.net.ssl.SSLServerSocket
Controls whether new SSL sessions may be established by the sockets which are created from this server socket.

setHostnameVerifier(HostnameVerifier) - Method in class javax.net.ssl.HttpsURLConnection
Sets the HostnameVerifier.

setNeedClientAuth(boolean) - Method in class javax.net.ssl.SSLSocket
Configures the socket to require client authentication.

setNeedClientAuth(boolean) - Method in class javax.net.ssl.SSLServerSocket
Controls whether the connections which are accepted must include successful client authentication.

setSessionCacheSize(int) - Method in interface javax.net.ssl.SSLSessionContext
Sets the size of the cache used for storing SSLSession objects grouped under this SSLSessionContext.

setSessionTimeout(int) - Method in interface javax.net.ssl.SSLSessionContext
Sets the timeout limit for SSLSession objects grouped under this SSLSessionContext.

setSSLSocketFactory(SSLSocketFactory) - Method in class javax.net.ssl.HttpsURLConnection
Sets the SSL socket factory to be used when creating sockets for secure https URL connections.

setUseClientMode(boolean) - Method in class javax.net.ssl.SSLSocket
Configures the socket to use client (or server) mode in its first handshake.

setUseClientMode(boolean) - Method in class javax.net.ssl.SSLServerSocket
Controls whether accepted connections are in the (default) SSL server mode, or the SSL client mode.

setWantClientAuth(boolean) - Method in class javax.net.ssl.SSLSocket
Configures the socket to request client authentication, but only if such a request is appropriate to the cipher suite negotiated.

setWantClientAuth(boolean) - Method in class javax.net.ssl.SSLServerSocket
Controls whether the connections which are accepted should request client authentication as part of the SSL negotiations.

SocketFactory - class javax.net.SocketFactory.
This class creates sockets.

SSLContext - class com.ibm.net.ssl.SSLContext.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by SSLContext.*

SSLContext - class javax.net.ssl.SSLContext.
Instances of this class represent a secure socket protocol implementation which acts as a factory for secure socket factories.

SSLContextSpi - class com.ibm.net.ssl.SSLContextSpi.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by SSLContextSpi.*

SSLContextSpi - class javax.net.ssl.SSLContextSpi.
This class defines the *Service Provider Interface* (**SPI**) for the SSLContext class.

SSLContextSpi() - Constructor for class com.ibm.net.ssl.SSLContextSpi
Deprecated.

SSLContextSpi() - Constructor for class javax.net.ssl.SSLContextSpi

SSLException - exception javax.net.ssl.SSLException.
Indicates some kind of error detected by an SSL subsystem.

SSLException(String) - Constructor for class javax.net.ssl.SSLException
Constructs an exception reporting an error found by an SSL subsystem.

SSLHandshakeException - exception javax.net.ssl.SSLHandshakeException.
Indicates that the client and server could not negotiate the desired level of security.

SSLHandshakeException(String) - Constructor for class javax.net.ssl.SSLHandshakeException
Constructs an exception reporting an error found by an SSL subsystem during handshaking.

SSLKeyException - exception javax.net.ssl.SSLKeyException.
Reports a bad SSL key.

SSLKeyException(String) - Constructor for class javax.net.ssl.SSLKeyException
Constructs an exception reporting a key management error found by an SSL subsystem.

SSLPeerUnverifiedException - exception javax.net.ssl.SSLPeerUnverifiedException.
Indicates that the peer's identity has not been verified.

SSLPeerUnverifiedException(String) - Constructor for class javax.net.ssl.SSLPeerUnverifiedException
Constructs an exception reporting that the SSL peer's identity has not been verifiied.

SSLPermission - class javax.net.ssl.SSLPermission.
This class is for various network permissions.

SSLPermission(String) - Constructor for class javax.net.ssl.SSLPermission
Creates a new SSLPermission with the specified name.

SSLPermission(String, String) - Constructor for class javax.net.ssl.SSLPermission
Creates a new SSLPermission object with the specified name.

SSLProtocolException - exception javax.net.ssl.SSLProtocolException.
Reports an error in the operation of the SSL protocol.

SSLProtocolException(String) - Constructor for class javax.net.ssl.SSLProtocolException
Constructs an exception reporting an SSL protocol error detected by an SSL subsystem.

SSLServerSocket - class javax.net.ssl.SSLServerSocket.
This class extends ServerSockets and provides secure server sockets using protocols such as the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols.

SSLServerSocketFactory - class javax.net.ssl.SSLServerSocketFactory.
SSLServerSocketFactorys create SSLServerSockets.

SSLSession - interface javax.net.ssl.SSLSession.
In SSL, sessions are used to describe an ongoing relationship between two entities.

SSLSessionBindingEvent - class javax.net.ssl.SSLSessionBindingEvent.
This event is propagated to a SSLSessionBindingListener.

SSLSessionBindingEvent(SSLSession, String) - Constructor for class javax.net.ssl.SSLSessionBindingEvent
Constructs a new SSLSessionBindingEvent.

SSLSessionBindingListener - interface javax.net.ssl.SSLSessionBindingListener.
This interface is implemented by objects which want to know when they are being bound or unbound from a SSLSession.

SSLSessionContext - interface javax.net.ssl.SSLSessionContext.
A SSLSessionContext represents a set of SSLSessions associated with a single entity.

SSLSocket - class javax.net.ssl.SSLSocket.
This class extends Sockets and provides secure socket using protocols such as the "Secure Sockets Layer" (SSL) or IETF "Transport Layer Security" (TLS) protocols.

SSLSocketFactory - class javax.net.ssl.SSLSocketFactory.
SSLSocketFactorys create SSLSockets.

SSLSocketFactory() - Constructor for class javax.net.ssl.SSLSocketFactory

startHandshake() - Method in class javax.net.ssl.SSLSocket
Starts an SSL handshake on this connection.

T

TrustManager - interface com.ibm.net.ssl.TrustManager.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by TrustManager.*

TrustManager - interface javax.net.ssl.TrustManager.
This is the base interface for JSSE trust managers.

TrustManagerFactory - class com.ibm.net.ssl.TrustManagerFactory.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by TrustManagerFactory.*

TrustManagerFactory - class javax.net.ssl.TrustManagerFactory.
This class acts as a factory for trust managers based on a source of trust material.

TrustManagerFactorySpi - class com.ibm.net.ssl.TrustManagerFactorySpi.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by TrustManagerFactorySpi.*

TrustManagerFactorySpi - class javax.net.ssl.TrustManagerFactorySpi.
This class defines the *Service Provider Interface* (**SPI**) for the TrustManagerFactory class.

TrustManagerFactorySpi() - Constructor for class com.ibm.net.ssl.TrustManagerFactorySpi
Deprecated.

TrustManagerFactorySpi() - Constructor for class javax.net.ssl.TrustManagerFactorySpi

---

V

valueBound(SSLSessionBindingEvent) - Method in interface javax.net.ssl.SSLSessionBindingListener
This is called to notify the listener that it is being bound into an SSLSession.

valueUnbound(SSLSessionBindingEvent) - Method in interface javax.net.ssl.SSLSessionBindingListener
This is called to notify the listener that it is being unbound from a SSLSession.

verify(String, SSLSession) - Method in interface javax.net.ssl.HostnameVerifier
Verify that the host name is an acceptable match with the server's authentication scheme.

X

X509KeyManager - interface com.ibm.net.ssl.X509KeyManager.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by X509KeyManager.*

X509KeyManager - interface javax.net.ssl.X509KeyManager.
Instances of this interface manage which X509 certificate-based key pairs are used to authenticate the local side of a secure socket.

X509TrustManager - interface com.ibm.net.ssl.X509TrustManager.
Deprecated. *As of JDK 1.4, this implementation-specific class was replaced by X509TrustManager.*

X509TrustManager - interface javax.net.ssl.X509TrustManager.
Instance of this interface manage which X509 certificates may be used to authenticate the remote side of a secure socket.

X509V1CertImpl - class com.ibm.security.cert.X509V1CertImpl.
The X509V1CertImpl class is used as a conversion wrapper around sun.security.x509.X509Cert certificates when running under JDK1.1.x.

X509V1CertImpl() - Constructor for class com.ibm.security.cert.X509V1CertImpl
Default constructor.

X509V1CertImpl(byte[]) - Constructor for class com.ibm.security.cert.X509V1CertImpl
Unmarshals a certificate from its encoded form, parsing the encoded bytes.

X509V1CertImpl(InputStream) - Constructor for class com.ibm.security.cert.X509V1CertImpl unmarshals an X.509 certificate from an input stream.

Notices

Java is a registered trademark of SUN. Inc.

HP-UX is a registered trademark Hewlet Packard, Inc.

AIX, Everyplace, z/OS, AS/400 and IBM are trademarks or registered trademarks of IBM Corporation in the United States, other countries, or both.

Pentium and X-Scale are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Red Hat is a trademark of Red Hat, Inc.

SuSE is a registered trademark of SuSE AG

Other company, product, and service names may be trademarks or service marks of others.