IBM Security Access Manager for Web
Version 7.0

Authorization Java Classes Developer Reference

SC23-6516-02

IBM Security Access Manager for Web
Version 7.0

Authorization Java Classes Developer Reference

SC23-6516-02

Note

Before using this information and the product it supports, read the information in “Notices” on page 37.

Edition notice

Note: This edition applies to version 7, release 0, modification 0 of IBM Security Access Manager (product number 5724-C87) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this publication

IBM Security Access Manager for Web, formerly called IBM Tivoli Access Manager for e-business, is a user authentication, authorization, and web single sign-on solution for enforcing security policies over a wide range of web and application resources.

This reference contains information about how to use Security Access Manager authorization Java™ classes and methods to enable an application to programmatically perform Security Access Manager authorization tasks. This document describes the Java implementation of the Security Access Manager authorization API. See the *IBM Security Access Manager for Web: Authorization Java Classes Developer Reference* for information regarding the Java implementation of these APIs.

Information about the `pdadmin` command-line interface (CLI) can be found in the *IBM Security Access Manager for Web: Command Reference*.

IBM Security Access Manager for Web Authorization Java Classes Developer Reference Guide explains how to configure and use the Security Access Manager Authorization Java Classes API (application programming interface).

Intended audience

This reference is for application programmers writing programs in and Java programming language to authorize the users and objects associated with the Security Access Manager product.

Readers must be familiar with:
- Microsoft Windows and UNIX operating systems
- Database architecture and concepts
- Security management
- Internet protocols, including HTTP, TCP/IP, File Transfer Protocol (FTP), and Telnet
- The user registry that Security Access Manager is configured to use
- Lightweight Directory Access Protocol (LDAP) and directory services, if used by your user registry
- Authentication and authorization

To enable Secure Sockets Layer (SSL) communication, you must be familiar with SSL protocol, key exchange (public and private), digital signatures, cryptographic algorithms, and certificate authorities.

This guide is for developers and system administrators working with the Security Access Manager Authorization Java Classes API.

Readers should be familiar with the following:
- Supported operating systems
- Database architecture and concepts

- Security management
- Internet protocols, including HTTP, TCP/IP, File Transfer Protocol (FTP), and Telnet
- Lightweight Directory Access Protocol (LDAP) and directory services
- Authentication and authorization

If you are enabling Secure Sockets Layer (SSL) communication, you also should be familiar with SSL protocol, key exchange (public and private), digital signatures, cryptographic algorithms, and certificate authorities.

Access to publications and terminology

This section provides:
- A list of publications in the "IBM Security Access Manager for Web library."
- Links to "Online publications" on page xii.
- A link to the "IBM Terminology website" on page xii.

IBM Security Access Manager for Web library

The following documents are in the IBM Security Access Manager for Web library:
- *IBM Security Access Manager for Web Quick Start Guide*, GI11-9333-01

  Provides steps that summarize major installation and configuration tasks.
- *IBM Security Web Gateway Appliance Quick Start Guide* – Hardware Offering

  Guides users through the process of connecting and completing the initial configuration of the WebSEAL Hardware Appliance, SC22-5434-00
- *IBM Security Web Gateway Appliance Quick Start Guide* – Virtual Offering

  Guides users through the process of connecting and completing the initial configuration of the WebSEAL Virtual Appliance.
- *IBM Security Access Manager for Web Installation Guide*, GC23-6502-02

  Explains how to install and configure Security Access Manager.
- *IBM Security Access Manager for Web Upgrade Guide*, SC23-6503-02

  Provides information for users to upgrade from version 6.0, or 6.1.x to version 7.0.
- *IBM Security Access Manager for Web Administration Guide*, SC23-6504-03

  Describes the concepts and procedures for using Security Access Manager. Provides instructions for performing tasks from the Web Portal Manager interface and by using the **pdadmin** utility.
- *IBM Security Access Manager for Web WebSEAL Administration Guide*, SC23-6505-03

  Provides background material, administrative procedures, and reference information for using WebSEAL to manage the resources of your secure Web domain.
- *IBM Security Access Manager for Web Plug-in for Web Servers Administration Guide*, SC23-6507-02

  Provides procedures and reference information for securing your Web domain by using a Web server plug-in.
- *IBM Security Access Manager for Web Shared Session Management Administration Guide*, SC23-6509-02

  Provides administrative considerations and operational instructions for the session management server.

- *IBM Security Access Manager for Web Shared Session Management Deployment Guide*, SC22-5431-00

  Provides deployment considerations for the session management server.
- *IBM Security Web Gateway Appliance Administration Guide*, SC22-5432-01

  Provides administrative procedures and technical reference information for the WebSEAL Appliance.
- *IBM Security Web Gateway Appliance Configuration Guide for Web Reverse Proxy*, SC22-5433-01

  Provides configuration procedures and technical reference information for the WebSEAL Appliance.
- *IBM Security Web Gateway Appliance Web Reverse Proxy Stanza Reference*, SC27-4442-01

  Provides a complete stanza reference for the IBM® Security Web Gateway Appliance Web Reverse Proxy.
- *IBM Security Access Manager for Web WebSEAL Configuration Stanza Reference*, SC27-4443-01

  Provides a complete stanza reference for WebSEAL.
- *IBM Global Security Kit: CapiCmd Users Guide*, SC22-5459-00

  Provides instructions on creating key databases, public-private key pairs, and certificate requests.
- *IBM Security Access Manager for Web Auditing Guide*, SC23-6511-03

  Provides information about configuring and managing audit events by using the native Security Access Manager approach and the Common Auditing and Reporting Service. You can also find information about installing and configuring the Common Auditing and Reporting Service. Use this service for generating and viewing operational reports.
- *IBM Security Access Manager for Web Command Reference*, SC23-6512-03

  Provides reference information about the commands, utilities, and scripts that are provided with Security Access Manager.
- *IBM Security Access Manager for Web Administration C API Developer Reference*, SC23-6513-02

  Provides reference information about using the C language implementation of the administration API to enable an application to perform Security Access Manager administration tasks.
- *IBM Security Access Manager for Web Administration Java Classes Developer Reference*, SC23-6514-02

  Provides reference information about using the Java language implementation of the administration API to enable an application to perform Security Access Manager administration tasks.
- *IBM Security Access Manager for Web Authorization C API Developer Reference*, SC23-6515-02

  Provides reference information about using the C language implementation of the authorization API to enable an application to use Security Access Manager security.
- *IBM Security Access Manager for Web Authorization Java Classes Developer Reference*, SC23-6516-02

  Provides reference information about using the Java language implementation of the authorization API to enable an application to use Security Access Manager security.

- *IBM Security Access Manager for Web Web Security Developer Reference*, SC23-6517-02

  Provides programming and reference information for developing authentication modules.
- *IBM Security Access Manager for Web Error Message Reference*, GI11-8157-02

  Provides explanations and corrective actions for the messages and return code.
- *IBM Security Access Manager for Web Troubleshooting Guide*, GC27-2717-01

  Provides problem determination information.
- *IBM Security Access Manager for Web Performance Tuning Guide*, SC23-6518-02

  Provides performance tuning information for an environment that consists of Security Access Manager with the IBM Tivoli Directory Server as the user registry.

Online publications

IBM posts product publications when the product is released and when the publications are updated at the following locations:

IBM Security Access Manager for Web Information Center
The http://pic.dhe.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.isam.doc_70/welcome.html site displays the information center welcome page for this product.

IBM Security Systems Documentation Central and Welcome page
IBM Security Systems Documentation Central provides an alphabetical list of all IBM Security Systems product documentation and links to the product information center for specific versions of each product.

Welcome to IBM Security Systems Information Centers provides and introduction to, links to, and general information about IBM Security Systems information centers.

IBM Publications Center
The http://www-05.ibm.com/e-business/linkweb/publications/servlet/pbi.wss site offers customized search functions to help you find all the IBM publications that you need.

IBM Terminology website

The IBM Terminology website consolidates terminology for product libraries in one location. You can access the Terminology website at http://www.ibm.com/software/globalization/terminology.

Related publications

This section lists the IBM products that are related to and included with the Security Access Manager solution.

Note: The following middleware products are not packaged with IBM Security Web Gateway Appliance.

IBM Global Security Kit

Security Access Manager provides data encryption by using Global Security Kit (GSKit) version 8.0.x. GSKit is included on the *IBM Security Access Manager for Web Version 7.0* product image or DVD for your particular platform.

GSKit version 8 includes the command-line tool for key management, GSKCapiCmd (**gsk8capicmd_64**).

GSKit version 8 no longer includes the key management utility, iKeyman (**gskikm.jar**). iKeyman is packaged with IBM Java version 6 or later and is now a pure Java application with no dependency on the native GSKit runtime. Do not move or remove the bundled `*java*/jre/lib/gskikm.jar` library.

The *IBM Developer Kit and Runtime Environment, Java Technology Edition, Version 6 and 7, iKeyman User's Guide for version 8.0* is available on the Security Access Manager Information Center. You can also find this document directly at:

> http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/security/60/iKeyman.8.User.Guide.pdf

Note:

GSKit version 8 includes important changes made to the implementation of Transport Layer Security required to remediate security issues.

The GSKit version 8 changes comply with the Internet Engineering Task Force (IETF) Request for Comments (RFC) requirements. However, it is not compatible with earlier versions of GSKit. Any component that communicates with Security Access Manager that uses GSKit must be upgraded to use GSKit version 7.0.4.42, or 8.0.14.26 or later. Otherwise, communication problems might occur.

IBM Tivoli Directory Server

IBM Tivoli Directory Server version 6.3 FP17 (6.3.0.17-ISS-ITDS-FP0017) is included on the *IBM Security Access Manager for Web Version 7.0* product image or DVD for your particular platform.

You can find more information about Tivoli Directory Server at:

> http://www.ibm.com/software/tivoli/products/directory-server/

IBM Tivoli Directory Integrator

IBM Tivoli Directory Integrator version 7.1.1 is included on the *IBM Tivoli Directory Integrator Identity Edition V 7.1.1 for Multiplatform* product image or DVD for your particular platform.

You can find more information about IBM Tivoli Directory Integrator at:

> http://www.ibm.com/software/tivoli/products/directory-integrator/

IBM DB2 Universal Database™

IBM DB2 Universal Database Enterprise Server Edition, version 9.7 FP4 is provided on the *IBM Security Access Manager for Web Version 7.0* product image or DVD for your particular platform. You can install DB2® with the Tivoli Directory Server software, or as a stand-alone product. DB2 is required when you use Tivoli Directory Server or z/OS® LDAP servers as the user registry for Security Access Manager. For z/OS LDAP servers, you must separately purchase DB2.

You can find more information about DB2 at:

http://www.ibm.com/software/data/db2

IBM WebSphere® products

The installation packages for WebSphere Application Server Network Deployment, version 8.0, and WebSphere eXtreme Scale, version 8.5.0.1, are included with Security Access Manager version 7.0. WebSphere eXtreme Scale is required only when you use the Session Management Server (SMS) component.

WebSphere Application Server enables the support of the following applications:
- Web Portal Manager interface, which administers Security Access Manager.
- Web Administration Tool, which administers Tivoli Directory Server.
- Common Auditing and Reporting Service, which processes and reports on audit events.
- Session Management Server, which manages shared session in a Web security server environment.
- Attribute Retrieval Service.

You can find more information about WebSphere Application Server at:

http://www.ibm.com/software/webservers/appserv/was/library/

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

Visit the IBM Accessibility Center for more information about IBM's commitment to accessibility.

Technical training

For technical training information, see the following IBM Education website at http://www.ibm.com/software/tivoli/education.

Support information

IBM Support provides assistance with code-related problems and routine, short duration installation or usage questions. You can directly access the IBM Software Support site at http://www.ibm.com/software/support/probsub.html.

The *IBM Security Access Manager for Web Troubleshooting Guide* provides details about:
- What information to collect before you contact IBM Support.
- The various methods for contacting IBM Support.
- How to use IBM Support Assistant.
- Instructions and problem-determination resources to isolate and fix the problem yourself.

Note: The **Community and Support** tab on the product information center can provide more support resources.

Chapter 1. Introduction to the authorization API

The IBM Security Access Manager Runtime for Java component includes the Java language version of a subset of the Security Access Manager authorization API.

The authorization API consists of a set of classes that provide Java applications with the ability to interact with Security Access Manager to make authentication and authorization decisions.

This chapter contains the following topics:
- "Accessing the Javadoc HTML documentation"
- "Authorization API components"
- "Requirements for developing Java applications" on page 3
- "Deploying a Java authorization API application" on page 4

Accessing the Javadoc HTML documentation

The Javadoc information is available in the IBM Security Access Manager for Web application developer kit (ADK). Use the Javadoc information along with this guide, and other Java reference materials, to add product authorization and security services to new or existing Java applications.

Application developers who update an existing Security Access Manager for Web application must consult the Javadoc HTML documentation for deprecated Java APIs before modifying the code.

Copy the Javadoc HTML information with the entire `AM_BASE/nls/javadocs` directory to another location on your development system. Uninstall the IBM Security Access Manager for Web ADK and runtime components. The Security Access Manager Runtime for Java component is the only component required for running Java applications. See Table 1 on page 2 for the Javadoc installation location.

Authorization API components

The authorization API Java classes are installed as part of the IBM Security Access Manager Runtime for Java component.

These classes communicate directly with the Security Access Manager authorization server by establishing an authenticated, Secure Socket Layer (SSL) session with the authorization server process. The authorization server services these requests in the same manner that it services requests from the authorization C API.

Table 1 on page 2 list the files related to the authorization API that are installed as part of the IBM Security Access Manager Runtime for Java component. The Javadoc information, even though it is installed as part of the Security Access Manager ADK component, is listed in the table for completeness.

Table 1. Files associated with the IBM Security Access Manager Runtime for Java and ADK components

Directory	Files	File description
AM_BASE/nls/javadocs /pdjrte/index.html	index.html (and many others)	Javadoc HTML documentation for the Java classes and methods provided with the Security Access Manager Java runtime component.
JAVA_HOME/lib/ext	PD.jar	The Java Archive (JAR) file which contains the classes and methods associated with the administration APIs. **Note:** When you use the **pdjrtecfg** command-line interface to configure the Security Access Manager Java runtime component to a particular JRE, the archive file is copied to *JAVA_HOME*/lib/ext. There is no need to modify the CLASSPATH in your environment to access the classes and methods defined in this archive file. For the IBM WebSphere Application Server version 8.x JRE, put **PD.jar** in WAS_HOME/tivoli/tam and add it to the CLASSPATH when using the JRE standalone. For example, if you are using the JRE outside of an IBM WebSphere Application Server JVM.
AM_BASE/example/ pdadminapi_demo/java	README.PDAdminDemo PDAdminDemo.java PDAdminDemo.class PDAdminDemo$ConsoleEraser.class	A demonstration program is provided to illustrate the use of the administration Java APIs. You can copy the demonstration program to any directory. The readme file explains how to run and recompile the demonstration program.
AM_BASE/example/authz_demo/ java	PDCallbackHandler.class PDDemoSetup.class PDDemoSetup.java PDJaasDemo$1.class PDJaasDemo.class PDJaasDemo.java PDListObjectsDemo.class PDListObjectsDemo.java PDPermissionDemo.class PDPermissionDemo.java README.JaznDemo	These files consist of various demonstrations which illustrates the use of Security Access Manager Java authorization APIs. See README.JaznDemo for a description on how to run the various demonstrations.
AM_BASE/example/ localremote_demo/java	PDLRAuthzDemo1.class PDLRAuthzDemo1.java PDLRAuthzDemo2$1.class PDLRAuthzDemo2$2.class PDLRAuthzDemo2.class PDLRAuthzDemo2.java PDLRExerciseDialog$1.class PDLRExerciseDialog$2.class PDLRExerciseDialog$3.class PDLRExerciseDialog$4.class PDLRExerciseDialog.class PDLRExerciseDialog.java PDLRTestDemo.class PDLRTestDemo.java PDtamdemoException.class PDtamdemoException.java PDTimer.class PDTimer.java README.PDLocalRemoteDemo	This file consists of a demonstration that illustrates the use of both the local and remote modes of Security Access Manager administration and authorization APIs. The demonstration provides a graphical user interface for defining the various setup parameters. See the README.PDlocalRemoteDemo for a description on how to generate the documentation for the demonstration classes.

To make the JAR files listed in Table 1 available to a particular JRE, see "Configuring the IBM Security Access Manager Runtime for Java component to a particular environment" on page 4.

Requirements for developing Java applications

To develop Java applications that use the Security Access Manager authorization API, you must install and configure the required software.

Security Access Manager software requirements

This section describes the software required to run Security Access Manager.

You must install and configure a Security Access Manager secure domain. If you do not have a Security Access Manager secure domain installed, install one before beginning application development. The minimum installation consists of a single system with the following Security Access Manager components installed:

- Security Access Manager runtime environment
- IBM Security Access Manager Runtime for Java component
- Security Access Manager policy server
- Security Access Manager authorization server
- Security Access Manager ADK

If you already have a Security Access Manager secure domain installed and want to add a development system to the domain, the minimum Security Access Manager installation consists of the following components:

- Security Access Manager runtime environment (see Note 1)
- IBM Security Access Manager Runtime for Java component
- Security Access Manager ADK

For Security Access Manager installation instructions, refer to the section of the *IBM Security Access Manager for Web: Installation Guide* for your operating system platform.

Note:

1. The Security Access Manager runtime environment component is *not* needed to develop or deploy a Security Access Manager Java application.
2. You can copy the Javadoc HTML information, consisting of the entire *AM_BASE*`/nls/javadocs` directory tree, to another location on your development system and then uninstall the Security Access Manager ADK and runtime components. Only the IBM Security Access Manager Runtime for Java component is necessary for running Java applications.

JRE requirements

You can use either of the supported JREs listed in the *IBM Security Access Manager for Web Product Overview* for developing and deploying your Security Access Manager Java applications.

The installation of an appropriate JRE is required when using the Security Access Manager authorization API Java classes and methods. The base installation DVD contains an optionally installable JRE.

After you have installed a suitable JRE, configure it for use with Security Access Manager as outlined in the next section, Configuring the IBM Security Access Manager Runtime for Java component to a particular environment.

Configuring the IBM Security Access Manager Runtime for Java component to a particular environment

Configure the IBM Security Access Manager Runtime for Java component to use the correct JRE on the system by using the **pdjrtecfg** command.

The **pdjrtecfg** command copies the Security Access Manager JAR files to the `JAVA_HOME/lib/ext` directory of the JRE. When the JAR files are copied to the directory, it automatically makes the Security Access Manager classes and methods available. The CLASSPATH in your environment does not need to be modified. The IBM Security Access Manager Runtime for Java component can be configured to several different JREs on the same system, if required. See the *IBM Security Access Manager for Web: Command Reference* for details.

Note: For WebSphere Application Server, version 8.0, the Security Access Manager JAR file is copied to `WAS_HOME/tivoli/tam directory`.

Security requirements

This section explains the security requirements for the `SvrSslCfg` class.

The `PD.jar` file is signed and verified in this version of Security Access Manager.

Use the `SvrSslCfg` Java class (`com.tivoli.pd.jcfg.SvrSslCfg`) to create configuration files that are to be used by Java applications. See "Configuring a Java application into the secure domain" on page 17 for details on using the `SvrSslCfg` class.

Note: The **`svrsslcfg`** command-line interface and the `SvrSslCfg` Java utility are *not* interchangeable. Do not use the **`svrsslcfg`** command-line interface to do any of the following tasks:
- Create configuration files to use with Java applications.
- Create configuration files to use with C applications.

Deploying a Java authorization API application

After developing and testing a Java application that uses the Security Access Manager authorization API, you can deploy the application to systems that are configured as part of a Security Access Manager secure domain.

The IBM Security Access Manager Runtime for Java component is the only Security Access Manager component that must be installed on a system to run a Security Access Manager Java application.

The IBM Security Access Manager Runtime for Java component is *not* needed for running Java applications.

Note: Information about installing the IBM Security Access Manager Runtime for Java component can be found in the *IBM Security Access Manager for Web: Installation Guide*.

For information about troubleshooting Java applications with Security Access Manager, see the *IBM Security Access Manager for Web: Troubleshooting Guide*.

Chapter 2. Authorization API Java classes overview

This section provides an overview of the Security Access Manager authorization API Java classes.

The classes are:
- "Classes from com.tivoli.pd.jazn package"
- "Classes from com.tivoli.pd.jutil package" on page 8

See the Javadoc information in the Security Access Manager ADK for detailed documentation about all these classes.

Review Appendix B, "Deprecated Java classes and methods," on page 35 before modifying an existing Java application. A number of classes and methods are deprecated in this version of Security Access Manager.

Classes from com.tivoli.pd.jazn package

This section describes classes from the com.tivoli.pd.jazn package.

Use these classes from the com.tivoli.pd.jazn package.

The classes are:
- "PDAuthorizationContext: method and constructor summary"
- "PDLoginModule: method and constructor summary"
- "PDPermission: method and constructor summary" on page 6
- "PDPrincipal: method and constructor summary" on page 7

PDAuthorizationContext: method and constructor summary

This section describes methods and constructors for the PDAuthorizationContext class.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Table 2. Methods and constructors for PDAuthorizationContext class

Methods and Constructors	Description
PDAuthorizationContext	Constructor that creates an instance of the `PDAuthorizationContext` class.
close	Closes the context.
getMode	Returns the mode this application is configured for.

PDLoginModule: method and constructor summary

The `PDLoginModule` class handles the authentication of a Security Access Manager user using the Java Authentication and Authorization Service (JAAS).

Then, it creates a `PDPrincipal` object containing the Security Access Manager user credentials when the authentication is successful.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Table 3. Methods and constructors for PDLoginModule class

Methods and Constructors	Description
`PDLoginModule`	Constructor that creates an instance of the `PDLoginModule` class.
abort	Aborts the authentication (second phase).
commit	Commits the authentication (second phase).
`getDefaultAuthorizationContext`	Gets the default Security Access Manager authorization context for all instances of the `PDLoginModule` class.
initialize	Initializes the `LoginModule`.
login	Authenticates the user (first phase).
logout	Logs the user out.
`setDefaultAuthorizationContext`	Sets default Security Access Manager authorization context for all instances of the `PDLoginModule` class.

PDPermission: method and constructor summary

The `PDLoginModule` class handles the authentication of a Security Access Manager user using the Java Authentication and Authorization Service (JAAS).

`PDPermission` uses Security Access Manager as the authorization engine for normal Java 2 permission checks.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Table 4. Methods and constructors for PDPermission class

Methods and Constructors	Description
PDPermission	Constructors that create an instance of the `PDPermission` class.
equals	Determines whether this `PDPermission` is equivalent to the input object.
getActions	Returns a String representation of this object.
getPDException	Provides access to any exception information received on the last implies (Permission) call.
hashCode	Returns the hash code value for this object.
implies	Methods that determine whether the Security Access Manager grants the permissions in this `PDPermission` object to the specified `PDPrincipal`.

PDPrincipal: method and constructor summary

The `PDPrincipal` class implements the Principal interface and contains the credentials of an authenticated Security Access Manager user.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Table 5. Methods and constructors for PDPrincipal class

Methods and Constructors	Description
PDPrincipal	Constructors that create an instance of the `PDPrincipal` class.
addAttribute	Returns a new `PDPrincipal` that contains the added credential attribute.
addAttrlist	Returns a new `PDPrincipal` that contains the modified credential attribute list.
addGroupMemberships	Returns a new `PDPrincipal` that adds these group memberships to the current `PDPrincipal`.
equals	Compares the specified Object with this `PDPrincipal` for equality.
getAttribute	Returns the values for the attribute.
getAttributeNames	Returns the attribute names in the credential attribute list.
getAttributeValue	Returns the value for the attribute.
getAttrlist	Returns a copy of the credential attribute list for this principal.
getEntitlements	Returns all the objects to which this `PDPrincipal` has the specified access.
getName	Returns a string name of this `PDPrincipal`.
getPAC	Obtains an architecture and network independent encoding of this principal.
hashCode	Returns a hash code for this `PDPrincipal`.
implies	Checks if whether the specified Subject is implied by this object.
readExternal	Reads the state of the `PDPrincipal` instance from a stream.
removeAttribute	Returns a new `PDPrincipal` that does not contain the named attribute.

Table 5. Methods and constructors for PDPrincipal class (continued)

Methods and Constructors	Description
removeGroupMemberships	Returns a new `PDPrincipal` that removes these group memberships from the current `PDPrincipal`.
setAttribute	Returns a new `PDPrincipal` that contains the modified attribute.
setAttrlist	Return a new `PDPrincipal` that contains the modified credential attribute list.
setContext	Sets the authorization context of this `PDPrincipal` instance.
toString	Returns a string representation of this `PDPrincipal`.
writeExternal	Saves the state of the `PDPrincipal` instance to a stream (that is, serializes it).

Classes from com.tivoli.pd.jutil package

This section describes the classes from the com.tivoli.pd.jutil package.

Use these classes from the com.tivoli.pd.jutil package.

The classes are:
- "PDAttrs: method and constructor summary"
- "PDAttrValue: method and constructor summary" on page 9
- "PDAttrValueList: method and constructor summary" on page 10
- "PDAttrValues: method and constructor summary" on page 10
- "PDStatics" on page 11

PDAttrs: method and constructor summary

The PDAttrs class represents a collection of attributes. Attributes are used to encapsulate input and output data sent to and received from authorization and administration service functions. Each attribute consists of entries that have a *name* and one or more *values*. The names are Strings, and the values can of type String, byte array, Long, or `PDAdmSvcPobj`.

Several of the constructors for this class use the `context` parameter, of class `com.tivoli.pd.jutil.PDBasicContext`. This class is a superclass of the Security Access Manager contexts. The context to be passed for the authorization APIs is a subclass such as PDContext.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Table 6. Methods and constructors for PDAttrs class

Methods and Constructors	Description
PDAttrs	Constructors that create an instance of the PDAttrs class.
add	Methods for adding the specified value to the collection of values for the specified name in this PDAttrs.
addAll	Adds all the elements in the specified PDAttrs to this PDAttrs.
allowDups	Returns the current value of allowDups.
clear	Clears the current PDAttrs.
clone	Clones the current PDAttrs.
delete	Removes the named attribute from the PDAttrs.
entrySet	Returns a set view of the entries in the PDAttrs.
equals	Indicates whether some other Object is equal to this one.
get	Deprecated. Use getValues instead.
getAttrlist_t	Method getAttrlist_t. Adds the contents of this PDAttrs to the attrlist_t data structure passed into this method.
getNames	Method getNames. Returns the keys in a String array.
getQoP	Returns the current value of QoP.
getValues	Returns the values to which this PDAttrs maps the specified key.
hashCode	Returns a hashcode for the current object.
iKeySet	Method iKeySet. Returns the keys HashSet in upper-cased Strings.
keySet	Returns a set view of the keys contained in this PDAttrs.
setAttrlist_t	Method setAttrlist_t. Sets the contents of this PDAttrs to the attrlist_t data structure passed into this method.
setQoP	Sets the current value of QoP.
size	Returns the number of key-values mappings in the current PDAttrs.
toString	Returns a String representation of this object.

PDAttrValue: method and constructor summary

The PDAttrValue class represents the value of a Security Access Manager attribute. A value might be a String, a byte array, a Long, or a PDAdmSvcPobj.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Table 7. Methods and constructors for PDAttrValue class

Methods and Constructors	Description
PDAttrValue	Constructors that create an instance of the `PDAttrValue` class.
clone	Returns a clone of the object.
equals	Indicates whether some other Object is equal to this one.
getType	Returns the type of the current attribute value.
getValue	Returns the value of the current attribute, which can then be examined.
hashCode	Returns a `hashcode` for the current object.
toString	Returns a String representation of this object.

PDAttrValueList: method and constructor summary

The `PDAttrValueList` class represents the list of values for one attribute. Each value must be a `PDAttrValue`. The list is ordered and allows duplicates.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Table 8. Methods and constructors for PDAttrValueList class

Methods and Constructors	Description
PDAttrValueList	Constructors that create an instance of the `PDAttrValueList` class.
add	Methods for inserting the specified element at the specified position in this list, moving all subsequent elements to a higher index.
addAll	Methods for inserting all the elements in the specified collection into this list, starting at the specified offset, shifting any subsequent elements to a higher index.
clone	Returns a clone of this object.
equals	Indicates whether some other Object is equal to this one.
hashCode	Returns a `hashcode` for the current object.
set	Replaces the element at the specified position in this list with the specified element.
toString	Returns a String representation of this object.

PDAttrValues: method and constructor summary

The `PDAttrValues` class represents a collection of values for a particular `PDAttr`. This particular implementation is a Set, so duplicates are not allowed in a particular `PDAttrValues` object.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Table 9. Methods and constructors for PDAttrValues class

Methods and Constructors	Description
PDAttrValues	Constructors that create an instance of the `PDAttrValues` class.
add	Methods for adding the input `PDAttrValue` to this `PDAttrValues`.
addAll	Adds all the elements in the specified collection to this collection.
clone	Returns a clone of this object.
encode	Returns a byte array which contains the DER encoded representation of this object.
equals	Indicates whether some other Object is equal to this one.
hashCode	Returns a `hashcode` for the current object.
toString	Returns a String representation of this object.

PDStatics

The PDStatics class contains various constants used in the Java administration and authorization classes.

See the Javadoc information in the Security Access Manager ADK for detailed documentation about this class.

Chapter 3. Java security

The Security Access Manager authorization Java classes provide an implementation of Java security code that is fully compliant with the Java 2 security model and the Java Authentication and Authorization Service (JAAS).

This chapter contains the following topics:

- "Java 2 security with Security Access Manager"
- "Java Authentication and Authorization Service (JAAS) model" on page 14

Java 2 security with Security Access Manager

The Java 2 security architecture is policy-based, and allows for fine-grained access control.

When the code is loaded, it is assigned *permissions* based on the security policy currently in effect. Each permission specifies a permitted access to a particular resource, such as *read* access to a specified file, or *connect* access to a specified host and port. The policy specifies which permissions are available for code from various signers and locations. The policy can be initialized from an external configuration file.

Code can access a resource only if the permission that guards the resource gives the code explicit permission. The new concepts of permission and policy enable the Java 2 to offer fine-grained, highly configurable, flexible, and extensible access control. Such access control can now be specified for all Java code, including applications, beans, and servlets.

The Security Access Manager authorization server provides an SSL-based access mode for handling remote authorization calls. The Security Access Manager Java authorization API uses this socket-based capability to provide functionality equivalent to that provided in the authorization C API by the `azn_decision_access_allowed_ext()` function.

The `azn_decision_access_allowed_ext()` function requires the following information:

- Authentication information
- Resource name
- Access mode

The Java 2 permission model provides the resource name and the access mode. The Java Authentication and Authorization Service (JAAS) extensions to the Java 2 model provide the authentication information.

Security Access Manager functions as a back end for normal Java 2 permission checks by providing:

- A custom JAAS `LoginModule` that manufactures authentication credentials.
- A custom permission class that knows how to locate and call Security Access Manager.

Java Authentication and Authorization Service (JAAS) model

The Java Authentication and Authorization Service model serves the browsers that first popularized Java well, as it effectively deals with the issues of mobile code.

The Java 2 permission model takes the following information into account:

- The physical origin (the directory or URL) of the classes that are currently active.
- The logical origin of those classes.
- The identity of the organization that produced the classes, as proved by digital signature.

JAAS augments the current Java 2 runtime with knowledge of the user who is runs the application. The knowledge provides the authentication information needed when implementing the security model.

JAAS augments the Java 2 security model to enable the following features:

- Specification of permissions based on a user identity.
- Enforcement of those permissions at application runtime.

The two features provide the authorization functionality needed when implementing the security model.

The following sections describe how Security Access Manager authorization Java Classes use the JAAS model:

- "Authenticating users and obtaining credentials"
- "Authorizing access requests" on page 15

Authenticating users and obtaining credentials

The Security Access Manager Java-based authentication feature is built around the Java Authentication and Authorization Services (JAAS) model.

Security Access Manager provides one JAAS `LoginModule`. You can use the module in two different ways:

- To authenticate a user and obtain the user credentials
- To obtain only the user credentials

Authenticating with a user name and password

This section explain how the JAAS `LoginModule` authenticates with a user name and password.

To authenticate a user, the JAAS `LoginModule` requires that the calling application to provide the following attributes:

- A principal name, specified as either a short name or an X.500 name (DN)
- A password

The `LoginModule` authenticates the principal and returns the Security Access Manager credential. The `LoginModule` expects the calling application to provide the following information:

- The user name, through a `javax.security.auth.callback.NameCallback`.
- The password, through a `javax.security.auth.callback.PasswordCallback`.

When the Security Access Manager credential is successfully retrieved, the `LoginModule` creates a Subject and a `PDPrincipal`.

Retrieving credentials without authenticating

To retrieve credentials without authenticating, the calling application can call the JAAS `LoginModule` with only a principal name specified as a short name or an X.500 distinguished name (DN).

The `LoginModule` expects the calling application to provide the user name through a `javax.security.auth.callback.NameCallback`.

The login configuration file

Use the login configuration file to specify whether a user name and password, or only a user name, are required at login.

You can use the optional entry `nameOnly` in the login configuration file to specify which of two login modes your application uses. You can configure the module to require either a user name and a password (default behavior), or only a user name.

To require only the user name, specify `nameOnly=true` in the configuration file. See Figure 1 on page 22.

If `nameOnly` is omitted or specified to be "false", both the user name and the password are required.

Authorizing access requests

This section explains how access requests are handled by Security Access Manager authorization Java classes.

The Security Access Manager authorization Java classes are built around JAAS and the Java 2 security model. The Security Access Manager API closely follows the Java 2 permission model.

Note: For more information on the Java 2 security model, see: http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html.

The Security Access Manager authorization API Java classes provide a permission class named `com.tivoli.pd.jazn.PDPermission` . This class extends the abstract class `com.ibm.IBMPermission`, which extends the abstract class `java.security.Permission`. The `PDPermission` class establishes the SSL-protected socket communications protocol which is used to talk to Security Access Manager.

Create an entry in the JAAS policy file to ensure that the JAAS security code calls the `implies()` method in the `PDPermission` class described here. You can specify the entry based on a particular codebase as required

Define your JAAS policy in its own file and specify the URL in the `java.security` file using the property auth.policy.url.X (where X is an integer). For example:

```
auth.policy.url.1=file:${java.home}/lib/security/jaas.policy
```

Alternatively, you can use the Java interpreter **-D** flag to specify the JAAS policy file. For example:

```
java -Dauth.policy.url.1=file:/opt/PolicyDirector/etc/jaas.policy
```

You can specify the JAAS policy directly in the `java.policy` file found in *`java_home`*`/lib/security`.

```
grant signedBy "xxx" codeBase "file:/E:/Program Files/aaa/bbb/ccc"
principal com.tivoli.pd.jazn.PDPrincipal "*" {
permission com.tivoli.pd.jazn.PDPermission "ignoreme" "a";
};
```

The contents of the action string `ignoreme` are unimportant because the `PDPermission` class ignores them. This is because Security Access Manager acts as the repository for security policy. The intent of this entry is to have the Java security code call the `implies()` method when a resource manager checks to see if a permission is held.

The `PDPermission` class implements constructors and supporting methods, including:

implies()
Checks whether Security Access Manager grants the specified permissions.

equals()
Determines if two `PDPermission` objects are equal.

getActions()
Returns the canonical string representation of the actions.

hashCode()
Returns the hash code value for the object.

The `implies()` method flow consists of the following steps:

1. Use the static `getSubject()` method to retrieve the current **Subject → Subject** that was created by the PDLoginModule class, and placed on the current thread of execution by the resource manager.
2. If the Subject contains a Principal of type `com.tivoli.pd.jazn.PDPrincipal`, then the appropriate credentials are secured for the call to Security Access Manager.

The following sample illustrates how a resource manager, such as a Web server or Enterprise JavaBeans container, places the Subject on the current thread of execution.

```
Subject.doAs(whoami, new java.security.PrivilegedAction() {
public java.lang.Object run() {}
});
```

At this point the `PDPermission` class has all the information required to make the authorization call to Security Access Manager.

The following code sample shows a typical authorization check that invokes Security Access Manager through the `PDPermission` class implementation. The `checkPermission()` method returns quietly unless it fails, in which case it throws a `java.lang.SecurityException`.

```
PDPermission perm = new PDPermission("/MyResourceManager/private",
"[simple]rT[newActionGroup1]Z");

SecurityManager.checkPermission(perm);
```

Chapter 4. Java application development

You can develop Java applications that use Security Access Manager.

For more information about Java application development, see the following topics:

- "Configuring a Java application into the secure domain"
- "Configuring the Java Authentication and Authorization Service" on page 21
- "Developing a resource manager" on page 22
- "Making authorization decisions outside of Java 2" on page 23
- "Obtaining entitlements for a specified user" on page 24

Configuring a Java application into the secure domain

Java applications that use Security Access Manager security must be configured into a Security Access Manager secure domain.

Security Access Manager provides a utility class called `com.tivoli.pd.jcfg.SvrSslCfg` that can be used to accomplish the necessary configuration and unconfiguration tasks.

This section describes configuration and unconfiguration tasks, and provides example command-line syntax for each task.

You can use `SvrSslCfg` to accomplish the following tasks:

- "Configuring an application server" on page 18
- "Unconfiguring an application server" on page 19
- "Adding a policy or authorization server" on page 20
- "Removing a policy or authorization server" on page 20
- "Changing a policy or authorization server" on page 20
- "Replacing a certificate" on page 20
- "Setting the port" on page 21
- "Setting the database directory" on page 21
- "Setting the database refresh interval" on page 21
- "Setting the application listening mode" on page 21
- "Setting the certificate refresh option" on page 21

The examples in this chapter use the values shown in Table 10:

Table 10. Sample information used for SvrSslCfg examples

Information	Value
Administrator user ID	`sec_master`
Administrator password	`secpw`

Table 10. Sample information used for SvrSslCfg examples (continued)

Information	Value
Policy server, TCP/IP communications port number, and rank (default port is 7135)	`ampolicy.myco.com:7135:1` This entry can also be used to specify a policy server *proxy*. The location, port, and rank of the policy server proxy must be specified. The default port for a proxy is 7138.
Authorization server, TCP/IP communications port number, and rank (default port is 7136)	`amazn.myco.com:7136:1`
Host name of Java application system	`jsys.myco.com`
TCP/IP port on which the application server listens for communications from the policy server	`999`
Application server password	`pw`
Security Access Manager application ID	`PDPermissionjapp` The application ID must be unique. Other instances of the application running on this or other systems must each be given a unique ID.
Security Access Manager domain	mydomain
Configuration file	(Windows example)`c:\am\config_file.conf` **Note:** SvrSslCfg creates the configuration file when called with `–action config`. When `SvrSslCfg` is called with other options (for example, `–action addsvr`), the configuration file is expected to exist.
Keystore file	(Windows example)`c:\am\keystore_file.ks` **Note:** SvrSslCfg creates this keystore file when called with `–action config`. When `SvrSslCfg` is called with other options (for example, `–action addsvr`), the keystore file is expected to exist.

A detailed command reference for the `–action config` class can be found in Appendix A, “com.tivoli.pd.jcfg.SvrSslCfg,” on page 27.

Configuring an application server

Configuring an application server creates user and server information in the user registry as well as creates local configuration and keystore files.

Security Access Manager uses a self-generated and self-signed certificate to authenticate its Secure Sockets Layer (SSL) communications. The Security Access Manager authorization API Java classes must be able to determine the certificate that Security Access Manager is using to establish its SSL communication.

You also must establish an identity for the Java application. The `SvrSslCfg` class is used to create a Security Access Manager user account for an application server and to store the server configuration and certificate information in local configuration and keystore files.

After obtaining the necessary information, use the `SvrSslCfg` option `-action config` to create the Security Access Manager application name, the configuration file, and the keystore file.

When using `-action config`, you must also specify whether you are creating or replacing the configuration and keystore files. The `-cfg_action create` option is used to initially create the configuration and keystore files. Use `cfg_action replace` if these files already exist. If the `-cfg_action create` option is used and the configuration or keystore files already exist, an exception is thrown.

Security Access Manager supports application servers in either remote mode or local mode. The following section shows a sample configuration command for each mode.

Configuring remote mode

onfiguring remote mode

Based on the sample information shown in Table 10 on page 17, the command to establish an SSL connection between `japp.myco.com` and the Security Access Manager secure domain, in remote mode, can be as follows:

```
java com.tivoli.pd.jcfg.SvrSslCfg -action config
-admin_id sec_master-admin_pwd secpw
-appsvr_id PDPermissionjapp -appsvr_pwd pw -host jsys.myco.com
-mode remote -port 999 -policysvr ampolicy.myco.com:7135:1
-authzsvr amazn.myco.com:7136:1 -cfg_file c:/am/config_file.conf
-key_file c:/am/keystore_file.ks -domain mydomain -cfg_action create
-certrefresh true
```

Configuring local mode

Based on the sample information shown in Table 10 on page 17, the command to establish an SSL connection between the Java application and Security Access Manager secure domain in local mode might be as follows:

```
java com.tivoli.pd.jcfg.SvrSslCfg -action config
-admin_id sec_master -admin_pwd secpw
-appsvr_id PDPermissionjapp -host jsys.myco.com
-mode local-port 999 -policysvr ampolicy.myco.com:7135:1
-authzsvr amazn.myco.com:7136:1-cfg_file c:/am/config_file.conf
-key_file c:/am/keystore_file.ks -domain mydomain -cfg_action create
-certrefresh true
```

Unconfiguring an application server

The `-action unconfig` option removes the user and server information from the user registry, deletes the local keystore file and removes information for this application from the configuration file but does not delete the configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action unconfig\
-admin_id sec_master -admin_pwd secpw\
-appsvr_id PDPermissionjapp -host jsys.myco.com\
-policysvr ampolicy.myco.com:7135:1 \
-cfg_file c:/am/config_file.conf -domain mydomain
```

The unconfiguration operation fails only if the caller is unauthorized or the policy server cannot be contacted.

Adding a policy or authorization server

The `-action addsvr` option adds a policy or authorization server to the application server configuration file.

To add a policy server:

```
java com.tivoli.pd.jcfg.SvrSslCfg -action addsvr \
-policysvr ampolicy3.myco.com:7135:2\
-cfg_file c:/am/config_file.conf
```

To add an authorization server:

```
java com.tivoli.pd.jcfg.SvrSslCfg -action addsvr \
-authzsvr am2azn.myco.com:7136:2\
-cfg_file c:/am/config_file.conf
```

Removing a policy or authorization server

Use the `-action rmsvr` option to remove a policy or authorization server from the configuration file.

To remove a policy server:

```
java com.tivoli.pd.jcfg.SvrSslCfg -action rmsvr\
-policysvr ampolicy.myco.com:7135:1\
-cfg_file c:/am/config_file.conf
```

To remove an authorization server:

```
java com.tivoli.pd.jcfg.SvrSslCfg -action rmsvr\
-authzsvr amazn.myco.com:7136:1\
-cfg_file c:/am/config)file.conf
```

Changing a policy or authorization server

Use the `-action chgsvr` option to change the port or rank for a policy or authorization server in the configuration file.

Do not use the `-action chgsvr` option to change the host name.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action chgsvr\
-policysvr ampolicy2.myco.com:7135:2\
-cfg_file c:/am/config_file.conf
```

or

```
java com.tivoli.pd.jcfg.SvrSslCfg -action chgsvr \
-authzsvr amazn.myco.com:7136:1 \
-cfg_file c:/am/config_file.conf
```

Replacing a certificate

The certificate in the keystore expires based on the certificate lifetime set on the policy server. After the certificate expires, the `-action replcert` option must be used to generate a new certificate. The new certificate replaces the existing certificate in the application server keystore file.

If a certificate become compromised, the `-action replcert` option can be used to invalidate an existing certificate.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action replcert\
-admin_id sec_master-admin_pwd secpw \
-appsvr_id PDPermissionjapp -cfg_file c:/am/config_file.conf
```

Setting the port

Use the -action setport option to set the port on which the application server listens.

Using the -action setport option only updates the application server configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setport\
-port 4321 -cfg_file c:/am/configfile
```

Setting the database directory

Use the -action setdbdir option on local-mode application servers to set the directory where a local copy of the policy database is stored.

Using the -action setdbdir option only updates the application server configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setdbdir\
-dbdir c:/production/policy -cfg_file c:/am/config_file.conf
```

Setting the database refresh interval

Use the -action setdbref option on local-mode application servers to set the refresh interval for the local copy of the policy database.

The time interval is specified in seconds. Using the -action setdbref option only updates the application server configuration file. The following example sets the interval to every 60 minutes.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setdbref \
-dbrefresh 3600 -cfg_file c:/am/config_file.conf
```

Setting the application listening mode

Use the -action setdblisten option on local-mode application servers to indicate whether the application listens for policy database update notifications.

Using the -action setdblisten option only updates the application server configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setdblisten\
-dblisten true -cfg_file c:/am/config_file.conf
```

Setting the certificate refresh option

Use the -action setcertref option on remote and local mode application servers to indicate whether the application server certificate is automatically renewed.

If set to true, the application server certificate is checked at application start time. If the certificate age is greater than one half its lifetime, the certificate is renewed.

Configuring the Java Authentication and Authorization Service

The Security Access Manager configuration steps follow the configuration methods supported by the Java Authentication and Authorization Service (JAAS).

This section describes how to set up and use a login configuration file with the Security Access Manager authorization API Java classes.

This section does not provide an overview of all the JAAS configuration options. To review the JAAS configuration information, see the following website: .http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136007.html

Complete the instructions in the following sections:
- "Creating a login configuration file"
- "Specify the login file location"

Creating a login configuration file

Use the sample file shown in Figure 1 as the basis for creating a login configuration file for use with Security Access Manager. No default login configuration file is shipped as part of Security Access Manager.

```
//// config.pd: Login configuration file for PDLoginModule

pd-debug {
com.tivoli.pd.jazn.PDLoginModule required debug=true;
};

pd {
com.tivoli.pd.jazn.PDLoginModule required;
};

pd-nopass {
com.tivoli.pd.jazn.PDLoginModule required nameOnly=true;
};
```

Figure 1. JAAS login configuration file

The last stanza allows applications that use `pd-nopass` in their LoginContext constructor to supply user names but not passwords. For more information about PDLoginModule and nameOnly, see the "The login configuration file" on page 15 section, or see the Javadoc information for `com.tivoli.pd.jazn.PDLoginModule`.

Specify the login file location

There are two ways to specify the login file location: by pointing to the login configuration file, or specifying the appropriate option in the command-line.

Choose one of the following ways to specify the location of the login file:
- Point to the login configuration file from the `JAVA_HOME/jre/lib/security/java.security` file.

  For example, a sample entry from the `java.security` file might look like this:

  ```
  login.config.url.1=file:d:/Java/j142ibm/jre/lib/security/config.pd
  ```
- Specify the appropriate `-D` option on the `java` command-line invocation, such as:

  ```
  –Djava.security.auth.login.config=./config.pd
  ```

  For more information, see the JAAS configuration documentation.

Developing a resource manager

A resource manager is a Java application that uses the JAAS and the Security Access Manager authorization API Java classes to make access control decisions.

The sample code in Figure 2 illustrates the tasks that the resource manager must perform.

```
// Identify the configuration status and callback routine
lc = new LoginContext("pd-debug", np);

// Drive the login() and commit() methods of the LoginModule class
lc.login();
whoami = lc.getSubject();
System.out.println(whoami);

// Become that user
Subject.doAsPrivileged(whoami, new java.security.PrivilegedAction() {
public java.lang.Object run() {
boolean worked;
java.security.Permission perm = new PDPermission("/test/private", "a");
try {
// sm is a reference to a SecurityManager
sm.checkPermission(perm);
worked = true;
}
catch (AccessControlException e) {
if (VERBOSE) e.printStackTrace();
worked = false;
}
if (worked) {
System.out.println("user " + user + " has
\"\""+perm.getActions()+"\" permission(s) to target
"+perm.getName());
} else {
System.out.println("user " + user + " DOES NOT HAVE
\"\""+perm.getActions()+"\" permission(s) to target
"+perm.getName());
}
}
}, (java.security.AccessControlContext)null ) ;
```

Figure 2. Resource manager task example

Making authorization decisions outside of Java 2

The Security Access Manager authorization API Java classes also support a completely Java-compliant usage of the Security Access Manager authorization check that is outside of the Java 2 and JAAS framework.

The `PDPrincipal` class includes the `implies()` method for performing authorization checks. To construct a `PDPrincipal`, a `PDAuthorizationContext` specifying the appropriate domain is required. Specifying the user name and password on the constructor results in authentication to Security Access Manager during construction of the object.

Specifying the user name and no password on the constructor results in a security check on the current environment.

The permission that must be held is:

```
permission javax.security.auth.AuthPermission "createPDPrincipal"
```

If authorized, the constructor retrieves the authentication information from Security Access Manager for that entity. The names that are supported on these constructors can either be Security Access Manager short names, or distinguished names.

Before calling the `implies()` method, construct a `PDAuthorization` context and construct a `PDPrincipal` object for the specified entity. Next, construct a `PDPermission` with the name of the requested resource, the protected object, and the requested action to be performed on that object.

Then invoke the `PDPrincipal.implies(PDPermission)` method to determine if the requested access to the specified object is allowed for the specified entity.

The sample in Figure 3 shows an example of how to perform these tasks.

```
PDAuthorizationContext ctxt = new PDAuthorizationContext(configURL);
PDPrincipal whoIsIt = new PDPrincipal(ctxt, "tom", "letmein".toCharArray());
PDPermission whatTheyWant = new PDPermission(ctxt, "everything", "abT");
boolean haveAccess = whoIsIt.implies(whatTheyWant);
if (haveAccess) {
// let them proceed...
} else {
// deny the requested access
}
```

Figure 3. Example showing authorization outside of Java 2

Obtaining entitlements for a specified user

The authorization API supports a service plug-in model that enables developers to add modules that extend the capabilities of Security Access Manager. The entitlements service plug-in is the only type of plug-in that you can call from a Java application at this time.

An entitlements service plug-in enables authorization API applications for a specific Security Access Manager secure domain to retrieve the entitlements for a user from the policy repository for that secure domain. An entitlement service allows a third-party application running in the secure domain to call a specific entitlements service based on its service ID. If no service ID is provided, the default entitlements service plug-in is called. An entitlements service plug-in, like other authorization service plug-ins, must be installed and configured before use.

Security Access Manager provides a default entitlement service called the Security Access Manager *protected objects entitlements service* that is specific to the Security Access Manager environment. The entitlement service plug-in accepts a single, multivalued string attribute that specifies one or more root nodes for searching the Security Access Manager protected object space along with an indicator of what access permissions are required. The plug-in returns a multi-valued attribute list of protected objects meeting the search criteria.

This entitlement service can be called from a Java application by using the `PDPrincipal.getEntitlements` method, which is equivalent to using the **`azn_entitlements_get_entitlements()`** function from a C application. Figure 4 on page 25 shows a call to the protected objects entitlements service requesting a list of objects in the `/AppData/AccountData` and `/AppData/EmployeeData` object trees to which the principal has **view** and **modify** permission.

```
PDAttrs attrsIn= new PDAttrs(myctxt, true);
PDAttrs attrsOut = new PDAttrs(myctxt, true);

// Does user have view and modify access to desired resources?

attrsIn.add(PDStatics.AZN_ENT_SVC_PD_POBJ_PATH,
"/AppData/AccountData");
attrsIn.add(PDStatics.AZN_ENT_SVC_PD_POBJ_PATH,
"/AppData/EmployeeData");
attrsIn.add(PDStatics.AZN_ENT_SVC_PD_POBJ_REQD_OPS,
"vm");

attrsOut = principal.jazn.getEntitlements(myctxt, PDStatics.AZN_ENT_SVC_PD_POBJ,
attrsIn);

// Is user entitled to anything?

PDAttrValues results = attrsOut.getValues(PDStatics.AZN_ENT_SVC_PD_POBJ_MATCHES);

if ((results == null) || (results.isEmpty())) {
System.out.println("Nothing found.");
break major;
}

// Process String or byte array results...
```

Figure 4. Using the PDPrincipal.getEntitlements method

The protected objects entitlements service returns a multivalued attribute list of the protected objects to which the principal has the specified access permission. The protected objects returned to the attribute list are either byte array or String entries. The sample code in Figure 5 demonstrates printing the results.

```
// Print output attributes if any returned
Set s = attrsOut.keySet();
if(!s.isEmpty())
{
System.out.println("Attributes returned: ");
System.out.println(attrs);
} else
System.out.println("No attributes returned.");
```

Figure 5. Processing PDAttrs returned

See the additional information about the entitlements service plug-in as well as the other types of authorization service plug-ins in the *IBM Security Access Manager for Web: Authorization C API Developer Reference*.

Appendix A. com.tivoli.pd.jcfg.SvrSslCfg

This class is used to configure, unconfigure, and modify the configuration information associated with a Security Access Manager Java application server.

```
public class SvrSslCfg extends java.lang.Object {
public static void main (java.lang.String[] argv)
throws PDException
}
```

The use of the `com.tivoli.pd.jcfg.SvrSslCfg` class can be summarized as follows:

```
java com.tivoli.pd.jcfg.SvrSslCfg -action ( config| unconfig | addsvr|
rmsvr| chgsvr| setport |
setdblisten | setdbref | replcert }
-admin_id admin_user_ID
-admin_pwd admin_password
-appsvr_id application_server_name
-appsvr_pwd application_server_password
-port port_number
-mode { local | remote }
-host Host_name_of_application_server
-policysvr policy_server_name:port:rank [,...]
-authzsvr authorization_server_name:port:rank [,...]
-cfg_file fully_qualified_name_of_configuration_file
-domain Tivoli_Acccess_Manager_domain
-key_file fully_qualified_name_of_keystore_file
-policysvr fully_qualified_name_of_truststore_file
-msg_id message_identifier
-dblisten { true | false }
-dbrefresh refresh_interval_in_seconds
-dbdir name_of_directory_for_local_policy_database
-cfg_action{ create | replace }
-certrefresh { true | false }-ssl_v3_enable { true | false }
-tls_v10_enable { true | false }
-tls_v11_enable { true | false }
-tls_v12_enable { true | false }
–cipher_suites java_cipher_suite_list
```

Compatibility Note: The `com.tivoli.mts.SvrSslCfg` class is deprecated in Security Access Manager. Existing applications must be modified to use the new **`com.tivoli.pd.jcfg.SvrSslCfg`** class as the deprecated class will be removed in a future version of the product.

After the successful configuration of a Security Access Manager Java application server, `SvrSslCfg` creates a user account and server entries that represent the Java application server in the Security Access Manager user registry.

In addition, `SvrSslCfg` creates a configuration file and two Java keystore files locally on the application server:

- A Java keystore file stores a client certificate
- The other Java keystore file stores a trusted signer certificate to validate the server certificate for secure channel communication.

The client certificate permits callers to make authenticated use of Security Access Manager services. Conversely, unconfiguration removes the user and server entries from the user registry and cleans up the local configuration and keystore files.

The contents of an existing configuration file can be modified by using the `SvrSslCfg` class. The configuration file and the keystore file must exist when calling `SvrSslCfg` with all options other than `–action config` or `–action unconfig`.

A complete list of the actions available in the `SvrSslCfg` class are outlined following the description of the parameters in Table 11.

You can specify multiple policy servers and authorization servers, giving each one a numeric rank, in the `-policysvr` and `-authzsvr` options of the `com.tivoli.pd.jcfg.SvrSslCfg` Java class.

The rank specifies in what order the application attempts to connect to the defined servers. For example, if two servers are specified, one with rank 1 and another with rank 2, the application attempts to connect to the server with rank 1. If a connection cannot be established to server 1, the application attempts to connect to the server with rank 2.

Even if only one server is specified, it still must have a rank setting.

Table 11. Description of parameters for the SvrSslCfg configuration action.

SvrSslCfg Parameter	**Value**
–admin_id *user_ID*	A Security Access Manager user with administrative privileges. This parameter is required.
–admin_pwd *password*	Password associated with the Security Access Manager administrative user specified. This parameter is required.
–appsvr_id *name*	The name of the application server. This parameter is required.
–port *port_number*	The TCP/IP port which the application server listens to for policy server notifications. This parameter is required.
–mode { local | remote }	Indicates whether the application server processes requests remotely or locally. This parameter is required.
–policysvr *hostname*:*port*:*rank* [,*hostname2*:*port2*:*rank2*...]	A list of Security Access Manager policy servers to which the application server can communicate. Format of this entry is host name, TCP/IP port number, and numeric rank, separated by colons. Multiple servers can be specified by separating them with commas. For example, the following indicates two policy servers, both using default TCP/IP port 7135, are available: `primary.myco.com:7135:1,secondary.myco.com:7135:2` This parameter is required.
–authzsvr *hostname*:*port*:*rank* [,*hostname2*:*port2*:*rank2*...]	A list of Security Access Manager authorization servers to which the application server can communicate. Format of this entry is host name, TCP/IP port number, and numeric rank, separated by colons. Multiple servers can be specified by separating them with commas. For example, the following indicates 2 authorization servers, both using default TCP/IP port 7136, are available: `secazn.myco.com:7136:2,primazn.myco.com:7136:1` This parameter is required.

Table 11. Description of parameters for the SvrSslCfg configuration action. (continued)

SvrSslCfg Parameter	Value
–cfg_file *file_name*	Fully qualified name of the configuration file on the application server. **`SvrSslCfg –action config`** creates this file. The file name must have a .conf suffix. You can specify any valid name. This parameter is required.
–key_file *file_name*	Fully qualified name of the keystore file on the application server. **`SvrSslCfg –action config`** creates this file. The file name must have a .ks suffix. You can specify any valid name. This parameter is required.
-policysvr_truststore *file_name*	Fully qualified file name of the truststore file for the signer certificate of the policy server. This parameter is required if you are generating the configuration parameters to connect to a policy server different from the one that is configured for this Java runtime environment. If this parameter is not supplied, the Java application server must be configured to the same policy server as the Java runtime environment. This parameter is optional.
–msg_id *message_identifier*	An identifier that determines the directory in which to locate the trace and log files that are generated when using this application server. This identifier is used only if Tivoli Common Directory logging is enabled for the IBM Security Access Manager Runtime for Java. See the *IBM Security Access Manager for Web: Troubleshooting Guide* for more information on Tivoli Common Directory logging, message files, and message file locations. This parameter is optional. There is no default value.
–domain *domain_name*	The Security Access Manager domain for the application server. This parameter is optional. The default value is the local domain.
–appsvr_pwd *password*	The password for the user account in the user registry associated with the application server. This parameter is optional. If it is specified, the password must meet the current password rules in effect. If it is omitted, a default password is automatically generated.
–host *host_name*	Host name of the application server. This parameter is optional. The default value is the local host.
–desc *description*	Description of the application server. This parameter is optional. The default value is empty (no description).
–groups *group_names*	The names of special groups the application server belongs to. This parameter is optional. The default value is empty (no special groups).

Table 11. Description of parameters for the SvrSslCfg configuration action. (continued)

SvrSslCfg Parameter	**Value**
–dblisten { true | false }	Indicates whether the application server listens for policy database updates. This parameter is optional. The default value is **true**. This parameter is ignored when the mode parameter is set to **remote**.
–dbdir *directory_name*	The name of the directory to be used for the local copy of the policy database. This parameter is optional. If it is not specified, the default directory is the db directory, located just under the Security Access Manager installation directory: `installation_directory/db` This parameter is ignored when the mode parameter is set to **remote**.
–dbrefresh *number_of_seconds*	Indicates the time interval, in seconds, that the application server polls the policy server for policy database updates. This parameter is optional. Value must be greater than or equal to zero. The default value is **600** seconds, or every 10 minutes. This parameter is ignored if the mode parameter is set to **remote**.
–cfg_action { create | replace }	Indicates whether the configuration and keystore files must be created on the application server or replaced. This parameter is optional. The default action is **replace**. When the `create` option is specified but the files exist, an exception is raised. When the `replace` option is specified, the configuration and keystore files must exist.
–certrefresh { true | false }	Indicates whether the application certificate must be renewed automatically at application startup. The certificate renewal is triggered when the certificate lifetime has past the half life point and is not expired. **Note:** If the certificate expires, it cannot be renewed by restarting the application. Use the following command to replace the certificate manually: `java com.tivoli.pd.jcfg.SvrSslCfg -action replcert -admin_id sec_master -admin_pwd pwd \ -cfg_file <conf file of Java application>`
ssl_v3_enable {true | false}	Indicates whether to enable or disable SSL v3 protocol for secure channel communications. This parameter is optional. The default value is **true**.
tls_v10_enable {true | false}	Indicates whether to enable or disable TLS v1.0 protocol for secure channel communications. This parameter is optional. The default value is **true**.
tls_v11_enable {true | false}	Indicates whether to enable or disable TLS v1.1 protocol for secure channel communications. This parameter is optional. The default value is **true**.
tls_v12_enable {true | false}	Indicates whether to enable or disable TLS v1.2 protocol for secure channel communications. This parameter is optional. The default value is **true**.

Table 11. Description of parameters for the SvrSslCfg configuration action. (continued)

SvrSslCfg Parameter	Value
–cipher_suites *java_cipher_suite_list*	The `java_cipher_suite_list` is a comma-separated list of Java cipher suite names. The cipher suite names can be found at http://publib.boulder.ibm.com/infocenter/java7sdk/v7r0/index.jsp?topic=%2Fcom.ibm.java.security.component.doc%2Fsecurity-component%2Fjsse2Docs%2Fciphersuites.html. For example: `-cipher_suites` *SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA, SSL_DHE_DSS_WITH_AES_128_CBC_SHA, SSL_DHE_DSS_WITH_AES_128_CBC_SHA256*

Note: The host name is used to build a unique name (identity) for the application. The **pdadmin** user list command displays the application identity name in the following format:

server_name/host_name

The **pdadmin server list** command displays the server name in a slightly different format:

server_name-host_name

–action config

The –action config option configures an application server.

Configuring a server creates user and server information in the user registry and creates local configuration and keystore files on the application server. Use the –action unconfig option to reverse this operation.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action config
-admin_id admin_user_ID
-admin_pwd admin_password
-appsvr_id application_server_name
-appsvr_pwd application_server_password
-port port_number
-mode { local | remote }
[ -host Host_name_of_application_server ]
-policysvr policy_server_name:port:rank [,...]
-authzsvr authorization_server_name:port:rank [,...]
-cfg_file fully_qualified_name_of_configuration_file
[ -domain Tivoli_Acccess_Manager_domain ]
-key_file fully_qualified_name_of_keystore_file
[ -policysvr_truststore fully_qualified_name_of_truststore_file ]
[ -cfg_action{ create | replace } ]

-certrefresh { true | false }
-ssl_v3_enable { true | false }
-tls_v10_enable { true | false }
-tls_v11_enable { true | false }
-tls_v12_enable { true | false }
-cipher_suites java_cipher_suite_list
```

–action unconfig

The –action unconfig option unconfigures an application server.

The **–action unconfig** option also removes the user and server information from the user registry, deletes the local keystore file and removes information for this application from the configuration file but does not delete the configuration file. The unconfiguration operation fails only if the caller is unauthorized or the policy server cannot be contacted.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action unconfig
-admin_id admin_user_ID
-admin_pwd admin_password
-appsvr_id application_server_name
[ -host host_name_of_application_server ]
-policysvr policy_server_name:port:rank [,...]
-cfg_file fully_qualified_name_of_configuration_file
[ -domain Tivoli_Acccess_Manager_domain ]
```

Note: This action can succeed when there is no configuration file. When the configuration file does not exist, it is created and used as a temporary file to hold configuration information during the operation, and then the file is deleted completely.

–action addsvr

The –action addsvr option adds a policy or authorization server to the application server configuration file.

The configuration file must exist when this action is called.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action addsvr
{ -policysvr policy_server_name |
-authzsvr authorization_server_name }
-cfg_file fully_qualified_name_of_configuration_file
```

–action rmsvr

Removes a policy or authorization server from the application server configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action rmsvr
{ -policysvr policy_server_name |
-authzsvr authorization_server_name }
-cfg_file fully_qualified_name_of_configuration_file
```

–action chgsvr

The –action chgsvr option changes the port or preference ranking of a policy or authorization server in the application server configuration file.

The configuration file must already exist when this action is called.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action chgsvr
{ -policysvr policy_server_name |
-authzsvr authorization_server_name }
-cfg_file fully_qualified_name_of_configuration_file
```

–action replcert

The –action replcert option replaces a certificate in the application server keystore file.

The certificate in the keystore expires based on the certificate lifetime set on the policy server. After the certificate expires, the `-action replcert` option must be used to generate a new certificate. If a certificate becomes compromised, the `-action replcert` option also can be used to invalidate an existing certificate.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action replcert
-admin_id admin_user_ID
-admin_pwd admin_password
-appsvr_id application_server_name
-cfg_file fully_qualified_name_of_configuration_file
```

The configuration file must already exist when this action is called.

–action setport

The –action setport option sets the port on which the application server listens for policy database notifications.

The **`–action setport`** option only updates the application server configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setport
-port port_number
-cfg_file fully_qualified_name_of_configuration_file
```

The configuration file must already exist when this action is called.

–action setdbdir

The –action setdbdir option sets the database directory.

The **`–action setdbdir`** option only updates the application server configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setdbdir
-dbdir name_of_directory_for_local_policy_database
-cfg_file fully_qualified_name_of_configuration_file
```

The configuration file must already exist when this action is called.

–action setdbref

The –action setdbref option sets the database refresh interval, in seconds.

The **`–action setdbref`** option only updates the application server configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setdbref
-dbrefresh refresh_interval_in_seconds
-cfg_file fully_qualified_name_of_configuration_file
```

The configuration file must already exist when this action is called.

–action setdblisten

The –action setdblisten option sets the application listening mode.

The **`–action setdblisten`** option only updates the application server configuration file.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setdblisten
-dblisten { true | false }
-cfg_file fully_qualified_name_of_configuration_file
```

The configuration file must already exist when this action is called.

–action setcertref

The –action setcertref option sets the application certificate refresh mode.

The **–action setcertref** option also indicates whether the application certificate must be renewed automatically. The renewal works only if the certificate lifetime has passed the half life point and has not already expired.

```
java com.tivoli.pd.jcfg.SvrSslCfg -action setcertref
-certrefresh {true|false}
-cfg_file fully_qualified_name_of_configuration_file
```

Appendix B. Deprecated Java classes and methods

For information about the deprecated Java classes and methods, see the Javadoc HTML documentation.

For details about accessing this HTML documentation, see “Accessing the Javadoc HTML documentation” on page 1.

Existing Java applications must be changed to use the indicated replacement class or method.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law :

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Trademarks

Index

Printed in USA

SC23-6516-02