



Sun Java™ System

Application Server 7

Administrator's Guide to Security

2004Q2

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817-5447

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

THIS PRODUCT CONTAINS CONFIDENTIAL INFORMATION AND TRADE SECRETS OF SUN MICROSYSTEMS, INC. USE, DISCLOSURE OR REPRODUCTION IS PROHIBITED WITHOUT THE PRIOR EXPRESS WRITTEN PERMISSION OF SUN MICROSYSTEMS, INC.

U.S. Government Rights - Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

Use is subject to license terms. This distribution may include materials developed by third parties.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, Sun™ ONE, Sun™ ONE Studio, iPlanet, J2EE, J2SE, Enterprise JavaBeans, EJB, JavaServer Pages, JSP, JDBC, JDK, JVM, Java Naming and Directory Interface, JavaMail, and the Java Coffee Cup logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon architecture developed by Sun Microsystems, Inc.

UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd.

This product is covered and controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright © 2004 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. détient les droits de propriété intellectuelle relatifs à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et ce sans limitation, ces droits de propriété intellectuelle peuvent inclure un ou plus des brevets américains listés à l'adresse <http://www.sun.com/patents> et un ou les brevets supplémentaires ou les applications de brevet en attente aux Etats-Unis et dans les autres pays.

CE PRODUIT CONTIENT DES INFORMATIONS CONFIDENTIELLES ET DES SECRETS COMMERCIAUX DE SUN MICROSYSTEMS, INC. SON UTILISATION, SA DIVULGATION ET SA REPRODUCTION SONT INTERDITES SANS L'AUTORISATION EXPRESSE, ECRITE ET PREALABLE DE SUN MICROSYSTEMS, INC.

L'utilisation est soumise aux termes de la Licence. Cette distribution peut comprendre des composants développés par des tierces parties.

Sun, Sun Microsystems, le logo Sun, Java, Solaris, Sun™ ONE, Sun™ ONE Studio, iPlanet, J2EE, J2SE, Enterprise JavaBeans, EJB, JavaServer Pages, JSP, JDBC, JDK, JVM, Java Naming and Directory Interface, JavaMail, et le logo Java Coffee Cup sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Ce produit est soumis à la législation américaine en matière de contrôle des exportations et peut être soumis à la réglementation en vigueur dans d'autres pays dans le domaine des exportations et importations. Les utilisations, ou utilisateurs finaux, pour des armes nucléaires, des missiles, des armes biologiques et chimiques ou du nucléaire maritime, directement ou indirectement, sont strictement interdites. Les exportations ou réexportations vers les pays sous embargo américain, ou vers des entités figurant sur les listes d'exclusion d'exportation américaines, y compris, mais de manière non exhaustive, la liste de personnes qui font objet d'un ordre de ne pas participer, d'une façon directe ou indirecte, aux exportations des produits ou des services qui sont régis par la législation américaine en matière de contrôle des exportations et la liste de ressortissants spécifiquement désignés, sont rigoureusement interdites.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.

Contents

About This Guide	9
Who Should Use This Guide	9
Using the Documentation	10
How This Guide Is Organized	12
Documentation Conventions	13
General Conventions	13
Conventions Referring to Directories	14
Contacting Sun	14
Give Us Feedback	15
Obtain Training	15
Contact Product Support	15
 Chapter 1 Introducing Sun Java System Application Server Security	17
Application Server Security	18
Certificate Administration	18
SSL/TLS Encryption	18
Authentication	19
Auditing	19
HTTP Server Security Features	19
HTTP Server User-Group Authentication	20
HTTP Server Host-IP Authentication	20
HTTP Server SSL Client Authentication	20
HTTP Server Access Control	21
Netscape API (NSAPI)	21
J2EE Application Security Features	22
Declarative Security	22
Programmatic Security	22

User Authentication	23
Realm Administration	23
Single Sign-On	23
Resource Authentication	23
Pluggable Authentication	24
Good Practices	24
Files Associated With Server Security	24
The init.conf File	25
The dbswitch.conf File	25
The server.xml File	26
The obj.conf File	26
The password.conf File	27
The certmap.conf File	27
ACL Files	28
The htaccess Files	28
Keyfile	29
The server.policy File	29
 Chapter 2 General Security Measures	31
About General Security	31
Limiting Physical Access	32
Using Firewalls	33
Single Firewall	33
Double Firewall - DMZ Configuration	34
Triple Firewall - DMZ With Database Protection	35
Limiting Administration Access	36
Managing Passwords	36
Creating Hard-to-Crack Passwords	37
Managing the Superuser Password	37
Changing Passwords or PINs	39
Using the password.conf File	39
Limiting Other Applications on the Server	40
Securing Against an Unprotected Server	41
 Chapter 3 Administering Certificates	43
About Certificates and Authentication	43
Implementing the Trust Database	44
Creating a Trust Database	45
Changing a Trust Database Password	46
Implementing a Certificate	46
Required CA Information	47
Requesting a Certificate	48

Generating a Self-Signed Certificate	50
Installing a Certificate	52
Using the Built-in Root Certificate Module	55
Managing Certificates	55
Managing CRLs and CKLs	56
Installing a CRL or CKL	57
Deleting a CRL or CKL	58
 Chapter 4 Administering SSL/TLS Encryption	59
About Encryption	59
SSL and TLS Protocols	60
Public and Private Keys	61
Task Sequence	61
Enabling SSL Communication with LDAP	62
Turning Security On	62
Turning Security On When Creating as HTTP Listener	63
Turning Security On When Editing an HTTP Listener	63
Enabling SSL and TLS	64
Configuring Security Globally	66
SSL Configuration File Directives	66
SSLCacheEntries	67
SSLClientAuthDataLimit	67
SSLClientAuthTimeout	67
SSLSessionTimeout	67
SSL3SessionTimeout	67
Setting Values for SSL Directives	68
Using External Encryption Modules	68
Installing the PKCS11Module	69
Starting the Server with an External Certificate	69
Enabling FIPS-140 Standard	71
Setting Strong Ciphers	72
Preventing Clients from Caching SSL Files	73
 Chapter 5 Administering HTTP Server Access Control	75
About HTTP Server Access Control	76
HTTP Server User-Group Authentication	77
Basic Authentication	78
SSL Authentication	78
Digest Authentication	80
Host-IP Authentication	81
Access Control List (ACL) Files	81
Client Authentication	82

Implementing Digest Authentication	83
Installing the Digest Authentication Plug-in	84
Digest Authentication on UNIX	84
Digest Authentication on Windows	84
Setting the Sun ONE Directory Server to Use the DES Algorithm	85
Implementing Host-IP Authentication	85
Working With ACL Files	86
ACL File Syntax	87
Type Statement	88
Authentication Statement	89
Authorization Statement	90
Hierarchy of Authorization Statements	91
Attribute Expressions	91
Operators	93
Sample ACL File	93
Writing Customized ACL Expressions	95
Setting Up Client Authentication	96
Setting Client Authentication for the Admin Server	96
Setting Client Authentication for a Server Instance	97
Working with the certmap.conf File	98
Default Properties	99
Creating Custom Properties	101
Sample Mappings	101
ACL/ACE Settings	103
Setting to Allow or Deny	104
Setting for User-Group Authentication	104
Specifying the From Host	106
Setting Access Rights	106
Referencing ACL Files in the obj.conf File	107
Configuring the ACL User Cache	108
ACLCacheLifetime	108
ACLUserCacheSize	108
ACLGroupCacheSize	108
Setting Access Control for a Server Instance	109
Restricting Access to Areas of Your Server	112
Restricting Access to the Entire Server	112
Restricting Access to a Directory (Path)	113
Restricting Access to a URI (Path)	114
Restricting Access to a File Type	115
Restricting Access Based on Time of Day	116
Restricting Access Based on Security	117
Turning Off Access Control	118
Responding When Access is Denied	118

Controlling Access for Virtual Servers	119
Accessing Databases from Virtual Servers	120
Using the dbswitch.conf File	120
Creating a New Authentication Database	121
Specifying Databases in the User Interface	121
Editing Access Control Lists for Virtual Servers	122
Using htaccess Files	123
Enabling htaccess from the User Interface	123
Enabling htaccess from init.conf	124
Using htaccess-register	125
Supported htaccess Directives	126
Index	131

About This Guide

This guide describes how to set up and administer security for the Sun Java™ System Application Server 7 product.

NOTE Not all of the content in this guide is applicable or usable for J2EE applications; some material only applies to HTTP server functionality. For information and instructions on developing secure J2EE applications, refer to the J2EE specifications and the *Sun Java System Application Server Developer's Guide*.

This preface addresses the following topics:

- [Who Should Use This Guide](#)
- [Using the Documentation](#)
- [How This Guide Is Organized](#)
- [Documentation Conventions](#)
- [Contacting Sun](#)

Who Should Use This Guide

This guide is intended for information technology administrators in the corporate enterprise who are familiar with implementing and administering enterprise security mechanisms for servers, including:

- Authentication
- Authorization

- Signing
- Encryption
- Auditing

Using the Documentation

The Sun Java System Application Server Standard and Enterprise Edition manuals are available as online files in Portable Document Format (PDF) and Hypertext Markup Language (HTML).

The following table lists tasks and concepts described in the Sun Java System Application Server manuals. The manuals marked *(updated for 7 2004Q2)* have been updated for the Sun Java System Application Server Standard and Enterprise Edition 7 2004Q2 release. The manuals not marked in this way have not been updated since the version 7 Enterprise Edition release.

Table 1 Sun Java System Application Server Documentation Roadmap

For information about	See the following
<i>(Updated for 7 2004Q2)</i> Late-breaking information about the software and the documentation. Includes a comprehensive, table-based summary of supported hardware, operating system, JDK, and JDBC/RDBMS.	<i>Release Notes</i>
Sun Java System Application Server 7 overview, including the features available with each product edition.	<i>Product Overview</i>
Diagrams and descriptions of server architecture and the benefits of the Sun Java System Application Server architectural approach.	<i>Server Architecture</i>
<i>(Updated for 7 2004Q2)</i> How to get started with the Sun Java System Application Server product. Includes a sample application tutorial. There are two guides, one for Standard Edition and one for Enterprise Edition.	<i>Getting Started Guide</i>
<i>(Updated for 7 2004Q2)</i> Installing the Sun Java System Application Server Standard Edition and Enterprise Edition software and its components, such as sample applications and the Administration interface. For the Enterprise Edition software, instructions are provided for implementing the high-availability configuration.	<i>Installation Guide</i>
<i>(Updated for 7 2004Q2)</i> Evaluating your system needs and enterprise to ensure that you deploy Sun Java System Application Server in a manner that best suits your site. General issues and concerns that you must be aware of when deploying an application server are also discussed.	<i>System Deployment Guide</i>

Table 1 Sun Java System Application Server Documentation Roadmap (*Continued*)

For information about	See the following
Creating and implementing Java™ 2 Platform, Enterprise Edition (J2EE™ platform) applications intended to run on the Sun Java System Application Server that follow the open Java standards model for J2EE components such as servlets, Enterprise JavaBeans™ (EJBs™), and JavaServer Pages™ (JSPs™). Includes general information about application design, developer tools, security, assembly, deployment, debugging, and creating lifecycle modules. A comprehensive Sun Java System Application Server glossary is included.	<i>Developer's Guide</i>
(Updated for 7 2004Q2) Creating and implementing J2EE web applications that follow the Java™ Servlet and JavaServer Pages (JSP) specifications on the Sun Java System Application Server. Discusses web application programming concepts and tasks, and provides sample code, implementation tips, and reference material. Topics include results caching, JSP precompilation, session management, security, deployment, SHTML, and CGI.	<i>Developer's Guide to Web Applications</i>
(Updated for 7 2004Q2) Creating and implementing J2EE applications that follow the open Java standards model for enterprise beans on the Sun Java System Application Server. Discusses Enterprise JavaBeans (EJB) programming concepts and tasks, and provides sample code, implementation tips, and reference material. Topics include container-managed persistence, read-only beans, and the XML and DTD files associated with enterprise beans.	<i>Developer's Guide to Enterprise JavaBeans Technology</i>
(Updated for 7 2004Q2) Creating Application Client Container (ACC) clients that access J2EE applications on the Sun Java System Application Server.	<i>Developer's Guide to Clients</i>
Creating web services in the Sun Java System Application Server environment.	<i>Developer's Guide to Web Services</i>
(Updated for 7 2004Q2) Java™ Database Connectivity (JDBC™), transaction, Java Naming and Directory Interface™ (JNDI), Java™ Message Service (JMS), and JavaMail™ APIs.	<i>Developer's Guide to J2EE Services and APIs</i>
Creating custom NSAPI plug-ins.	<i>Developer's Guide to NSAPI</i>
(Updated for 7 2004Q2) Information and instructions on the configuration, management, and deployment of the Sun Java System Application Server subsystems and components, from both the Administration interface and the command-line interface. Topics include cluster management, the high-availability database, load balancing, and session persistence. A comprehensive Sun Java System Application Server glossary is included.	<i>Administration Guide</i>
(Updated for 7 2004Q2) Editing Sun Java System Application Server configuration files, such as the <code>server.xml</code> file.	<i>Administrator's Configuration File Reference</i>
Configuring and administering security for the Sun Java System Application Server operational environment. Includes information on general security, certificates, and SSL/TLS encryption. HTTP server-based security is also addressed.	<i>Administrator's Guide to Security</i>

Table 1 Sun Java System Application Server Documentation Roadmap (Continued)

For information about	See the following
Configuring and administering service provider implementation for J2EE™ Connector Architecture (CA) connectors for the Sun Java System Application Server. Topics include the Administration Tool, Pooling Monitor, deploying a JCA connector, and sample connectors and sample applications.	<i>J2EE CA Service Provider Implementation Administrator's Guide</i>
(Updated for 7 2004Q2) Migrating your applications to the new Sun Java System Application Server programming model, specifically from iPlanet Application Server 6.x and Sun ONE Application Server 7.0. Includes a sample migration.	<i>Migrating and Redeploying Server Applications Guide</i>
(Updated for 7 2004Q2) How and why to tune your Sun Java System Application Server to improve performance.	<i>Performance Tuning Guide</i>
(Updated for 7 2004Q2) Information on solving Sun Java System Application Server problems.	<i>Troubleshooting Guide</i>
(Updated for 7 2004Q2) Information on solving Sun Java System Application Server error messages.	<i>Error Message Reference</i>
(Updated for 7 2004Q2) Utility commands available with the Sun Java System Application Server; written in manpage style.	<i>Utility Reference Manual</i>
Using the Sun™ Java System Message Queue 3.5 software.	The Sun Java System Message Queue documentation at: http://docs.sun.com/db?p=prod/s1.slmsgqu

How This Guide Is Organized

This guide includes the following components:

- “Introducing Sun Java System Application Server Security” on page 17
- “General Security Measures” on page 31
- “Administering Certificates” on page 43
- “Administering SSL/TLS Encryption” on page 59
- “Administering HTTP Server Access Control” on page 75

Documentation Conventions

This section describes the types of conventions used throughout this guide:

- [General Conventions](#)
- [Conventions Referring to Directories](#)

General Conventions

The following general conventions are used in this guide:

- **File and directory paths** are given in UNIX® format (with forward slashes separating directory names). For Windows versions, the directory paths are the same, except that backslashes are used to separate directories.

- **URLs** are given in the format:

`http://server.domain/path/file.html`

In these URLs, *server* is the server name where applications are run; *domain* is your Internet domain name; *path* is the server's directory structure; and *file* is an individual filename. Italic items in URLs are placeholders.

- **Font conventions** include:
 - The `monospace` font is used for sample code and code listings, API and language elements (such as function names and class names), file names, pathnames, directory names, and HTML tags.
 - *Italic* type is used for code variables.
 - *Italic* type is also used for book titles, emphasis, variables and placeholders, and words used in the literal sense.
 - **Bold** type is used as either a paragraph lead-in or to indicate words used in the literal sense.
- **Installation root directories** for most platforms are indicated by *install_dir* in this document. Exceptions are noted in [“Conventions Referring to Directories” on page 14](#).

By default, the location of *install_dir* on **most** platforms is:

- Solaris and Linux file-based installations:

user's home directory/sun/appserver7

- Windows, all installations:

system drive: \Sun\AppServer7

For the platforms listed above, *default_config_dir* and *install_config_dir* are identical to *install_dir*. See “[Conventions Referring to Directories](#)” on page 14 for exceptions and additional information.

- **Instance root directories** are indicated by *instance_dir* in this document, which is an abbreviation for the following:

default_config_dir/domains/*domain*/*instance*

- **UNIX-specific descriptions** throughout this manual apply to the Linux operating system as well, except where Linux is specifically mentioned.

Conventions Referring to Directories

By default, when using the Solaris package-based or Linux RPM-based installation, the application server files are spread across several root directories. This guide uses the following document conventions to correspond to the various default installation directories provided:

- *install_dir* refers to /opt/SUNWappserver7, which contains the static portion of the installation image. All utilities, executables, and libraries that make up the application server reside in this location.
- *default_config_dir* refers to /var/opt/SUNWappserver7/domains, which is the default location for any domains that are created.
- *install_config_dir* refers to /etc/opt/SUNWappserver7/config, which contains installation-wide configuration information such as licenses and the master list of administrative domains configured for this installation.

Contacting Sun

You might want to contact Sun Microsystems in order to:

- [Give Us Feedback](#)
- [Obtain Training](#)
- [Contact Product Support](#)

Give Us Feedback

If you have general feedback on the product or documentation, please send this to <http://www.sun.com/hwdocs/feedback>

Obtain Training

Application Server training courses are available at:

http://training.sun.com/US/catalog/enterprise/web_application.html/

Visit this site often for new course availability on the Sun Java System Application Server.

Contact Product Support

If you have problems with your system, contact customer support using one of the following mechanisms:

- The online support web site at:
<http://www.sun.com/supporttraining/>
- The telephone dispatch number associated with your maintenance contract

Please have the following information available prior to contacting support. This helps to ensure that our support staff can best assist you in resolving problems:

- Description of the problem, including the situation where the problem occurs and its impact on your operation
- Machine type, operating system version, and product version, including any patches and other software that might be affecting the problem. Here are some of the commonly used commands:
 - **Solaris:** `pkginfo, showrev`
 - **Linux:** `rpm`
 - **All:** `asadmin version --verbose`
- Detailed steps on the methods you have used to reproduce the problem
- Any error logs or core dumps
- Configuration files such as:

- *instance_dir*/config/server.xml
- a web application's web.xml file,
when a web application is involved in the problem
- For an application, whether the problem appears when it is running in a cluster or standalone

Introducing Sun Java System Application Server Security

This section discusses fundamental security concepts and provides an overview of security features and functionality as applied in the Sun Java System Application Server 7 environment.

NOTE	Not all the content in this guide is applicable or usable for J2EE applications; some material only applies to the HTTP server. For information and instructions on developing secure J2EE applications, refer to the J2EE specifications and the <i>Sun Java System Application Server Developer's Guide</i> .
-------------	---

This section addresses the following topics:

- [Application Server Security](#)
- [HTTP Server Security Features](#)
- [J2EE Application Security Features](#)
- [Good Practices](#)
- [Files Associated With Server Security](#)

Application Server Security

As the administrator responsible for server security, you are focussing on protecting the Sun Java System Application Server and its data from unauthorized access, damage (both intentional and unintentional), theft, and misrepresentation. This is done by using a combination of good security practices and a set of security tools, which include such mechanisms as digital certificates, encryption, authorization, and auditing.

In the Sun Java System Application Server environment, your general areas of responsibility include:

- [Certificate Administration](#)
- [SSL/TLS Encryption](#)
- [Authentication](#)
- [Auditing](#)

Certificate Administration

A *certificate* consists of digital data that specifies the name of an individual, company, or other entity, and certifies that the public key included in the certificate belongs to that entity. Both clients and servers can have certificates.

Information on how certificates work in the Sun Java System Application Server environment is contained in [“Administering Certificates” on page 43](#).

SSL/TLS Encryption

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient; *decryption* is the process of transforming encrypted information so that it is intelligible again.

A *cipher* is a cryptographic algorithm (a mathematical function), used for encryption or decryption. The Secure Sockets Layer (SSL) and Transport Layer Security (TLS) protocols supported by the Sun Java System Application Server contain numerous cipher suites. Some ciphers are stronger and more secure than others.

SSL 3.0 and TLS 1.0 encryption protocols are supported. Information on encryption can be found in [“Administering SSL/TLS Encryption” on page 59](#).

Authentication

Authentication is the mechanism by which callers and service providers prove to one another that they are acting on behalf of specific users or systems. When the proof is bidirectional, it is referred to as *mutual authentication*. For example, the user may enter a user name and password in a web browser, and if those credentials match the permanent profile stored in the active database domain, the user is authenticated. The user is associated with this authenticated *security identity* for the remainder of the session.

Server authentication refers to the confident identification of a server by a client; that is, identification of the organization assumed to be responsible for the server at a particular network address.

In *virtual server authentication*, you can have a different certificate database for each virtual server on your system. Each virtual server database can contain multiple certificates. Virtual servers can also have different certificates within each instance.

Auditing

Auditing is the method by which significant events are recorded for subsequent examination, such as errors or security breaches. All authentication events are logged to the Sun Java System Application Server logs. A complete access log provides a sequential trail of Sun Java System Application Server access events.

Information on logging is contained in the *Sun Java System Application Server Administration Guide*.

HTTP Server Security Features

NOTE	The features described as HTTP server features are applicable only to the HTTP server side of the Sun Java System Application Server and not for J2EE applications. In some cases, equivalent functionality is available to J2EE applications as well.
-------------	--

The HTTP server security features include:

- [HTTP Server User-Group Authentication](#)
- [HTTP Server Host-IP Authentication](#)

- [HTTP Server SSL Client Authentication](#)
- [HTTP Server Access Control](#)
- [Netscape API \(NSAPI\)](#)

HTTP Server User-Group Authentication

User-Group authentication requires that users authenticate themselves before access can be granted. This is done by entering a user name and password, using a client certificate, or using the digest authentication plug-in. Types of User-Group authorization supported by the Sun Java System Application Server include Basic, Default, SSL, Digest, and Custom.

For information on HTTP server User-Group authentication, refer to [“HTTP Server User-Group Authentication” on page 77](#) and [“Implementing Host-IP Authentication” on page 85](#).

For information on user-group authentication for J2EE applications, refer to the *Sun Java System Application Server Developer's Guide*.

HTTP Server Host-IP Authentication

Host-IP authentication, also known as Host-IP access control, is a method of limiting access to the Admin Server, or the files and directories on your web site, by making them available only to clients who are using specific computers.

Information on HTTP server Host-IP authentication is contained in [“Implementing Host-IP Authentication” on page 85](#).

HTTP Server SSL Client Authentication

Client authentication refers to authenticating client certificates by cryptographically verifying the certificate signature and the certificate chain leading to the CA on the trust CA list. Clients can have multiple certificates, much like a person might have several different pieces of identification.

Information on HTTP server client authentication is contained in [“Setting Up Client Authentication” on page 96](#).

NOTE SSL client authentication is also available for J2EE applications, as described in the *Sun Java System Application Server Developer's Guide*.

HTTP Server Access Control

By creating a hierarchy of rules called *access control entries* (ACEs) you can allow or deny access to individuals, groups, or other entities such as particular servers or applications. Each ACE specifies whether or not the server should check the next ACE in the hierarchy. The collection of ACEs you create is called an *access control list* (ACL).

There are many options for restricting access to your HTTP server content, among them:

- [Restricting Access to the Entire Server](#)
- [Restricting Access to a Directory \(Path\)](#)
- [Restricting Access to a URI \(Path\)](#)
- [Restricting Access to a File Type](#)
- [Restricting Access Based on Time of Day](#)
- [Restricting Access Based on Security](#)

Information on HTTP server access control is contained in “[Administering HTTP Server Access Control](#)” on page 75.

Netscape API (NSAPI)

A C language API which provides a number of HTTP-centric utility functions, the NSAPI allows plugins to provide Server Application Function (SAF) functions that participate in request processing and other server activities.

Information can be found in the *Sun Java System Application Server Developer's Guide to NASPI*.

J2EE Application Security Features

The J2EE application authentication and authorization requirements are defined by the J2EE specification and are briefly listed here.

NOTE For developing J2EE application security, use the security mechanisms as described in the J2EE specifications and the *Sun Java System Application Server Developer's Guide*.

The following J2EE security features are supported in the Sun Java System Application Server environment:

- [Declarative Security](#)
- [Programmatic Security](#)
- [User Authentication](#)
- [Realm Administration](#)
- [Single Sign-On](#)
- [Resource Authentication](#)
- [Pluggable Authentication](#)

Declarative Security

In declarative security, authorization is handled by the container. Deployment descriptors are referenced to determine whether the principal associated with the current security context is permitted access to the requested operation.

Web applications may also specify transport guarantee requirements of confidentiality or integrity. This translates to requiring SSL for such resources.

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

Programmatic Security

In programmatic security, authorization is handled by the application code directly. This code is written by the developer.

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

User Authentication

Three caller authentication paths exist: web client, J2EE application client (running in the application container), and external RMI/IIOP clients that do not use the Sun Java System Application Server container. Form authentication is supported.

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

Realm Administration

The Administration interface provides the capability to add/edit/delete supported realms from the server. Realms included in the Sun Java System Application Server are `file`, `ldap`, `certificate`, and `solaris`.

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

Single Sign-On

For single sign-on, a user's authentication state can be shared across multiple J2EE applications in a single virtual server instance.

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

Resource Authentication

The Sun Java System Application Server supports authentication into external resources (which may require separate authentication).

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

Pluggable Authentication

Pluggable authentication allows for J2EE applications to use the Java Authentication and Authorization Service (JAAS) feature from the J2SE platform. Developers can plug in their own authentication mechanisms.

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

Good Practices

There are many precautions you can take to protect your Sun Java System Application Server resources. Some involve mechanisms (such as authentication or encryption) and many are simply based on security-aware operational practices and common sense.

Some good practices include:

- Limiting physical access to the Sun Java System Application Server
- Setting up firewalls
- Limiting access to administration capabilities
- Managing passwords
- Limiting the presence of other applications on the server
- Configuring for protected and unprotected servers

These topics are discussed in [“General Security Measures” on page 31](#).

Files Associated With Server Security

Many of the Sun Java System Application Server configuration files are used to define security parameters for the server. The main security-related tasks associated with each file are listed briefly in the following sections:

- [The init.conf File](#)
- [The dbswitch.conf File](#)
- [The server.xml File](#)
- [The password.conf File](#)

- [The certmap.conf File](#)
- [ACL Files](#)
- [The htaccess Files](#)
- [Keyfile](#)
- [The server.policy File](#)

Details on the contents of the Sun Java System Application Server configuration files are contained in the *Sun Java System Application Server Administrator's Configuration File Reference*.

The init.conf File

The `init.conf` file contains low-level server configuration information, such as the path the server is installed to, performance tuning options, location of plugin shared objects, and so on. This is the startup file. When the Sun Java System Application Server starts up, it looks in this file to establish a set of global variable settings that affect the server instance's behavior and configuration.

Security-related tasks include:

- To use the `chroot` command to secure an unprotected server, all paths in `init.conf` must be absolute; paths in `obj.conf` must be relative to the `chroot` directory. Refer to [“Securing Against an Unprotected Server” on page 41](#) for guidelines.
- Installing an SSL-enabled server creates SSL directive entries in the `init.conf` file for global security parameters. Refer to [“Configuring Security Globally” on page 66](#) for instructions.
- You can control the ACL user cache by setting directives in the `init.conf` file. Refer to [“Configuring the ACL User Cache” on page 108](#) for further information.
- To manually enable your server to use the `htaccess` file, you need to first modify the server's `init.conf` file to load, initialize, and activate the plug-in. Refer to [“Enabling htaccess from init.conf” on page 124](#) for instructions.

The dbswitch.conf File

NOTE This section only applies to HTTP server content.

The `dbswitch.conf` file specifies the LDAP directory used by the Sun Java System Application Server. It is only read at server startup.

You can globally define user authentication databases in the `dbswitch.conf` file. Refer to [“Accessing Databases from Virtual Servers” on page 120](#) for further information.

The server.xml File

The `server.xml` file is the main configuration file for the Sun Java System Application Server. Security-related tasks include:

- Handles configuration of HTTP listeners (including SSL ciphers, certificates, and so on), virtual servers, access control lists, and so on. Handles the relationships between these entities.
- Contains a list of security domains with configuration data for the provider class and realm-specific properties. Refer to the *Sun Java System Application Server Developer's Guide* for more information on security domains (realms).
- SSL properties for virtual servers can be found on a per-server basis in the `ssl` element of the `server.xml` file. Refer to [“Configuring Security Globally” on page 66](#) for more information.
- By manually editing the `server.xml` file, you can tell the Sun Java System Application Server to start with an external server certificate. Refer to [“Starting the Server with an External Certificate” on page 69](#) for more information.

Further information on the `server.xml` file can be found in the *Sun Java System Application Server Administrator's Configuration File Reference*.

The obj.conf File

NOTE	This section only applies to HTTP server content.
-------------	---

The `obj.conf` file contains directives that instruct the Sun Java System Application Server on how to handle requests from clients. Security-related tasks include:

- HTTP server authentication uses the default method you specify in the `obj.conf` file, or Basic if there is no setting in the `obj.conf` file. For more information, refer to [“HTTP Server User-Group Authentication” on page 77](#).

- If you have named ACLs or created separate ACL files, you can reference them in the `obj.conf` file. For further information on doing this, refer to [“Referencing ACL Files in the obj.conf File” on page 107](#).
- Handles configuration of the NSAPI request processing path (each virtual server may have its own `obj.conf` file). Further information is contained in the *Sun Java System Application Server Developer’s Guide to NASPI*.

The password.conf File

If you want an SSL/TLS-enabled Sun Java System Application Server to be able to restart unattended when configured for SSL, you can save the trust database password in a `password.conf` file.

NOTE Be sure that your system is adequately protected so that this file and the key databases are not compromised. Such protection is discussed in [“Limiting Physical Access” on page 32](#).

Further information on the `password.conf` file can be found in [“Using the password.conf File” on page 39](#) and the *Sun Java System Application Server Administrator’s Configuration File Reference*.

The certmap.conf File

NOTE This section only applies to HTTP server content.

The `certmap.conf` file specifies how a certificate, designated by name, is mapped to an LDAP entry, designated by issuerDN. The `certmap.conf` file provides the following information:

- Where in the LDAP tree the server should begin its search
- What certificate attributes the Sun Java System Application Server should use as search criteria when searching for the entry in the LDAP directory
- Whether or not the server goes through an additional verification process

Refer to [“Working with the certmap.conf File” on page 98](#) for further information.

ACL Files

Access control list (ACL) files are text files that contain lists identifying who can access the resources stored on your Sun Java System Application Server.

NOTE	The access control methods described in this document should not be used for J2EE application development. Using these methods, especially ACLs, could cause your applications to work unpredictably and be inconsistent with the J2EE model. For application development, use the J2EE security mechanisms as described in the J2EE specifications and the <i>Sun Java System Application Server Developer's Guide</i> .
-------------	---

By default, the Sun Java System Application Server uses a single ACL file that contains all the lists for accessing your server. As an alternative to this default, you can create multiple ACL files and reference them in the `obj.conf` file.

Information on working with ACL files is contained in [“Administering HTTP Server Access Control” on page 75](#). Additional information can be found in the *Sun Java System Application Server Developer's Guide to NASPI*.

The htaccess Files

NOTE	This section only applies to HTTP server content.
-------------	---

The `htaccess` files are dynamic configuration files that store a subset of configuration options. You can use `htaccess` files in combination with the Sun Java System Application Server standard access controls (standard access controls are always applied before any `htaccess` access controls).

Information on working with `htaccess` files is contained in [“Using htaccess Files” on page 123](#).

Keyfile

The keyfile contains the list of users for the `file` realm (applicable only for J2EE applications). Every server instance has a default keyfile which is empty. Users can be added through the Administration interface or the command-line interface.

By default, the `file` realm is always set to use this file, with the name `keyfile`. However, the name and location of this file can be changed by editing the `file` realm properties in the `server.xml` file.

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

The server.policy File

The `server.policy` file contains the J2SE policy configuration which will be in effect for all the Java code running in an instance.

Refer to the *Sun Java System Application Server Developer's Guide* for more information.

General Security Measures

In addition to using security mechanisms such as authentication, encryption, and ACL files, as well as taking advantage of the J2EE authentication and authorization mechanisms, there are a number of manual steps you can take to make your Sun Java System Application Server more secure.

This section contains the following topics:

- [About General Security](#)
- [Limiting Physical Access](#)
- [Using Firewalls](#)
- [Limiting Administration Access](#)
- [Managing Passwords](#)
- [Limiting Other Applications on the Server](#)
- [Securing Against an Unprotected Server](#)

About General Security

Networks face risks from external and internal attackers, who use a variety of tactics to gain access to your server and the information on it. The Sun Java System Application Server offers secure connections between the server and the client. However, it can't control the security of information once the client has the information, nor can it control access to the server machine itself and its directories and files.

Being aware of these limitations helps you understand what situations to avoid. For example, you might acquire credit card numbers over an SSL connection, but are those numbers stored in a secure file on the server machine? What happens to those numbers after the SSL connection is terminated? You are responsible for securing any information clients send to you through SSL.

Limiting Physical Access

The simple security measure of physically protecting your server from access is often overlooked. Keep the server machine in a locked room that only authorized people can enter. This prevents anyone from attacking the server machine itself.

- **Root password**—It is crucial to protect your machine's administrative (root) password. As with all passwords, but particularly the root password, be sure to select passwords that are difficult to guess.
- **Application server configuration**—Some of the Sun Java System Application Server configuration files (such as `server.xml`, various J2EE application descriptor XML files, `password.conf`) may contain cleartext passwords for a number of external resources (such as JDBC database passwords and SSL databases) that the application server needs to authenticate to during its operation. All such configuration files must be carefully protected.

By default all the configuration files in the `/config` directory as well as the `/application` directory are only readable by the instance owner. To protect the password data, it is important to maintain these directories with these default restricted-access permissions.

- **Backup tapes**—You must protect your backup tapes as vigilantly as you protect data on the server.

NOTE Because some of configuration files may contain cleartext passwords, when the Sun Java System Application Server file systems are backed up these passwords will, if present, be contained in these backups, and can be retrieved and potentially abused by anyone who has access to the backup media.

- **Ports**—Disable any ports not used on the machine. Use routers or firewall configurations to prevent incoming connections to anything other than the absolute minimum set of ports. This means that the only way to get a shell is to physically use the server's machine, which should already be in a restricted area.

- Security hardening of the operating system—This is recommended and should be considered a requirement on any production system that is accessible through the Internet.

Operating system hardening is very platform-specific and cannot be covered in any detail in this documentation. Please consult your platform vendor. See, for example, the JASS Toolkit for Solaris

<http://www.sun.com/software/security/jass/>

The Sun Java System Application Server assumes that those who have access to the physical server will not abuse the server channel. It is very important that you take whatever precautions you can to restrict server access only to authorized, well intentioned users.

Using Firewalls

This section examines some common firewall configurations and the parameters to be configured for correct functioning. This is general information that pertains to the Sun Java System Application Server. For details, refer to the documentation from your particular firewall vendor.

The following topics are addressed in this section:

- [Single Firewall](#)
- [Double Firewall - DMZ Configuration](#)
- [Triple Firewall - DMZ With Database Protection](#)

Single Firewall

The simplest and most common firewall configuration places a single firewall between the Sun Java System Application Server server and the Internet browser. The firewall needs to be configured to allow HTTP connections to the HTTP port (default 80) and/or the HTTPS port (default 443) as appropriate for web container access.

NOTE If direct RMI/IIOP access to EJBs is allowed from the Internet, the IIOP/RMI listener port (default 3700) must be opened as well. However, this practice is strongly discouraged as it creates potential security risks.

The advantage of the single firewall configuration is simplicity. The biggest disadvantage is that there is a single line of defense. If the firewall is breached, the only defense is the security of each of the individual machines within the private network.

The following table summarizes the protocols and ports which need to be configured to the firewall to permit correct functioning. The left column indicates the protocol used, the center column indicates the port, and the right column indicates the type of communication.

Table 2-1 Double Firewall Protocols and Ports

Protocol	Wall	Port	Reason
TCP/IP	Outer	80 (default)	HTTP requests
TCP/IP	Outer	443	HTTPS requests

For information on these ports, refer to the Sun Java System Application Server *Administrator's Guide* and the Administration interface online help.

Double Firewall - DMZ Configuration

The double firewall, also known as the demilitarized zone (DMZ) configuration, is becoming more common as many organizations allow limited access to their private networks to partners and customers. The double layer of protection combined with active monitoring of activity at each firewall and within the DMZ will detect most attempts to penetrate the internal network. The security provided is much better than the single firewall configuration.

The double firewall configuration provides:

- An outer firewall between the Internet browser and a routing web server/application server located in the DMZ
- An inner firewall between the routing server in the DMZ and the protected Sun Java System Application Server
- A proxy plugin that forwards requests to the Sun Java System Application Server behind the second firewall

In the double firewall configuration, the outer firewall must be configured to allow for HTTP and HTTPS transactions. The inner firewall must be configured to allow the HTTP server plug in to communicate with the Sun Java System Application Server server(s) behind the firewall.

The following table summarizes the protocols and ports which need to be configured to the firewall to permit correct functioning. The left column indicates the protocol used, the second column indicates which firewall the protocol/port applies to, the third column indicates the port, and the right column indicates the type of communication.

Table 2-2 Single Firewall Protocols and Ports

Protocol	Wall	Default Port	Reason
TCP/IP	Outer	80	HTTP requests to routing server
TCP/IP	Outer	443	HTTPS requests to routing server
TCP/IP	Inner	80	HTTP requests to Sun Java System Application Server
TCP/IP	Inner	443	HTTPS requests to Sun Java System Application Server

For information on these ports, refer to the Sun Java System Application Server *Administrator's Guide* and the Administration interface online help.

Triple Firewall - DMZ With Database Protection

In some corporate settings, databases reside on their own networks, protected by firewalls. The triple firewall configuration provides maximum security for what may be the most important corporate asset: the data contained within the corporate database. The firewall between the LAN and the database systems provides protection against internal as well as external threats.

The connections to the database use standard access mechanisms such as Open DataBase Connectivity (ODBC), Java DataBase Connectivity (JDBC), and database vendor-supplied connector libraries. The connections to the database are not different from any other application, so the firewall configuration for the database protection layer will conform to the standard settings required for access to the specific database in use.

Limiting Administration Access

If you use remote configuration, be sure to set access control to allow administration access by only a few users and computers.

You should always turn on encryption for the master Admin Server. If you don't use an SSL connection for administration, you must be very cautious when performing remote server administration over the unsecure network. Anyone can intercept your administrative password and reconfigure your servers.

If you want your Admin Server to provide end-user access to the LDAP server or local directory information, consider maintaining two Admin Servers and using cluster management, so that the SSL-enabled Admin Server acts as the master server, and the other Admin Server is available for end-user access. Refer to [“Enabling SSL Communication with LDAP” on page 62](#) for further information.

Instructions for implementing cluster management can be found in the Sun clustering documentation collection.

Managing Passwords

There are a number of passwords with your server: the administrative password, the private key password, database passwords, and so on. Your administrative password is the most important password of all, since anyone with that password can configure any and all servers on your computer. Your private key password is next most important. If someone gets your private key and your private key password, they can create a fake server that appears to be yours, or intercept and change communications to and from your server.

A good password is one you'll remember but others won't guess. For example, you might remember *MCi12!mo* as “My Child is 12 months old!” A bad password is your child's name or birth date.

The following topics address additional information on passwords:

- [Creating Hard-to-Crack Passwords](#)
- [Managing the Superuser Password](#)
- [Changing Passwords or PINs](#)
- [Using the password.conf File](#)

Creating Hard-to-Crack Passwords

There are some simple guidelines that will help you create a stronger password.

It is not necessary to incorporate all the following guidelines in one password, but the more of these guidelines you use, the harder your password will be to crack:

- Passwords should be 6-14 characters long (observing whatever length limitation your system has).
- Do not use the illegal characters: *, ", or spaces
- Do not use dictionary words (any language)
- Do not make common letter substitutions, like replacing E with 3, or L with 1
- Do include characters from as many of these classes as possible:
 - Uppercase letters
 - Lowercase letters
 - Numbers
 - Symbols

Managing the Superuser Password

You can configure superuser access for your Admin Server. In this case, the superuser is the person who has access to change any or all parts of the server configuration (not to be confused with the system's superuser/root). These settings affect only the superuser account. That is, if your Admin Server uses distributed administration, you need to set up additional access controls for the administrators you enable.

The superuser's user name and password are kept in a file called *intsall_dir*/domains/*domain_dir*/admin-server/config/admpw. If you forget the user name, you can view this file to obtain the actual name; however, the password is encrypted and unreadable. The file has the format `username:password`.

If you forget the password, you can edit the `admpw` file and simply delete the encrypted password.

CAUTION Because you can edit the `admpw` file, it is very important that you keep the server machine in a secure place and restrict access to its file system:

- On UNIX/Linux systems, consider changing the file ownership so that it is writable only by root or whatever system user runs the Admin Server daemon. By default, the `/config` directory is readable only by the instance owner which protects the directory and other sensitive files. Be sure these permissions are not altered.
 - On Windows systems, restrict the file ownership to the user account the Admin Server uses.
-

To set up superuser access for the Admin Server, perform the following steps in the Administration interface:

1. Access Admin Server and select Security.
2. Select Access Control.

The Superuser Access Control page is displayed.

3. Enter the host names that are allowed superuser access to the Admin Server.
4. Enter the IP addresses that are allowed superuser access to the Admin Server.
5. Enter the authentication user name.
6. Enter the authentication password.

For a list of guidelines to consider when changing a password, see [“Creating Hard-to-Crack Passwords” on page 37](#).

7. Re-enter the authentication password.
8. Click OK.
9. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
10. Stop and start the server for changes to take effect.

Changing Passwords or PINs

It's good practice to change your trust database/key pair file password or PIN periodically. If your Admin Server is SSL-enabled, this password is required when starting the server. You should only change this password on your local machine. Refer to [“Changing a Trust Database Password” on page 46](#) for instructions.

It is important to make sure your key-pair file is protected. The Admin Server stores key-pair files in the `/config` directory of the instance. By default, the `/config` directory files are set to be readable only by the instance owner. It is important to monitor these permissions to ensure that the file permissions aren't inadvertently changed later by backup scripts or other events.

It's also important to know if the file is stored on backup tapes or is otherwise available for someone to intercept. If so, you must protect your backup tapes as carefully as you protect your server data.

Using the password.conf File

By default, the Sun Java System Application Server prompts you for the SSL key database password at startup. If you want to be able to restart an unattended Sun Java System Application Server, you must save the password in a `password.conf` file.

NOTE	Only use the <code>password.conf</code> file if your system is adequately protected so that this file and the key databases cannot be compromised.
-------------	--

- For UNIX—Normally, you cannot start a UNIX SSL-enabled server with the `/etc/rc.local` or the `/etc/inittab` files because the server requires a password before starting. Although you can start an SSL-enabled server automatically if you keep the password in plain text in a file, this is not recommended. The server's `password.conf` file should be owned by root or the user who installed the server, with only the owner having read and write access.

NOTE	Leaving the SSL-enabled server's password in the <code>password.conf</code> file is a large security risk. Anyone who can access the file has access to the SSL-enabled server's password. Consider the security risks before keeping the SSL-enabled server's password in the <code>password.conf</code> file.
-------------	---

- For Windows—If you have a New Technology File System (NTFS) file system, you should protect the directory that contains the `password.conf` file by restricting its access, even if you do not use the file. The directory should have read/write permissions for the Admin Server user and the Sun Java System Application Server user. Protecting the directory prevents others from creating a false `password.conf` file.

NOTE On Windows, you cannot protect directories or files on File Allocation Table (FAT) file systems by restricting access to them.

If security risks are not a concern for you, follow these steps to start your SSL-enabled server automatically:

1. Make sure SSL is on.
2. Create a new `password.conf` file in the `config` subdirectory of the server instance.
 - If you are using the internal PKCS11 software encryption module that comes with the server, enter the following information:

`internal:your_password`
 - If you are using a different PKCS11 module (for hardware encryption or hardware accelerators), specify the name of the PKCS11 module, followed with the password. For example:

`nFast:your_password`
3. Stop and start your server for the new setting to take effect.

Limiting Other Applications on the Server

It is possible to circumvent Sun Java System Application Server security by exploiting weaknesses in other programs running on your server. An obvious prevention step is to disable any unnecessary programs and services running on your server.

- UNIX—Carefully choose the processes that are started from the `inittab` script and the `rc` script.
 - Don't run `telnet` or `rlogin` from the server machine.

- Don't have `rdist` on the server machine. While the purpose of `rdist` is to distribute files, it can also be used by an attacker to update files on the server machine illegally.
- Windows—Be selective about which drives and directories you share with other machines. Also, carefully choose which users have accounts or Guest privileges.

Be careful about what programs you or other people install on your server. Other people's programs might have security holes that they may or may not be aware of. Worst of all, someone might install a malicious program designed specifically to subvert your security. Always examine programs carefully before you allow them on your server.

Securing Against an Unprotected Server

If you want to have both protected and unprotected servers, you should operate the unprotected server on a different machine from the protected one.

If your resources are limited and you must run an unprotected server on the same machine as your protected server, do the following:

- Different port numbers—Make sure that the protected server and the unprotected server are assigned different port numbers. The registered default port numbers are:
 - 443 for the protected server
 - 80 for the unprotected server
- For UNIX—Redirect the document root directory using the `chroot` tool.

The UNIX `chroot` command allows you to create a second root directory to limit the server to specific directories. Refer to the man page for guidelines on using this command.

In the Administration interface, you can specify the `chroot` directory for a specific virtual server by performing the following steps:

1. Access App Server Instances and select the server instance from the left pane.
2. Select HTTP Server and Virtual Servers.
3. Select the virtual server you want to specify the `chroot` directory for.

The page for the General tab is displayed.

4. Scroll down the page until you find the Chroot field.
5. Enter the full pathname in the Chroot directory.
6. Click Save.
7. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
8. Stop and start the server for changes to take effect.

Administering Certificates

This section describes how to set up and administer the trust database, certificates, and certificate-related lists for your Sun Java System Application Server 7 environment.

This section addresses the following topics:

- [About Certificates and Authentication](#)
- [Implementing the Trust Database](#)
- [Implementing a Certificate](#)
- [Using the Built-in Root Certificate Module](#)
- [Managing Certificates](#)
- [Managing CRLs and CKLs](#)

About Certificates and Authentication

Authentication is the process of confirming an identity. In the context of network interactions, authentication is the confident identification of one party by another party. Certificates are one way of supporting authentication.

A *certificate* consists of digital data that specifies the name of an individual, company, or other entity, and certifies that the public key, included in the certificate, belongs to that entity. Both clients and servers can have certificates.

A certificate is issued and digitally signed by a *Certificate Authority (CA)*. The CA can be a company that sells certificates over the Internet, or it can be a department responsible for issuing certificates for your company's intranet or extranet. You decide which CAs you trust enough to serve as verifiers of other people's identities.

In addition to a public key and the name of the entity identified by the certificate, a certificate also includes an expiration date, the name of the CA that issued the certificate, and the digital signature of the issuing CA. For more information regarding the content and format of a certificate, see *Introduction to SSL* at the following location:

<http://docs.sun.com/db/prod/3802#hic>

The task sequence for setting up basic security is:

1. Create the trust database.

Refer to “[Creating a Trust Database](#)” on page 45.

2. Request a certificate.

Refer to “[Requesting a Certificate](#)” on page 48.

3. Install the certificate.

Refer to “[Installing a Certificate](#)” on page 52.

4. Activate encryption.

Refer to “[Administering SSL/TLS Encryption](#)” on page 59.

Other administrative tasks associated with certificates are discussed in “[Managing Certificates](#)” on page 55 and “[Managing CRLs and CKLs](#)” on page 56.

Implementing the Trust Database

In the Sun Java System Application Server, the Admin Server and each server instance has its own certificate and key-pair file, referred to as a *trust database*.

NOTE	Before requesting a server certificate, you must create a trust database for identifying your trusted entities.
-------------	---

In the trust database you create and store the public and private keys, referred to as your *key-pair file*. The key-pair file is used for SSL encryption. You will use the key-pair file when you request and install your server certificate, which is stored in the trust database after installation. The key-pair file is stored encrypted in the `/config` directory of the instance.

The Admin Server has only one trust database, while each server instance can have its own trust database. The certificate and key-pair database files are named after the server instance that uses them. Virtual servers are covered by the trust database created for their server instance.

As the administrator, you manage the trust database and its constituent certificates, including the server certificate and all the included CAs.

This section addresses the following:

- [Creating a Trust Database](#)
- [Changing a Trust Database Password](#)

Creating a Trust Database

When you create the trust database, you specify a password that will be used for a key-pair file. You will also need this password to start a server using encrypted communications.

To create a trust database on your local machine, perform the following steps in the Administration interface:

1. Access App Server Instances and select the server instance.
2. Access Security.
3. Click Manage Database.
4. Click the Create Database link.

The Initialize Trust Database page is displayed.

5. Enter a password for the database.
6. Repeat the password.
7. Click OK.
8. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
9. Stop and restart the server for changes to take effect.

Changing a Trust Database Password

To change an existing trust database password, perform the following steps in the Administration interface:

1. Access App Server Instances and select the server instance.
2. Access Security.
3. Click Manage Database.
4. Click the Change Password link.

The Change the Key Pair File Password page is displayed.

5. Select the cryptographic module from the dropdown list.
6. Enter the old password.
7. Enter the new password.
8. Repeat the new password.
9. Click OK.
10. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
11. Stop and restart the server for changes to take effect.

Implementing a Certificate

After creating a trust database for your server, you can request a certificate and submit it to a CA. If your company has its own internal CA, request your certificate from them. If you plan to purchase your certificate from a commercial CA, choose a CA and ask about the specific format of the information they require.

The Admin Server can have only one server certificate, while a server instance can have its own server certificate. You can select a server instance certificate for each virtual server.

The following topics address implementing certificates:

- [Required CA Information](#)
- [Requesting a Certificate](#)
- [Generating a Self-Signed Certificate](#)

- [Installing a Certificate](#)

Required CA Information

Before you begin the request process, make sure you know what information your CA requires. Whether you are requesting a server certificate from a commercial CA or an internal CA, you need to provide the following information:

- **Common Name**—The fully qualified hostname used in DNS lookups (for example, `www.sun.com`). This is the hostname in the URL that a browser uses to connect to your site. If these two names don't match, a client is notified that the certificate name doesn't match the site name, creating doubt about the authenticity of your certificate. Some CAs might have different requirements, so it's important to check with them.

You can also enter wildcard and regular expressions in this field if you are requesting a certificate from an internal CA. However, most vendors will not approve a certificate request with a wildcard or regular expression entered for the common name.

- **email Address**—Your business email address. This is used for correspondence between you and the CA.
- **Organization**—The official, legal name of your company, educational institution, partnership, and so on. Most CAs require that you verify this information with legal documents (such as a copy of your business license).
- **Organizational Unit**—An optional field that describes an organization within your company. This can also be used to specify a less formal company name (without the Inc., Corp., and so on).
- **Locality**—An optional field that usually describes the city, principality, or country for the organization.
- **State or Province**—Usually a required field, but can be optional for some CAs. Most CAs won't accept abbreviations, but check with them to be sure.
- **Country**—A required field that contains the two-character abbreviation of your country name, in ISO format. The country code for the United States is US.

All this information is combined as a series of attribute-value pairs called the *distinguished name (DN)*, which uniquely identifies the subject of the certificate.

If you are purchasing your certificate from a commercial CA, you must contact the CA to find out what additional information is required before they will issue a certificate. Most CAs require that you prove your identity. For example, they want to verify your company name and who is authorized by the company to administer the server. They might also ask whether you have the legal right to use the information you provide.

Some commercial CAs offer certificates with greater detail and veracity to organizations or individuals who provide more thorough identification. For example, you might be able to purchase a certificate stating that the CA has not only verified that you are the rightful administrator of the `www.your_company.com` computer, but that you are a company that has been in business for three years, and has no outstanding customer litigation.

Requesting a Certificate

After you have created your trust database, you are ready to request a certificate.

To request a certificate from a CA, perform the following steps in the Administration interface:

1. Access App Server Instances and select the server instance in the left pane.
2. Access Security.
3. Select Certificate Management.
4. Click the Request link.

The Request a Server Certificate page is displayed.

5. Select if this is a new certificate or a certificate renewal.

Many certificates expire after a set period of time, such as six months or a year. Some CAs will automatically send you a renewal.

6. Perform the following steps to specify how you want to submit the request for the certificate:
 - If the CA expects to receive the request in an email message, check CA Email and enter the email address of the CA. For a list of CAs, click List of available certificate authorities.
 - If you are requesting the certificate from an internal CA that is using the Sun Java System Certificate Server, click CA URL and enter the URL for the Certificate Server. This URL should point to the certificate server's program that handles certificate requests.

7. Select the cryptographic module for the key-pair file you want to use when requesting the certificate from the drop-down list.

8. Enter the password for your key-pair file.

This is the password you specified when you created the trust database, unless you selected a cryptographic module other than the internal module. The server uses the password to get your private key and encrypt a message to the CA. The server then sends both your public key and the encrypted message to the CA. The CA uses the public key to decrypt your message.

9. Enter your identification information.

The format of this information varies by CA.

10. Verify your work to ensure accuracy.

The more accurate the information, the faster your certificate is likely to be approved. If your request is going to a certificate server, you'll be prompted to verify the form information before the request is submitted.

11. Click OK.

12. Access App Server Instances and your server instance in the left pane, then click Apply Changes.

13. Stop and restart the server for changes to take effect.

The server generates a certificate request that contains your information. The request has a digital signature created with your private key. The CA uses a digital signature to verify that the request wasn't tampered with during routing from your server machine to the CA. In the rare event that the request is tampered with, the CA will usually contact you by phone.

If you choose to email the request, the server composes an email message containing the request and sends the message to the CA. Typically, the certificate is then returned to you using email. If instead you specified a URL to a certificate server, your server uses the URL to submit the request to the certificate server. You might get a response using email or other means, depending on the CA.

The CA will notify you if it agrees to issue you a certificate. In most cases, the CA will send your certificate using email. If your organization is using a certificate server, you may be able to search for the certificate by using the certificate server's forms.

NOTE Not everyone who requests a certificate from a commercial CA is given one. Many CAs require you to prove your identity before issuing you a certificate. Also, it can take anywhere from one day to two months to get approval. You are responsible for promptly providing all the necessary information to the CA.

Once you receive the certificate, you can install it. In the meantime, you can still use your server without encryption.

Generating a Self-Signed Certificate

Use `certutil` to generate self-signed certificates. It is available as an add-on to the Sun Java System Application Server at:

http://www.sun.com/software/download/app_servers.html

For information on `certutil`, visit the following URL:

<http://www.mozilla.org/projects/security/pki/nss/tools/certutil.html>

NOTE You can not use `keytool` to generate certificates with Sun Java System Application Server.

Certificates generated with `keytool` are not compatible with Sun Java System Application Server.

To setup self-signed certificate using `certutil` from Sun ONE Web Server 6.0:

1. Set the environment to run `certutil`:

```
setenv LD_LIBRARY_PATH install_dir/bin/https/lib
setenv PATH install_dir/bin/https/admin/bin
```

2. Run certutil.

Create the key database using -N option:

For admin certificate:

```
- certutil -N -d certdir -P https-admserv-hostname-
```

For server certificate:

```
- certutil -N -d certdir -P https-hostname+domain-hostname-
```

3. Enter a new password that is minimum 8 characters long.

Note: You can create the key database using the Administration GUI by selecting Security -> Create Database option.

4. To generate the self-signed certificate, use the -s option:

```
install_dir/alias/certutil -S -n "selfsignedcert" -x -m 1 -d . -v 57 -1 -2 -5  
-t "CTu,CTu,CTu" -s "CN="Self-signed certificate", O="mycompany.com",  
C="US" -P "https-admserv-scholar-"
```

5. You will be prompted to enter password for "NSS Certificate DB". Enter the password.

The following message appears on the console:

Generating key. This may take a few moments...

```
0 - Digital Signature  
1 - Non-repudiation  
2 - Key encipherment  
3 - Data encipherment  
4 - Key agreement  
5 - Cert signing key  
6 - CRLsigning key  
Other to finish
```

6. Select 0, 5, 9. The following messages will be displayed:

```
Is this a critical extension [y/n]?
Enter Y
Is this a CA certificate [y/n]
Enter Y
Enter the path length constraint, enter to skip[<0 for unlimited path]:
Press Enter
Is this a critical extension [y/n]

0 - SSL Client
1 - SSL Server
2 - S/MIME
3 - Object Signing
4- Reserved for future use
5 - SSL CA
6 - S/MIME CA
7 - Object Signing CA
Other to finish
```

7. Select: 1, 5, 9. The following inputs will be asked:

```
Is this a critical extension [y/n]?
Y
```

8. Make sure that the self-signed certificate is generated. In the Administration Console, choose Admin Server -> Security -> Manage Certificates. Next, choose, Instance Server -> Security -> Manage Certificates.

If the certificate generation is successful, it displays “selfsignedcert”.

9. Run the Web server in secure mode using “self-signed cert”.

Select Admin Server -> Preferences -> Edit Listen Sockets -> Turn on Security -> Click Attributes. Select “selfsignedcert” under CertificateNickName, restart the admin server.

Select Instance Server -> Preferences -> Edit Listen Sockets -> Turn on Security -> Click Attributes. Select “selfsignedcert” under CertificateNickName. Apply change and restart the instance server.

Installing a Certificate

When you receive your certificate back from the CA, it will be encrypted with your public key so that only you can decrypt it. After you enter the correct password for your trust database, you will be able to decrypt and install your certificate.

There are three types of certificates:

- Your own server's certificate to present to clients
- A CA's own certificate for use in a certificate chain

A *certificate chain* is a hierarchical series of certificates signed by successive certificate authorities. A CA certificate identifies a CA and is used to sign certificates issued by that authority. A CA certificate can in turn be signed by the CA certificate of a parent CA, and so on, up to a root CA.

- A trusted CA's certificate

NOTE If your CA doesn't automatically send you their certificate, you should request it. Many CAs include their certificate in the email with your certificate, and your server installs both certificates at the same time.

The server will use the key-pair file password you specify to decrypt the certificate when you install it. You can either save the email somewhere accessible to the server, or copy the text of the email and be ready to paste the text into the Install Certificate form, as described here.

To install a certificate from a CA, perform the following steps in the Administration interface:

1. Access App Server Instances and select the server instance in the left pane.
2. Access Security.
3. Select Certificate Management.
4. Click the Install link.

The Install a Server Certificate is displayed.

5. Install a Server Certificate Page

Select the type of certificate you are installing:

- This Server—for a single certificate associated only with your server
- Server Certificate Chain—for a CA's certificate to include in a certificate chain
- Trusted Certificate Authority (CA)—for a certificate of a CA that you want to accept as a trusted CA for client authentication

6. Select the cryptographic module from the drop-down list.
7. Enter the password for your key-pair file.
8. Leave the name for the certificate field blank if it will be the only one used for this server instance, unless:
 - Multiple certificates will be used for virtual servers. In this case, enter a certificate name unique within the server instance.
 - Cryptographic modules other than internal are used. In this case, enter a certificate name unique across all server instances within a single cryptographic module.

If a name is entered, it will be displayed in the Manage Certificates list, and should be descriptive. For example, “United States Postal Service CA” is the name of a CA, while “VeriSign Class 2 Primary CA” describes both a CA and the type of certificate.

NOTE When no certificate name is entered, the default value is applied.

9. Select one:
 - Message is in this file. In this case, enter the full pathname to the saved email
 - Message text (with headers). In this case, paste the email text.

If you copy and paste the text, be sure to include the headers “Begin Certificate” and “End Certificate,” including the beginning and ending hyphens.

10. Click OK.
11. Select one:
 - Add Certificate—to install a new certificate.
 - Replace Certificate—to install a certificate renewal.
12. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
13. Stop and restart the server for changes to take effect.

The certificate is stored in the server’s certificate database. The file name will be `cert7.db`.

Using the Built-in Root Certificate Module

The dynamically loadable root certificate module included with the Sun Java System Application Server contains the root certificates for many CAs. The root certificate module simplifies upgrading your root certificates. To install well-known CA certificates, you can update the root certificate module file to a newer version as it becomes available through future versions of the Sun Java System Application Server, or in service packs.

Because the root certificate is implemented as a PKCS11 cryptographic module, you can never delete the root certificates it contains; the option to delete will not be offered when managing these certificates. To remove the root certificates from your server instances, you can disable the root certificate module by deleting the following in the server's `alias` file:

- `libnssckbi.so` (on most UNIX platforms)
- `nssckbi.dll` (on Windows), located under `install_directory/bin/`

NOTE	You can modify the trust information of the root certificates. The trust information is written to the certificate database for the server instance being edited, not back to the root certificate module itself.
-------------	---

Managing Certificates

You can view or delete the trust settings of the various certificates installed on your server. This includes your own certificate and certificates from CAs. Certificate information includes the owner and who issued it.

Trust settings allow you to set client trust or unset server trust. For LDAP server certificates, the server must be trusted.

To manage certificates, perform the following steps in the Administration interface:

1. Access App Server Instances and select the server instance.
2. Access Security.
3. Select Certificate Management.
4. Click the Manage link.

- If you are managing a certificate for a default configuration using the internal cryptographic module, a list of all installed certificates with their type and expiration date is displayed. All certificates are stored in the *instance_dir/config* directory
 - If you are using an external cryptographic module, such as a hardware accelerator, you will first need to enter your password for each specific module and click OK. The certificate list will be updated to include certificates in the module.
5. Click the Certificate Name you want to manage.
An Edit Server Certificate page is displayed with management options for that type of certificate.
 6. In the Edit Server Certificate window you may select:
 - For certificates obtained internally—Delete Certificate or Quit
 - For CA certificates—Set client trust, Unset server trust, or Quit

NOTE Only CA certificates will allow you to set or unset client trust. Some external cryptographic modules will not allow certificates to be deleted.

You are prompted to confirm your edits.

7. Select OK or Cancel.
8. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
9. Stop and start the server for changes to take effect.

Managing CRLs and CKLs

A *certificate revocation list* (CRL) and *compromised key list* (CKL) publishes any certificates and keys that either client users or server users should no longer trust. Typical situations include:

- If data in a certificate changes, for example, a user changes offices or leaves the organization before the certificate expires, the certificate is revoked, and its data appears in a CRL.

- If a key is tampered with or otherwise compromised, the key and its data appear in a CKL.

Both CRLs and CKLs are produced and periodically updated by a CA. As the administrator, you can install new CRLs or CKLs that you obtain from the CA, or delete existing CRLs or CKLs from your system.

The following topics address managing CRLs and CKLs:

- [Installing a CRL or CKL](#)
- [Deleting a CRL or CKL](#)

Installing a CRL or CKL

To obtain a CRL or CKL from a CA, perform the following steps in the Administration interface:

1. Obtain the CA's URL for downloading CRLs or CKLs.
2. Enter the URL in your browser to access the site.
3. Follow the CA's instructions for downloading the CRL or CKL to a local directory.
4. In the Admin interface, access App Server Instances and select the server instance.
5. Access Security.
6. Select CRL/CKL.
7. Click the Install link.

The Install a Certificate Revocation List/Compromised Key List page is displayed.

8. Select one:
 - Certificate Revocation List
 - Compromised Key List
9. Enter the full pathname to the associated file.
10. Click OK.
 - If you selected Certificate Revocation List, the Add Certificate Revocation List page appears listing CRL information.

- If you selected Compromised Key List, the Add Compromised Key List page appears listing CKL information.

NOTE If a CRL or CKL list already exists in the database, a Replace Certificate Revocation List or Replace Compromised Key List page appears. In this case, click Replace.

11. Click Add.
12. Click OK.
13. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
14. Stop and start the server for changes to take effect.

Deleting a CRL or CKL

To delete a CRL or CKL, perform the following steps in the Administration interface:

1. Access App Server Instances and select the server instance.
2. Access Security.
3. Select CRL/CKL.
4. Click the Manage link.

The Manage a Certificate Revocation List/Compromised Key List page is displayed with all installed Server CRLs and CKLs listed along with their expiration dates.
5. Select a Certificate Name from either the Server CRLs or Server CKLs list.
6. Select one:
 - Delete CRL
 - Delete CKL
7. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
8. Stop and start the server for changes to take effect.

Administering SSL/TLS Encryption

Before beginning any of the tasks described in this section, you should already have a certificate and be familiar with the basic concepts of public-key cryptography, including encryption and decryption, public and private keys, digital certificates, and encryption protocols.

This section addresses the following topics:

- [About Encryption](#)
- [Enabling SSL Communication with LDAP](#)
- [Turning Security On](#)
- [Enabling SSL and TLS](#)
- [Configuring Security Globally](#)
- [Using External Encryption Modules](#)
- [Setting Strong Ciphers](#)
- [Preventing Clients from Caching SSL Files](#)

For more information on encryption and related topics, see *Introduction to SSL* in the Sun ONE Directory Server document collection.

About Encryption

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient; *decryption* is the process of transforming encrypted information so that it is intelligible again. The Sun Java System Application Server 7 supports the Secure Sockets Layer (SSL) and Transport Layer Security (TLS) encryption protocols.

A *cipher* is a cryptographic algorithm (a mathematical function), used for encryption or decryption. A *cipher suite* is a set of ciphers. The SSL and TLS protocols contain numerous cipher suites. Some ciphers are stronger and more secure than others. Generally speaking, the more bits a cipher uses, the harder it is to decrypt the data.

In any two-way encryption process, both parties must use the same ciphers. Because a number of ciphers are available, you need to enable your Sun Java System Application Server for those that are most commonly used in your environment.

SSL and TLS Protocols

The Sun Java System Application Server supports the Secure Sockets Layer (SSL) 3.0 and the Transport Layer Security (TLS) 1.0 protocols for encrypted communication. SSL and TLS are application independent, and higher-level protocols can be layered transparently on them.

SSL and TLS protocols support a variety of ciphers used to authenticate the server and client to each other, transmit certificates, and establish session keys. Clients and servers may support different cipher suites, depending on factors such as which protocol they support, company policies on encryption strength, and government restrictions on export of encrypted software. Among other functions, the SSL and TLS handshake protocols determine how the server and client negotiate which cipher suites they will use to communicate.

During a secure connection, the client and the server agree to use the strongest cipher they have both enabled for communication. You can choose ciphers from the SSL2, SSL3, and TLS protocols.

NOTE	Improvements to security and performance were made after SSL version 2.0; you should not use SSL 2.0 unless you have clients that are not capable of using SSL 3.0. Client certificates are not guaranteed to work with SSL 2.0 ciphers.
-------------	--

Public and Private Keys

The encryption cipher process alone isn't enough to secure your server's confidential information. A key must be used with the encrypting cipher to produce the actual encrypted result, or to decrypt previously encrypted information. The encryption process uses two keys to achieve this result: a public key and a private key. Information encrypted with a public key can be decrypted only with the associated private key. The public key is published as part of a certificate; only the associated private key is safeguarded.

For description of the various cipher suites, and more information about keys and certificates, see *Introduction to SSL* at the following location:

<http://docs.sun.com/db/prod/3802#hic>

Task Sequence

The task sequence for setting up basic security is:

1. Install a certificate.
Refer to “[Administering Certificates](#)” on page 43.
2. Enable encryption communication with LDAP.
Refer to “[Enabling SSL Communication with LDAP](#)” on page 62.
3. Turn security on.
Refer to “[Turning Security On](#)” on page 62.
4. Enable encryption protocols.
Refer to “[Enabling SSL and TLS](#)” on page 64.
5. Set global security using SSL directives.
“[Configuring Security Globally](#)” on page 66.

Other administrative tasks associated with encryption are discussed in “[Using External Encryption Modules](#)” on page 68, “[Setting Strong Ciphers](#)” on page 72, and “[Preventing Clients from Caching SSL Files](#)” on page 73.

Enabling SSL Communication with LDAP

After a server certificate has been installed, you are ready to activate encryption on the Sun Java System Application Server. It is important that you immediately configure your Admin Server to communicate with the LDAP database using SSL.

NOTE This section applies only to HTTP server functionality. Setting SSL communication to LDAP here has no relation to the LDAP communication done by J2EE applications (which use LDAP as described in the *Sun Java System Application Server Developer's Guide*).

To enable SSL, perform the following steps:

1. Access App Server Instances and select the server instance in the left pane.
2. Access Security in the left pane.
3. Select Configure Directory Service.
4. Click Yes next to Use Secure Sockets Layer (SSL) for connections?
5. Click Save Changes.

You are asked if you want to switch to the standard port.

6. Click OK to change your port to the standard port for LDAP over SSL.
7. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
8. Stop and start the server for changes to take effect.

Turning Security On

You must turn security on before you can configure any other security settings. The following sections describe the ways of turning security on:

- [Turning Security On When Creating as HTTP Listener](#)
- [Turning Security On When Editing an HTTP Listener](#)
- [Enabling SSL and TLS](#)

Turning Security On When Creating as HTTP Listener

To turn security on when creating a new HTTP listener, perform the following steps:

1. Access App Server Instances and select the server instance in the left pane.
2. Access HTTP Server and HTTP Server Listeners in the left pane.
The HTTP Listeners page is displayed.
3. Click New.
The listener settings page is displayed.
4. Check Security Enabled under the SSL/TLS Settings to turn security on.
5. Under Certificate Nickname, select the certificate. For example, `Server-Cert`.
6. Enter the rest of the information for the new HTTP listener. Additional information on the fields is contained in the online help.
7. Click OK.
8. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
9. Stop and start the server for changes to take effect.

Turning Security On When Editing an HTTP Listener

To turn security on when editing an existing HTTP listener, perform the following steps:

1. Access App Server Instances and select the server instance in the left pane.
2. Access HTTP Server and HTTP Listeners in the left pane.
3. Select a listener.
The HTTP Listeners settings page is displayed.

NOTE If you have an external module installed, the Manage Server Certificates page appears requiring the external module's password before you can continue.

4. Under the SSL/TLS Settings, check Security Enabled to turn security on.
5. Under Certificate Nickname, select the certificate. For example, `Server-Cert`.
6. Click Save.
7. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
8. Stop and start the server for changes to take effect.

Enabling SSL and TLS

To protect the security of your application server, you should enable SSL by enabling the SSL2, SSL3, and TLS encryption protocols and selecting the various cipher suites.

The default settings allow the most commonly-used ciphers. Unless you have a compelling reason for not using a specific cipher suite, you should allow them all. For more information regarding specific ciphers, see *Introduction to SSL* at the following location:

<http://docs.sun.com/db/prod/3802#hic>

CAUTION Do not select “No Encryption, only MD5 message authentication”. If no other ciphers are available on the client side, the server will default to this setting and no encryption will occur.

Before enabling SSL and TLS, security must be turned on, and you must have at least one certificate installed.

To enable SSL and TLS, perform the following steps:

1. Access App Server Instances and select the server instance in the left pane.
2. Access HTTP Servers and HTTP Server Listeners in the left pane.

3. Select the HTTP listener you want.

The HTTP Listeners setting page is displayed.

NOTE If you have an external module installed, the Manage Server Certificates page appears requiring the external module's password before you can continue.

4. Under the SSL/TLS Settings, check the appropriate boxes associated with SSL and TLS, including all the ciphers.

NOTE Unless you have a compelling reason for not using a specific cipher suite, you should allow them all.

5. For rollback:

- TLS must also be enabled on the browser seeking access to your server:
- For Netscape Navigator 6.0, check both TLS and SSL3.
- For Microsoft Internet Explorer 5.0 and 5.5, use the TLS Rollback option.
- For TLS Rollback, check TLS and make sure both SSL3 and SSL2 are disabled.

6. Click Save.

7. Access App Server Instances and your server instance in the left pane, then click Apply Changes.

8. Stop and start the server for changes to take effect.

The `init.conf` file is automatically modified to show security on, and all virtual servers are automatically assigned the default security parameters.

After you have enabled SSL on a server, its URLs use `https` instead of `http`. URLs that point to documents on an SSL-enabled server have this format:

```
https://servername.[domain.[dom]]:[port#]
```

For example:

```
https://admin.sun.com:443
```

NOTE If you use the default secure HTTP port number (443), you don't have to enter the port number in the URL.

Configuring Security Globally

Installing an SSL-enabled server creates directive entries in the `init.conf` file for global security parameters. Security must be enabled for virtual server security settings to work. SSL properties for virtual servers can be found on a per-server basis in the `ssl` element of the `server.xml` file.

The Security directive globally enables or disables SSL by making certificates available to the server instance. If enabled, the user is prompted for the administrator password to access certificates, and so on).

NOTE When you create a secure HTTP listener through the Administration interface, security is automatically turned on globally in the `init.conf` file. When you create a secure HTTP listener manually in the `server.xml` file, security must be turned on by editing the `init.conf` file.

The following topics address global security configuration:

- [SSL Configuration File Directives](#)
- [Setting Values for SSL Directives](#)

SSL Configuration File Directives

To configure security globally, you must set values for the following SSL configuration file directives in the `init.conf` file:

- [SSLCacheEntries](#)
- [SSLClientAuthDataLimit](#)
- [SSLClientAuthTimeout](#)
- [SSLSessionTimeout](#)
- [SSL3SessionTimeout](#)

SSLCacheEntries

Specifies the number of SSL sessions that can be cached. There is no upper limit.

Syntax

`SSLCacheEntries number`

If *number* is 0, the default value, which is 10000, is used.

SSLClientAuthDataLimit

Specifies the maximum amount of application data, in bytes, that is buffered during the client certificate handshake phase. The default value is 1048576 (1 MB).

SSLClientAuthTimeout

Specifies the number of seconds after which the client certificate handshake phase times out. The default value is 60.

SSLSessionTimeout

Controls SSL2 session caching.

Syntax

`SSLSessionTimeout seconds`

The *seconds* value is the number of seconds until a cached SSL2 session becomes invalid. If the `SSLSessionTimeout` directive is specified, the value of seconds is silently constrained to be between 5 and 100 seconds. The default value is 100.

SSL3SessionTimeout

Controls SSL3 session caching.

Syntax

`SSL3SessionTimeout seconds`

The *seconds* value is the number of seconds until a cached SSL3 session becomes invalid. The default value is 86400 (24 hours). If the `SSL3SessionTimeout` directive is specified, the value of seconds is silently constrained to be between 5 and 86400 seconds.

Setting Values for SSL Directives

To set values for your SSL configuration file directives, perform the following steps:

1. Access App Server Instances and select the server instance in the left pane.
2. Access HTTP Server and HTTP Listener in the left pane.
3. Click the HTTP listener you want.

The HTTP Listener settings page is displayed.

4. In the SSL/TLS Settings section, check the Security Enabled checkbox.
5. Under Certificate Nickname, select the certificate. For example, `Server-Cert`.
6. Click Save.
7. Select HTTP Server in the left pane.

The HTTP Server page is displayed.

8. Select the Advanced tab.

The Advanced settings page is displayed.

9. Choose the SSL link.

The SSL directives table is displayed.

10. Select the cipher suites and values.
11. Click OK.
12. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
13. Stop and start the server for changes to take effect.

Using External Encryption Modules

The Sun Java System Application Server supports the two methods for using external cryptographic modules (such as smart cards or token rings): Public Key Cryptography Standard (PKCS#11) and Federal Information Processing Standards (FIPS-140).

NOTE You will need to add the PKCS #11 module before enabling the FIPS-140 encryption standard.

This section addresses the following topics:

- [Installing the PKCS11Module](#)
- [Enabling FIPS-140 Standard](#)
- [Starting the Server with an External Certificate](#)

Installing the PKCS11Module

The Sun Java System Application Server supports Public Key Cryptography Standard (PKCS) #11, which defines the interface used for communication between SSL and PKCS11 modules. PKCS11 modules are used for standards-based connectivity to SSL hardware accelerators. Imported certificates and keys for external hardware accelerators are stored in the `secmod.db` file, which is generated when the PKCS11 module is installed.

Starting the Server with an External Certificate

If you install a certificate for your server into an external PKCS11 module (for example, a hardware accelerator), the server will not be able to start using that certificate until you enable the HTTP listener to use that certificate.

The server always tries to start with the certificate named `Server-Cert`. However, certificates in external PKCS11 modules include one of the module's token names in their identifier. For example, a server certificate installed on an external smartcard reader called `smartcard0` would be named `smartcard0:Server-Cert`.

To start a server with a certificate installed in an external module, you'll need to specify the certificate name. Using the Administration interface, request and install the certificate specifying the hardware cryptographic module to use.

After you install the certificate in the external hardware token, you can see it in the Manage Certificate page, but not in the HTTP Listener page due to the inherent limitation of the Administration interface.

At this point, you must use the command-line interface to edit the HTTP listener to select the certificate for SSL, change the port number, and so on.

1. Edit the HTTP listener to select the certificate:

```
/sun/appserver7/bin/asadmin create-ssl
    -user admin
    -password netscape
    -host qa280r-1.red.ipplanet.com
    -port 8888
    -type http-listener
    -certname nobody@apprealm:Server-Cert
    -instance server1
    -ssl3enabled=true
    -ssl3tlsciphers +rsa_rc4_128_md5
    http-listener-1
```

To find out the external certificate nickname, go to Manage Certificates page, and enter the key password for external tokens. The certificate name will be displayed, such as `nobody@apprealm:Server-Cert`.

2. Enable security on the HTTP listener:

```
/sun/appserver7/bin/asadmin set
    -user admin
    -password netscape
    -host qa280r-1.red.ipplanet.com
    -port 8888
    server1.http-listener.http-listener-1.securityEnabled=true
```

3. Change the port number of the HTTP listener:

```
/sun/appserver7/bin/asadmin set
    -user admin
    -password netscape
    -host qa280r-1.red.ipplanet.com
    -port 8888
    server1.http-listener.http-listener-1.port=443
```

4. Apply the changes thus far:

```
/sun/appserver7/bin/asadmin reconfig
    -u admin
    -w netscape
    -H qa280r-1.red.ipplanet.com
    -p 8888
    server1
```

5. Stop and start the server to enable the HTTP listener to listen on SSL.

Enabling FIPS-140 Standard

PKCS11 APIs enable communication with software or hardware modules that perform cryptographic operations. Once PKCS11 is installed on your Sun Java System Application Server, you can configure the server to be Federal Information Processing Standards (FIPS)-140 compliant.

NOTE These libraries are included only in SSL version 3.0.

To enable FIPS-140, perform the following steps:

1. Install the plug-in following the FIPS-140 instructions.
2. In the Administration interface, access App Server Instances and select the server instance in the left pane.
3. Access HTTP Server and HTTP Listeners in the left pane.
4. Click the link for the HTTP listener you want.
The HTTP Listeners page is displayed.
5. In the SSL/TLS Settings section of the page, check Security Enabled if it is not already checked.
6. Check SSL3 Enabled, if it is not already checked.
7. Check all the SSL/TSL Ciphers, if they are not already checked.
8. Click Save.
9. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
10. Stop and start the server for changes to take effect.

Setting Strong Ciphers

The Strong Ciphers option presents a choice of 168, 128, or 56-bit secret key size for access or no restriction. You can specify a file to be served when the restriction is not met. If no file is specified, the Sun Java System Application Server returns a Forbidden status.

If you select a key size for access that is not consistent with the current cipher settings, the Sun Java System Application Server warns that you need to enable ciphers with larger secret key sizes.

The implementation of the key size restriction is based on an NSAPI `PathCheck` directive in the `obj.conf` file, rather than Service `fn=key-toosmall`. This directive is:

```
PathCheck fn="ssl-check" [secret-keysize=nbits] [bong-file=filename]
```

where:

nbits is the minimum number of bits required in the secret key.

filename is the name of a file (not a URI) to be served if the restriction is not met.

`PathCheck` returns `REQ_NOACTION` if SSL is not enabled, or if the `secret-keysize` parameter is not specified. If the secret key size for the current session is less than the specified `secret-keysize`, the function returns `REQ_ABORTED` with a status of `PROTOCOL_FORBIDDEN` if `bong-file` is not specified, or else `REQ_PROCEED`, and the path variable is set to the `bong-file` *filename*. Also, when a key size restriction is not met, the SSL session cache entry for the current session is invalidated, so that a full SSL handshake will occur the next time the same client connects to the server.

NOTE	The Strong Ciphers option removes any Service <code>fn=key-toosmall</code> directives that it finds in an object when it adds a <code>PathCheck</code> <code>fn=ssl-check</code> .
-------------	--

To use the Strong Ciphers option, perform the following steps:

1. Access App Server Instances and select the server instance in the left pane.
2. Access HTTP Server in the left pane.
3. Select Virtual Servers and click the virtual server you want.
4. Select the HTTP/HTML tab, then click the Strong Ciphers link.

The Enforce Strong Security Requirements page is displayed.

5. Choose to edit:
 - From the drop down list
 - By clicking Browse
 - By clicking Wildcard
6. Select the secret key size restriction:
 - 168 bit or larger
 - 128 bit or larger
 - 56 bit or larger
 - No restrictions
7. Enter the file location of the message that will be used to reject access.
8. Click OK.
9. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
10. Stop and start the server for changes to take effect.

Preventing Clients from Caching SSL Files

You can prevent pre-encrypted files from being cached by a client by adding the following line inside the `<HEAD>` section of a file in HTML:

```
<meta http-equiv="pragma" content="no-cache">
```


Administering HTTP Server Access Control

This section provides information on the mechanisms and procedures used for controlling access to HTTP server resources.

NOTE The content in this chapter cannot be applied to J2EE applications. For J2EE application development, refer to the J2EE security mechanisms as described in the J2EE specifications and the *Sun Java System Application Server Developer's Guide*.

This section addresses the following topics:

- [About HTTP Server Access Control](#)
- [Implementing Digest Authentication](#)
- [Implementing Host-IP Authentication](#)
- [Working With ACL Files](#)
- [Setting Up Client Authentication](#)
- [ACL/ACE Settings](#)
- [Referencing ACL Files in the obj.conf File](#)
- [Configuring the ACL User Cache](#)
- [Setting Access Control for a Server Instance](#)
- [Restricting Access to Areas of Your Server](#)
- [Turning Off Access Control](#)
- [Responding When Access is Denied](#)

- [Controlling Access for Virtual Servers](#)
- [Using htaccess Files](#)

About HTTP Server Access Control

Access control is the means of securing your Sun Java System Application Server by controlling who and what has access to it. For example, you can specify who has full control of all the servers installed on a machine and who has partial control of one or more servers.

NOTE	Before you can apply access control on the Admin Server set up an administration group. The instructions in this section assume you have already defined users and groups in your directory database.
-------------	---

Access is allowed or denied based on:

- Who is making the request
- Where the request is coming from
- When the request is happening (for example, time of day)
- What type of connection is being used (such as SSL)

The security mechanisms that control access to your HTTP server include various authentication restrictions and ACL files.

This section addresses the following topics:

- [HTTP Server User-Group Authentication](#)
- [Host-IP Authentication](#)
- [Access Control List \(ACL\) Files](#)
- [Client Authentication](#)

HTTP Server User-Group Authentication

User-Group authentication requires that users authenticate themselves before access can be granted. This is done by entering a user name and password, using a client certificate, or a using the digest authentication plug-in. The Sun Java System Application Server receives this information encrypted or unencrypted, depending on whether encryption is turned on for your server.

NOTE In J2EE applications, user-group authentication is provided through realms (security domains). For information on developing realms for J2EE application security, refer to the *Sun Java System Application Server Developer's Guide*.

The default authentication used is the default method you specify in the `obj.conf` file, or Basic authentication if there is no setting in the `obj.conf` file.

If you select Default as your authentication method, the ACL rule doesn't specify a method in the ACL file. Choosing Default allows you to easily change the methods for all ACLs by editing one line in the `obj.conf` file.

Using the default is the preferred method.

There are three User-Group authentication methods, all of which require a directory server:

- [Basic Authentication](#)
- [SSL Authentication](#)
- [Digest Authentication](#)

NOTE If you want to require client authentication on your Admin Server, you must manually edit the ACL files in the `obj.conf` file, changing the method to SSL. For further information on client authentication, refer to [“Setting Up Client Authentication” on page 96](#).

Basic Authentication

Basic authentication requires a user to enter a user name and password to access your Sun Java System Application Server or web site. You must first create and store a list of users and groups in an LDAP database, such as the Sun ONE Directory Server. The directory server must be installed on a different server root than your Sun Java System Application Server, or on a directory server installed on a remote machine.

Basic authentication is the default setting.

NOTE	Using Basic authentication without SSL encryption sends the user name and password in unencrypted text across the network. There is a risk that network packets could be intercepted, and the user name and password could be stolen. Basic authentication is most effective when combined with SSL encryption, Host-IP authentication, or both. Using Digest authentication avoids this problem.
-------------	---

SSL Authentication

SSL authentication requires that the Sun Java System Application Server confirm user identities using the users' security certificates. This can happen in two ways:

- Using the information in the client certificate as proof of identity
- Verifying a client certificate published in an directory (additional)

When you set the server to use certificate information for authenticating the client, the Sun Java System Application Server does the following:

- Verifies that the certificate is from a trusted authority (CA). If not, the authentication fails and the transaction ends.
- If the certificate is from a trusted certificate CA, maps the certificate to a user's entry using the `certmap.conf` file. To learn how to set up the certificate mapping file see [“Working with the certmap.conf File” on page 98](#).
- If the certificate maps correctly, checks the ACL rules specified for that user.

NOTE	Even if the certificate maps correctly, ACL rules may deny access to the user.
-------------	--

User-Group SSL Authentication

If you set the server's access control to use the User-Group SSL authentication method, the following must be true:

- A valid certificate issued by a trusted CA is presented.
- The certificate is mapped to a valid user in the directory database.
- The access control list (ACL) evaluates properly

Client SSL Authentication to Specific Resources

Requiring client authentication for controlling access to specific resources differs from requiring client authentication for all connections to the server. If you set the server to require client authentication for all connections, the client only needs to present a valid certificate issued by a trusted CA. To learn how to turn on client authentication, see [“Setting Up Client Authentication” on page 96](#).

When you require client SSL authentication, you need to have SSL ciphers enabled for your server. Refer to [“Enabling SSL and TLS” on page 64](#) for further information.

To successfully gain access to an SSL-authenticated resource, the client certificate must be from a CA trusted by the server. The client certificate needs to be published in a directory server if the server's `certmap.conf` file is configured to compare the client's certificate in the browser with the client certificate in the directory server. However, the `certmap.conf` file can be configured to compare only selected information from the certificate to the directory server entry. For example, you could configure the `certmap.conf` file to compare only the user ID and email address in the browser certificate with the directory server entry. To learn more about the `certmap.conf` file and certificate mapping, see [“Working with the certmap.conf File” on page 98](#).

NOTE

Only the User-Group SSL authentication method requires modifying the `certmap.conf` file (because the certificate is checked against the directory database); requiring client authentication for all connections to the server does not. If you choose to use client certificates, you should increase the value of the `AcceptTimeout` directive in the `init.conf` file.

Digest Authentication

Digest authentication allows the user to authenticate based on user name and password without sending the user name and password as cleartext. The browser uses the MD5 algorithm to create a digest value using the user's password and some information provided. This digest value is also computed on the server side using the Digest authentication plug-in, and compared against the digest value provided by the client. If the digest values match, the user is authenticated.

For this to work, your directory server needs access to the user's password in cleartext. The Sun ONE Directory Server includes a reversible password plug-in that uses a symmetric encryption algorithm to store data in an encrypted form. Later the data can be decrypted to its original form. Only the Sun ONE Directory Server holds the key to the data.

When processing an ACL with `method=digest`, the server attempts to authenticate using the following process:

- Checking for authorization request header—If not found, an error is generated with a Digest challenge, and the process stops.
- Checking for authorization type—If Authentication type is Digest, the server does the following:
 - Checks nonce—If this is not a valid, fresh nonce is generated by this server, an error is generated, and the process stops. If stale, an error is generated with `stale=true`, and the process stops.
 - Checks realm—If it does not match, an error is generated, and the process stops.
 - Checks existence of user in LDAP directory—If not found, an error is generated, and the process stops.
 - Gets request-digest value from directory server and checks for match to client's request-digest—If it does not match, an error is generated, and the process stops.
 - Constructs Authorization-Info header and inserts into server headers.

The server tries to authenticate against the directory database based on the ACL method as specified in the following table.

Table 5-1 Digest Authentication Challenges

ACL Method Configured	Authentication DB Supports Digest Authentication	Authentication DB Does Not Support Digest Authentication
Default None specified	Digest and Basic	Basic
Basic	Basic	Basic
Digest	Digest	ERROR

For more information, refer to [“Implementing Host-IP Authentication” on page 85](#).

Host-IP Authentication

Host-IP access control is a method of limiting access to the Admin Server, or the files and directories on your web site by making them available only to clients using specific computers. You do this by specifying the hostnames or IP addresses for the computers that you want to allow or deny access to. You can use wildcard patterns to specify multiple computers or entire networks.

Host-IP authentication appears seamless to users, who can access the files and directories immediately without entering a user name or password. Further information can be found in [“Implementing Host-IP Authentication” on page 85](#).

Access Control List (ACL) Files

ACL files are text files that contain lists identifying who can access the resources stored on your Sun Java System Application Server. By creating a hierarchy of rules called *access control entries* (ACEs) you can allow or deny access to individuals, groups, or other entities such as particular servers or applications. Each ACE specifies whether or not the server should check the next ACE in the hierarchy. The collection of ACEs you create is called an *access control list* (ACL).

By default, the Sun Java System Application Server uses a single ACL file that contains all the lists for accessing your server. As an alternative to this default, you can create multiple ACL files and reference them in the `obj.conf` file.

When the Sun Java System Application Server receives a request for a page, the server uses the rules in the ACL file to determine if it should grant access or not. The rules can reference the hostname or IP address of the computer sending the request. The rules can also reference users and groups stored in an LDAP directory such as the Sun ONE Directory Server.

After determining which virtual server to use for an incoming request, the Sun Java System Application Server determines if any ACLs are configured for that virtual server. If ACLs are found that apply to the current request, the Sun Java System Application Server evaluates their ACEs to determine whether access should be allowed (granted) or denied (refused).

Information on using ACLs is contained in [“Working With ACL Files” on page 86](#).

Client Authentication

When client authentication is enabled, the client’s certificate is required before the server will send a response to a query. The Sun Java System Application Server supports authenticating client certificates by matching the CA in the client certificate with a CA trusted for signing client certificates.

When the Sun Java System Application Server receives a request from a client, it asks for the client’s certificate before proceeding. Some clients send the client certificate to the server along with the request.

NOTE	Before mapping client certificates to LDAP, you need to set up the required ACLs. More information on access control can be found in “Working With ACL Files” on page 86
-------------	--

1. The Sun Java System Application Server tries to match the CA to the list of trusted CAs in the Admin Server.

If there isn’t a match, the Sun Java System Application Server ends the connection.

2. If there is a match, the server continues processing the request.
3. After verifying the certificate is from a trusted CA, the server maps the certificate to an LDAP entry by:
 - Mapping the issuer and subject DN from the client certificate to a branch point in the LDAP directory.

- Searching the LDAP directory for an entry that matches the information about the subject (end-user) of the client certificate.

The server uses a certificate mapping file called `certmap.conf` to determine how to do the LDAP search. The mapping file tells the server what values to take from the client certificate (such as the user's name, email address, and so on). The server uses these values to search for a user entry in the LDAP directory, but first the server needs to determine where in the LDAP directory it needs to start its search. The certificate mapping file tells the server where to start.

- (Optional) Verifying the client certificate with one in the LDAP entry that corresponds to the DN.
4. Once the server knows where to start its search and what it needs to search for, it performs the search in the LDAP directory. If it finds no matching entry or more than one matching entry, and the mapping is *not* set to verify the certificate, the search fails.

You can specify the expected behavior in the ACL. For example, you can specify that Sun Java System Application Server accepts only you if the certificate match fails. For more information regarding how to set the ACL preferences, see [“ACL File Syntax” on page 87](#).

5. After the server finds a matching entry and certificate in the LDAP directory, it can use that information to process the transaction. For example, some servers use certificate-to-LDAP mapping to determine access to a server.

Information on implementation is contained in [“Setting Up Client Authentication” on page 96](#).

Implementing Digest Authentication

If you are unfamiliar with how Digest authentication works, refer to [“Digest Authentication” on page 80](#).

For Digest authentication, you need to enable the reversible password plug-in and the `digestauth`-specific plug-in included with the Sun Java System Application Server. To configure your server to process Digest authentication, you will need to set the `digestauth` property of the database definition in the `dbswitch.conf` file.

The tasks required for implementing User-Group Digest authentication are addressed in the following sections:

- [Installing the Digest Authentication Plug-in](#)

- [Setting the Sun ONE Directory Server to Use the DES Algorithm](#)

Installing the Digest Authentication Plug-in

The Digest authentication plug-in consists of a shared library found in both `libdigest-plugin.lib` and `libdigest-plugin.ldif`.

Digest Authentication on UNIX

To install the Digest authentication plug-in on UNIX, perform the following steps:

1. Make sure this shared library resides on the same server machine that the Sun ONE Directory Server is installed on.
2. Make sure you know the Directory Manager password.
3. Modify the `libdigest-plugin.ldif` file changing all references to `/path/to` to the location where you installed the Digest plug-in shared library.
4. To install the plug-in, enter the following command:

```
% ldapmodify -D "cn=Directory Manager" -w password -a <
libdigest-plugin.ldif
```

Digest Authentication on Windows

To install the Digest authentication plug-in on Windows, perform the following steps:

NOTE	You will need to copy several <code>.dll</code> files from the Sun Java System Application Server installation to your Sun ONE Directory Server machine in order for the Sun ONE Directory Server to start properly with the Digest plug-in.
-------------	--

1. Access the shared libraries in the Sun Java System Application Server installation in:

install_dir\bin

2. Copy the files:

- o `nsldap32v50.dll`
- o `libnspr4.dll`

- libplds4.dll

3. Paste them into either:

- \Winnt\system32
- Sun ONE Directory Server installation directory:
[*server_root*]\bin\sldap\server

Setting the Sun ONE Directory Server to Use the DES Algorithm

The DES algorithm is needed to encrypt the attribute where the Digest password is stored. To set the Sun ONE Directory Server to use the DES algorithm, perform the following steps:

1. Launch the Sun ONE Directory Server Console.
2. Open your Sun ONE Directory Server instance.
3. Select the Configuration tab.
4. Click the + sign next to plug-ins.
5. Select the DES plug-in.
6. To add a new attribute, choose Add.
7. Enter `iplanetReversiblePassword`.
8. Click Save.
9. Stop and start the server for changes to take effect.

NOTE In order to set a digest authentication password in the `iplanetReversiblePassword` attribute for a user, your entry must include the `iplanetReversiblePasswordobject` object.

Implementing Host-IP Authentication

Since more than one person may use a particular computer, Host-IP authentication is more effective when combined with User-Group authentication. If both User-Group and Host-IP authentication are used, a user name and password *is* required for access.

Host-IP authentication does not require DNS to be configured on your Sun Java System Application Server, but you must have DNS running in your network and your Sun Java System Application Server must be configured to use it.

NOTE You can enable DNS on the HTTP Server->Tuning page of the Administration interface.

Enabling DNS degrades the performance of the Sun Java System Application Server because the server is required to do DNS look-ups. To reduce the impact of DNS look-ups on your Sun Java System Application Server performance, resolve IP addresses only for access control and CGI instead of resolving IP addresses for every request. To minimize DNS impact, append `iponly=1` to `AddLog fn="flex-log" name="access"` in your `obj.conf` file:

```
AddLog fn="flex-log" name="access" iponly=1
```

Working With ACL Files

The main ACL file name is `generated.server-id.acl`; the temporary working file is called `genwork.server-id.acl`. If you use the Admin Server to configure access, these two files will exist. However, if you want more complex restrictions, you can create multiple files, and reference them from the `server.xml` file. There are also a few features available only by editing the files, such as restricting access to the server based on the time of day or day of the week.

Access control files are stored in the directory *instance_dir*/config where *instance_dir* is the name of the instance. For example, the access control files for the initial server instance are stored in the *install_dir*/domains/domain1/server1/config directory.

The following topics are addressed in this section:

- [ACL File Syntax](#)
- [Type Statement](#)
- [Authentication Statement](#)
- [Authorization Statement](#)
- [Sample ACL File](#)
- [Writing Customized ACL Expressions](#)

ACL File Syntax

ACL files must follow a specific format and syntax. A typical ACL definition in the ACL list is made up of a number of statements about type, the authentication method, and the authorization method.

A snippet from an ACL file might look like this:

```
version 3.0;
# The "default" rules apply to the entire document
acl "default";
authenticate (user,group) {
  database = "default";
  method = "basic";
};
deny (all)
user = "anyone";
allow (read,execute,list,info)
(user = "all");
};
```

The components of this snippet include:

- **Version Line**—All ACL files begin with a line specifying the version number. There can be only one version line in an ACL file. For example:

```
version 3.0;
```

- **Type statement**—Identifies the type of ACL the definition is for. For example:

```
acl "default";
```

- **Authentication statement**—Optionally specifies the authentication method. For example:

```
authenticate (user,group) {
  database = "default";
  method = "basic";
};
```

- **Authorization statement**—Specifies who is allowed or denied access to a server resource. For example:

```
deny (all)
(user = "anyone");
allow (read,execute,list,info)
(user = "all");
```

Input strings can contain the following characters:

- Letters a through z
- Numbers 0 through 9
- Period and underscore

If you use any other characters, you must use double quotation marks (") around the characters.

- A single statement is placed on its own line and is terminated with a semicolon.
- Multiple statements are placed within braces.
- In a list of items, the items must be separated from each other with commas, and the list must be enclosed in double quotation (") marks.

Type Statement

Each ACL definition in the file contains a statement that identifies what type of ACL it is. This type can be one of the following:

- Path ACL—Specifies an absolute path to the resource it affects.
- URI (Uniform Resource Indicator) ACL—Specifies a directory or file relative to the server's document root.
- Named ACL—Specifies a name that is referenced in resources in the `obj.conf` file. The server comes with a default named resource that allows read access to anyone and write access to users in the LDAP directory. Even though you can create a named ACL from the Sun Java System Application Server, you must manually reference the named ACLs with resources in the `obj.conf` file.

A type line begins with the letters `acl`, followed by the type information in double quotation marks, ended by a semicolon. The type information for each ACL must be a unique name—even among different ACL files. The following lines are examples of several different types of ACLs:

```
acl "path=C:/Sun/Servers/docs/mydocs/";
acl "default";
acl "uri=/mydocs/";
```

Path and URI ACLs can include wildcards at the end of the entry. For example: `/a/b/*`. Wildcards placed anywhere except at the end of the entry will not work.

Authentication Statement

Optionally, ACLs can specify the authentication method the Sun Java System Application Server must use when processing the ACL. There are three general methods:

- **Basic authentication—(Default)** Requires the user to enter a user name and password before accessing a resource.

By default, the Sun Java System Application Server uses the Basic method for any ACL that doesn't specify a method.
- **SSL authentication**—Requires the user to have a client certificate. The Sun Java System Application Server must have encryption turned on, and the user's certificate issuer must be in the list of trusted CAs.
- **Digest authentication**—Requires the user to enter a user name and password before accessing a resource.

NOTE	In this case, your Sun Java System Application Server authentication database must be able to handle Digest authentication sent by a user.
-------------	--

Each authentication statement specifies what attributes (users, groups, or both users and groups) the Sun Java System Application Server must authenticate.

Examples

Specifies Basic authentication with users matched to individual users in the database or directory:

```
authenticate (user) {
    method = "basic";
};
```

Specifies the User-Group SSL authentication method:

```
authenticate (user, group) {
    method = "ssl";
};
```

Allows access to any user whose user name begins with the letters `sales`:

```
authenticate (user) {
    allow (all)
        user = sales*
};
```

If the last line were changed to `group=sales`, the ACL would fail because the authenticate line specifies `user`, not `group`.

Authorization Statement

An *authorization statement* specifies who is allowed or denied access to a server resource. Each ACL entry can include one or more authorization statements. Use the following syntax when writing authorization statements:

```
allow|deny [absolute] (right[,right...]) attribute expression;
```

Each line starts with the keyword `allow` or `deny`.

Because of the hierarchy of rules, it is usually a good idea to deny access to everyone in the top-level rule, then specifically allow access for users, groups, or computers in subsequent rules.

Scenario

If you allow full access to a directory called `/my_stuff`, then you want a subdirectory called `/my_stuff/personal` that will allow access to only a few users, the access control on the subdirectory won't work—everyone allowed access to the `/my_stuff` directory will also be allowed access to the `/my_stuff/personal` directory. To prevent this, first create a rule for `/my_stuff/personal` that denies access to everyone, then allow access for the few users you specify.

NOTE In some cases, if you set the default ACL to deny access to everyone, then your other ACL rules won't need a deny all rule.

The following authorization statement denies access to everyone:

```
deny (all)
  user = "anyone";
```

The following sections are association with authorization statements:

- [Hierarchy of Authorization Statements](#)
- [Attribute Expressions](#)
- [Operators](#)

Hierarchy of Authorization Statements

ACLs have a hierarchy based on the resource. For example, if the Sun Java System Application Server receives a request for the document (URI) `/my_stuff/web/presentation.html`, the server builds a list of ACLs that apply for this URI.

- First the Sun Java System Application Server adds ACLs listed in the `check-acl` statements of the server's `obj.conf` file.
- Then the server appends matching URI and PATH ACLs.

All statements are evaluated in order, unless absolute ACL statements are present. If an absolute `allow` or absolute `deny` statement evaluates to true, the server stops processing and accepts this result.

If there are multiple ACLs that match, the Sun Java System Application Server uses the last statement that matches. However, if you use an absolute statement, the server stops looking for other matches and uses the ACL containing the absolute statement. If you have two absolute statements for the same resource, the Sun Java System Application Server uses the first one in the file and stops looking for other resources that match.

The following example stops searching when it reaches the absolute `"joe"`, even if there might be additional users named `joe`:

```
version 3.0;
acl "default";
authenticate (user,group) {
    prompt="Sun Java System Application Server";
};
allow (read,execute,list,info)
    user = "anyone";
allow (write,delete)
    user = "all";
acl "uri=/my_stuff/web/presentation.html";
deny (all)
    user = "anyone";
allow (all)
    user = "joe";
```

Attribute Expressions

An *attribute expression* defines who is allowed or denied access based on their user name, group name, host name, or IP address. The following are examples of how access is given to different people or computers:

```
user = "anyone"
```

```

user = "smith*"
group = "sales"
dns = "*.sun.com"
dns = "*.sun.com,*.mozilla.com"
ip = "198.*"
ciphers = "rc4"
ssl = "on"
timeofday = <0800 or timeofday=1700
dayofweek = "Sat,Sun"

```

You can also restrict access to your server by time of day (based on the local time on the server) using the `timeofday` attribute. Refer to [“Restricting Access Based on Time of Day” on page 116](#) for additional information.

For example, the `timeofday` attribute can restrict access to certain users during specific hours.

NOTE Use 24-hour time to specify times. For example, use 0400 to specify 4:00 a.m. or 2230 for 10:30 p.m.

Example

Restricts access to a group of users called guests between 8:00 a.m. and 4:59 p.m.:

```

allow (read
    (group="guests") and
    (timeofday<0800 or timeofday=1700));

```

You can also restrict access by day of the week using the `dayofweek` attribute with the following three-letter abbreviations to specify days of the week: Sun, Mon, Tue, Wed, Thu, Fri, Sat.

Example

Allows access for users in the premium group any day and any time. Users in the discount group get access all day on weekends and on weekdays anytime except 8am-4:59pm.

```
allow (read)
(group="discount" and dayofweek="Sat,Sun") or
(group="discount" and (dayofweek="mon,tue,wed,thu,fri" and
(timeofday<0800 or timeofday=1700)))
or
(group="premium");
```

Operators

You can use various operators in attribute expressions. Parentheses delineate the operator order of precedence. With user, group, DNS, and IP, you can use the following operators:

- and
- or
- not
- = (equals)
- != (not equal to)

With the `timeofday` and `dayofweek` attributes, you can use:

- > greater than
- < less than
- >= greater than or equal to
- <= less than or equal to

Sample ACL File

The following sample ACL file contains the two default entries for the Admin Server (`admin-server`), plus an additional entry that allows users in the `admin-reduced` group to access the Admin Server.

```
version 3.0;
# The following "es-internal" rules protect files such
# as icons and images related to Sun Java System Application Server.
# These "es-internal" rules should not be modified.
acl "es-internal";
allow (read, list, execute,info) user = "anyone";
deny (write, delete) user = "anyone";
```

```

# The following "default" rules apply to the entire document
# directory of Sun Java System Application Server. In this example,
the rules
# are set up so that "all" users in the directory server are
# allowed to read, execute, list, and get information.
# The "all" users are not allowed to write to or delete any files.
# All clients that accesses the document directory of the web
# server will be required to submit a username and password
# since this example is using the "basic" method of
# authentication. A client must be in the directory server
# to gain access to this default directory since "anyone"
# not in the directory server is denied, and "all" in the
# directory server are allowed.
acl "default";
authenticate (user,group) {
    database = "default";
    method = "basic";
};
deny (all)
(user = "anyone");
allow (read,execute,list,info)
(user = "all");
# The following rules deny access to the directory "web"
# to everyone not in the directory server and deny everyone
# in the directory server who is not in GroupB.
# Only the users in GroupB are allowed read, execute, list,
# and info permissions. GroupA can not gain access to the
# directory "web" even though (in the ACL rule below) they
# can access the directory "my_stuff". Furthermore, members
# of GroupB can not write or delete files.
acl "path=/export/user/990628.1/docs/my_stuff/web/";
authenticate (user,group) {
    database = "default";
    method = "basic";
};
deny (all)
(user = "anyone");

allow (read,execute,list,info)
(group = "GroupB");

# The following rule denies everyone not in the directory
# server and denies everyone in the directory server except
# user with the ID of "SpecificMemberOfGroupB". The ACL rule
# in this setting also has a requirement that the user
# connect from a specific IP address. The IP address setting
# in the rule is optional; it has been added to for extra
# security. Also, this ACL rule has a Customized prompt
# of "Presentation Owner". This Customized prompt appears
# in the username and password dialog box in the client's

```

```
# browser.

acl
"path=/export/user/990628.1/docs/my_stuff/web/presentation.html";
authenticate (user,group) {
    database = "default";
    method = "basic";
    prompt = "Presentation Owner";
};
deny (all)
(user = "anyone" or group = "my_group");
allow (all)
(user = "SpecificMemberOfGroupB") and
(ip = "208.12.54.76");

# The following ACL rule denies everyone not in the directory
# server and everyone in the directory server except for
# GroupA and GroupB access to the directory "my_stuff"
acl "path=/export/user/990628.1/docs/my_stuff/";
authenticate (user,group) {
    database = "default";
    method = "basic";
};
deny (all)
(user = "anyone");
allow (read,execute,list,info)
(group = "GroupA,GroupB");
```

Example

If a user requests: `http://server_name/my_stuff/web/presentation.html`, the Sun Java System Application Server first checks access control for the entire server. If the ACL for the entire server was set to continue, the server would check for an ACL for the directory `my_stuff`. If an ACL exists, the server checks the ACEs within the ACL, then moves on to the next directory. This process continues until an ACL is found that denies access, or until the final ACL for the requested URL (in this case, the file `presentation.html`) is reached.

Writing Customized ACL Expressions

You can enter custom expressions for an ACL. There are a few features that are available only by editing the ACL file or by creating custom expressions. For example, you can restrict access to your server depending on the time of day, day of the week, or both.

NOTE Only select this option if you are familiar with the syntax and structure of ACL files.

The following customized expression shows how you could restrict access by time of day and day of the week. This example assumes you have two groups in your LDAP directory: the regular group is allowed access Monday through Friday, 8:00am to 5:00pm. The critical group is allowed access all the time.

```
allow (read)
{
    (group=regular and dayofweek="mon,tue,wed,thu,fri");
    (group=regular and (timeofday>=0800 and timeofday<=1700));
    (group=critical)
}
```

Setting Up Client Authentication

NOTE Certificate expired—If the certificate has expired, the Sun Java System Application Server logs an error, rejects the certificate, and returns a message to the client. You can view which certificates have expired in the Admin Server Manage Certificates page.

You can set up client authentication for the Admin Server or for a server instance. Instructions are contained in the following sections:

- [Setting Client Authentication for the Admin Server](#)
- [Setting Client Authentication for a Server Instance](#)
- [Working with the certmap.conf File](#)

Setting Client Authentication for the Admin Server

To set up client authentication at the Admin Server level:

1. Access the Admin Server and select the HTTP Listener tab in the right pane.
The Admin Server HTTP Listener page is displayed.
2. Click SSL/TLS Enabled to turn security on (default is off).
3. Click Client Authentication Enabled to turn client authentication on (default is off).
4. Click Save.
5. Select the Control tab on the right pane and click Apply Changes.
6. Stop and start the server for changes to take effect.

To start the Admin Server, go to

install_dir/domains/domain1/admin-server/bin/startserv

Setting Client Authentication for a Server Instance

To set up client authentication at the Server Instance level:

1. Access the App Server Instance and select the server instance in the left pane.
2. Expand HTTP Server in the left pane and click HTTP Listeners.
The HTTP Listener page is displayed, listing the listener instances.
3. Select the instance you want.
The HTTP Listener page for that instance is displayed.
4. Click the SSL/TLS Enabled checkbox to turn security on (default is off).
5. Click the Client Authentication Enabled checkbox to turn it on (default is off).
6. Click Save.
7. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
8. Stop and start the server for changes to take effect.

NOTE There is a single certificate trust database per Sun Java System Application Server instance. All the secure virtual servers running under that server instance share the same list of trusted client CAs. If two virtual servers require different trusted CAs, these virtual servers should be run in different server instances with separate trust databases.

Working with the certmap.conf File

Certificate mapping is the mechanism by which a server looks up a user entry in the LDAP directory. You can use the `certmap.conf` file to configure how a certificate, designated by name, is mapped to an LDAP entry. You edit this file and add entries to match the organization of your LDAP directory and to list the certificates you want your users to have. Users can be authenticated based on user ID, email, or any other value used in the `subjectDN`. Specifically, the mapping file provides the following information:

- Where in the LDAP tree the server should begin its search
- What certificate attributes the server should use as search criteria when searching for the entry in the LDAP directory
- Whether or not the server goes through an additional verification process

The certificate mapping file is located here:

`/instance_dir/config/certmap.conf`

The file contains one or more named mappings, each applying to a different CA. A mapping has the following syntax:

```
certmap <name> <issuerDN>
<name>:<property> [<value>]
```

The first line specifies a name for the entry and the attributes that form the DN found in the CA certificate. The name is arbitrary; you can define it to be whatever you want. However, `issuerDN` must *exactly* match the issuer DN of the CA who issued the client certificate. For example, the following two `issuerDN` lines differ only in the spaces separating the attributes, but the server treats these two entries as different:

```
certmap iplanet1 ou=Sun Certificate Authority,o=Sun,c=US
certmap iplanet2 ou=Sun Certificate Authority,o=Sun, c=US
```

TIP If you are using the Sun ONE Directory Server and experiencing problems in matching the `issuerDN`, check the Sun ONE Directory Server error logs for useful information.

The following topics address the `certmap.conf` file:

- [Default Properties](#)
- [Creating Custom Properties](#)
- [Sample Mappings](#)

Default Properties

The second and subsequent lines in the named mapping match properties with values. The `certmap.conf` file has six default properties (you can use the certificate API to customize your own properties):

- **DNComps**—A list of comma-separated attributes used to determine where in the LDAP directory the Sun Java System Application Server should start searching for entries that match the user's information (that is, the owner of the client certificate). The server gathers values for these attributes from the client certificate and uses the values to form an LDAP DN, which then determines where the server starts its search in the LDAP directory. For example, if you set **DNComps** to use the `o` and `c` attributes of the DN, the server starts the search from the `o=<org>, c=<country>` entry in the LDAP directory, where `<org>` and `<country>` are replaced with values from the DN in the certificate.

Note the following situations:

- If there isn't a **DNComps** entry in the mapping, the server uses either the **CmapLdapAttr** setting or the entire subject DN in the client certificate (that is, the user's information).
 - If the **DNComps** entry is present but has no value, the server searches the entire LDAP tree for entries matching the filter.
- **FilterComps**—A list of comma-separated attributes used to create a filter by gathering information from the user's DN in the client certificate. The server uses the values for these attributes to form the search criteria used to match entries in the LDAP directory. If the server finds one or more entries in the LDAP directory that match the user's information gathered from the certificate, the search is successful and the server optionally performs a verification.

For example, if `FilterComps` is set to use the email and user ID attributes (`FilterComps=e,uid`), the server searches the directory for an entry whose values for email and user ID match the end user’s information gathered from the client certificate. Email addresses and user IDs are good filters because they are usually unique entries in the directory. The filter needs to be specific enough to match one and only one entry in the LDAP database.

The following table contains a list of the x509v3 certificate attributes.

Table 5-2 Attributes for x509v3 Certificates

Attribute	Description
c	Country
o	Organization
cn	Common name
l	Location
st	State
ou	Organizational unit
uid	UNIX userid
email	email address

NOTE The attribute names for the filters need to be attribute names from the certificate, not from the LDAP directory. For example, some certificates have an `e` attribute for the user’s email address; whereas LDAP calls that attribute `mail`.

- `verifycert`—Tells the server whether it should compare the client’s certificate with the certificate found in the LDAP directory. It takes two values: `on`, and `off`. You should only use this property if your LDAP directory contains certificates. This feature is useful to ensure that your users have a valid, unrevoked certificate.
- `CmapLdapAttr`—A name for the attribute in the LDAP directory that contains subject DN’s from all certificates belonging to the user. The default for this property is `certSubjectDN`. This attribute isn’t a standard LDAP attribute, so to use this property, you have to extend the LDAP schema.

If this property exists in the `certmap.conf` file, the server searches the entire LDAP directory for an entry whose attribute (named with this property) matches the subject's full DN (taken from the certificate). If the search doesn't find any entries, the server retries the search using the `DNComps` and `FilterComps` mappings.

This approach to matching a certificate to an LDAP entry is useful when it's difficult to match entries using `DNComps` and `FilterComps`.

- **Library**—A property whose value is a pathname to a shared library or DLL. Use this property only if you create your own properties using the certificate API. For more information, see the *Sun Java System Application Server Developer's Guide to NSAPI*.
- **InitFn**—A property whose value is the name of an init function from a custom library. Use this property only if you create your own properties using the certificate API.

For more information on these properties, refer to the examples described in [“Sample Mappings” on page 101](#).

Creating Custom Properties

You can use the client certificate API to create your own properties. For information on programming and using the client certificate API, see the *Sun Java System Application Server Developer's Guide to NSAPI*.

Once you have a custom mapping, reference the mapping as follows:

```
<name>:library <path_to_shared_library>
<name>:InitFn <name_of_init_function>
```

For example:

```
certmap default1 o=Netscape Communications, c=US
default1:library /usr/netscape/enterprise/auth-db/plugin.so
default1:InitFn plugin_init_fn
default1:DNComps ou o c
default1:FilterComps l
default1:verifycert on
```

Sample Mappings

The `certmap.conf` file should have at least one entry. The following examples illustrate the different ways you can use the `certmap.conf` file.

Example 1

This example represents a `certmap.conf` file with only one default mapping:

```
certmap default default
default:DNComps ou, o, c
default:FilterComps e, uid
default:verifycert on
```

Using this example, the server starts its search at the LDAP branch point containing the entry `ou=<orgunit>, o=<org>, c=<country>` where the text in `<>` is replaced with the values from the subject's DN in the client certificate.

The server then uses the values for the email address and user ID from the certificate to search for a match in the LDAP directory. When it finds an entry, the server verifies the certificate by comparing the one the client sent to the one stored in the directory.

Example 2

This example file has two mappings, one for default and another for the US Postal Service:

```
certmap default default
default:DNComps
default:FilterComps e, uid

certmap usps ou=United States Postal Service, o=usps, c=US
usps:DNComps ou,o,c
usps:FilterComps e
usps:verifycert on
```

When the server gets a certificate from anyone other than the US Postal Service, it uses the default mapping, which starts at the top of the LDAP tree and searches for an entry matching the client's email and user ID. If the certificate is from the US Postal Service, the server starts its search at the LDAP branch containing the organizational unit and searches for matching email addresses. If the certificate is from the USPS, the server verifies the certificate; other certificates are not verified.

CAUTION The issuer DN (that is, the CA's information) in the certificate must be identical to the issuer DN listed in the first line of the mapping. In the previous example, a certificate from an issuer DN that is `o=United States Postal Service,c=US` won't match because there isn't a space between the `o` and the `c` attributes.

Example 3

The following example uses the `CmapLdapAttr` property to search the LDAP database for an attribute called `certSubjectDN` whose value exactly matches the entire subject DN taken from the client certificate.

```
certmap myco ou=My Company Inc, o=myco, c=US
myco:CmapLdapAttr certSubjectDN
myco:DNComps o, c
myco:FilterComps mail, uid
myco:verifycert on
```

If the client certificate subject is:

```
uid=Walt Whitman, o=LeavesOfGrass Inc, c=US
```

the server first searches for entries that contain the following information:

```
certSubjectDN=uid=Walt Whitman, o=LeavesOfGrass Inc, c=US
```

If one or more matching entries are found, the server proceeds to verify the entries. If no matching entries are found, the server uses `DNComps` and `FilterComps` to search for matching entries.

In this example, the server would search for `uid=Walt Whitman` in all entries under `o=LeavesOfGrass Inc, c=US`.

NOTE	This example assumes the LDAP directory contains entries with the attribute <code>certSubjectDN</code> .
-------------	--

ACL/ACE Settings

The following topics are addressed in this section:

- [Setting to Allow or Deny](#)
- [Setting for User-Group Authentication](#)
- [Specifying the From Host](#)
- [Setting Access Rights](#)

Setting to Allow or Deny

You can specify the action the server takes when a request matches the access control rule.

- Allow—Users or systems can access the requested resource.
- Deny—Users or systems cannot access the resource.

The server goes through the list of ACEs to determine access permission. For example, the first ACE is usually to deny access to everyone. If *continue* is *not* checked, everyone is denied access to the resource.

If the first ACE is set to *continue*, the server checks the second ACE in the list, and if it matches, goes to the next ACE. The server continues down the list until it reaches either an ACE that doesn't match, or an ACE that matches but is set to *not continue*. The last matching ACE determines if access is allowed or denied.

Setting for User-Group Authentication

With User-Group authentication, users are prompted to enter a user name and password before they can access the resource specified in the access control rule. The Sun Java System Application Server checks the lists of users and groups stored in your LDAP server.

You can allow or deny access to everyone in the database, you can allow or deny specific people by using wildcard patterns, or you can select who to allow or deny from lists of users and groups.

- Anyone—(Default) No authentication is required. Anyone can access the resource without having to enter a user name or password. However, the user might be denied access based on other settings, such as host name or IP address.
- Authenticated people only
 - All in the authentication database—Allows a match for any user who has an entry in the database.
 - Only the following people—Allows you to specify which users and groups to match. You can list users or groups of users individually by separating the entries with commas, or with a wildcard pattern, or you can select from the lists of users and groups stored in the database. Group matches all users in the groups you specify. User matches the individual users you specify.

- Prompt for authentication—Allows you to enter message text that appears in the authentication dialog box. You can use this text to describe what the user needs to enter. Depending on the operating system, the user will see about the first 40 characters of the prompt. The user name and password are associated them with the prompt text. When the user accesses files and directories of the server having the same prompt, the user names and passwords won't need to be entered again. If you want users to authenticate again for specific files and directories, change the prompt for the ACL on that resource.
- Authentication methods—Specifies the method the server uses for getting authentication information from the client. Uses the default method you specify in the `obj.conf` file, or Basic if there is no setting in the `obj.conf` file.
 - Basic—(Default) Uses the HTTP method to get authentication information from the client. The user name and password are only encrypted if encryption is turned on for the Sun Java System Application Server. Refer to [“Basic Authentication” on page 78](#) for additional information.

NOTE The Admin Server offers only the Basic method of authentication.

- SSL—Uses the client certificate to authenticate the user. To use this method, SSL must be turned on for the Sun Java System Application Server. When encryption is on, you can combine Basic and SSL methods. Refer to [“SSL Authentication” on page 78](#) for additional information.
- Digest—Uses the user name and password without sending the user name and password as cleartext. The browser uses the MD5 algorithm to create a digest value using the user's password and some information provided. Refer to [“Digest Authentication” on page 80](#) and [“Implementing Host-IP Authentication” on page 85](#) for additional information.
- Authentication database—Allows you to select a database the Sun Java System Application Server will use to authenticate users. If you choose Default, the server looks for users and groups in an LDAP directory.

If you want to configure individual ACLs to use different databases, select Other as the authentication method, and choose the database from the drop-down list. Non-default databases and LDAP directories need to have already been specified in the `instance_dir/config/dbswitch.conf` file.

Specifying the From Host

You can restrict access to the Admin Server or your web site based on which computer the request comes from.

- Anyplace—Allows access to all users and systems.
- Only from—Allows you to restrict access to specific host names or IP addresses.

If you select the Only from option, enter a wildcard pattern or a comma-separated list in the Host Names or IP Addresses fields. Restricting by hostname is more flexible than by IP address: if a user's IP address changes, you won't need to update this list. Restricting by IP address, however, is more reliable: if a DNS lookup fails for a connected client, hostname restriction cannot be used.

You can only use the * wildcard notation for wildcard patterns that match the computers' host names or IP addresses. For example, to allow or deny all computers in a specific domain, you would enter a wildcard pattern that matches all hosts from that domain, such as *.sun.com. You can set different hostnames and IP addresses for superusers accessing the Admin Server.

For more information, refer to [“Implementing Host-IP Authentication” on page 85](#).

NOTE For hostnames, the * must replace an entire component of the name. For example, *.sun.com is acceptable, but *users.sun.com is not. When the * appears in a hostname, it must be the leftmost character. For example, *.sun.com is acceptable, but users.*.com is not.

For the IP address, the * must replace an entire byte in the address. For example, 198.95.251.* is acceptable, but 198.95.251.3* is not. When the * appears in an IP address, it must be the right-most character. For example, 198.* is acceptable, but not 198.*.251.30.

Setting Access Rights

Access rights restrict access to files and directories on your web site. In addition to allowing or denying all access rights, you can specify a rule that allows or denies partial access rights. For example, you can allow users read-only access to your files, so they can view the information, but deny them write access, so they cannot change the files.

- All Access Rights—(Default) Allows or denies all rights.

- Only the following rights—Allows you to select a particular combination of rights the following rights to be allowed or denied:
 - Read—Allows users to view files, including includes the HTTP methods GET, HEAD, POST, and INDEX.
 - Write—Allows users to change or delete files, including the HTTP methods PUT, DELETE, MKDIR, RMDIR, and MOVE. To delete a file, a user must have both write and delete rights.
 - Execute—Allows users to execute server-side applications, such as CGI programs, Java applets, and agents.
 - Delete—Allows users who also have write privileges to delete files or directories.
 - List—Allows users to uses INDEX method to access lists of the files in directories.
 - Info—Allows users to receive information about the URI, for example HEAD.

Referencing ACL Files in the obj.conf File

If you have named ACLs or created separate ACL files, you can reference them in the `obj.conf` file. Do this in the `PathCheck` directive using the `check-acl` function. The line has the following syntax:

```
PathCheck fn="check-acl" acl="aclname"
```

The `aclname` is the unique name of an ACL as it appears in any ACL file.

For example, you might add the following lines to your `obj.conf` file if you want to restrict access to a directory using the ACL named `testacl`:

```
<Object ppath="/usr/ns-home/docs/test/*"
PathCheck fn="check-acl" acl="testacl"
</Object
```

In the previous example, the first line is the object that states which server resource you want to restrict access to. The second line is the `PathCheck` directive that uses the `check-acl` function to bind the ACL name (`testacl`) to the object in which the directive appears.

The `testacl` ACL can appear in any ACL file referenced in the `init.conf` file.

Configuring the ACL User Cache

By default, the Sun Java System Application Server caches user and group authentication results in the ACL user cache. This section describes the ACL user cache directives that are contained in the `init.conf` file:

- [ACLCacheLifetime](#)
- [ACLUserCacheSize](#)
- [ACLGroupCacheSize](#)

ACLCacheLifetime

`ACLCacheLifetime` determines the number of seconds before cache entries expire. Each time an entry in the cache is referenced, its age is calculated and checked against `ACLCacheLifetime`. The entry is not used if its age is greater than or equal to the `ACLCacheLifetime`. If this value is set to 0, the cache is turned off.

If you use a large number for this value, you may need to restart the Sun Java System Application Server when you make changes to the LDAP entries. For example, if this value is set to 120 seconds, the Sun Java System Application Server might be out of sync with the LDAP server for as long as two minutes. If your LDAP is not likely to change often, use a large number. Default value is 120.

ACLUserCacheSize

`ACLUserCacheSize` determines the number of users in the User Cache. New entries are added to the head of the list; entries at the end of this list are recycled to make new entries when the cache reaches its maximum size.

Default value is 200.

ACLGroupCacheSize

`ACLGroupCacheSize` determines how many group IDs can be cached for a single UID/cache entry. Unfortunately non-membership of a user in a group is not cached, and will result in several LDAP directory accesses on every request. Default value is 4.

For more information on ACL file directives, see the *Sun Java System Application Server Administrator's Configuration File Reference*.

Setting Access Control for a Server Instance

You can create, edit, or delete access control for a specific server instance.

NOTE	If deleting, you should not delete all the ACL rules from the ACL files. At least one ACL file containing a minimum of one ACL rule is required to start the server. Deleting all ACL rules and restarting the server will result in a syntax error.
-------------	--

To create access control for a server instance, perform the following steps:

1. Access App Server Instance and HTTP Server.
2. Select ACLs.

The ACLs page displays the existing ACLs.

3. Click the ACL File to edit.

The Edit ACL file page is displayed.

4. Click the Edit ACL File button.

The Access Control List Management page is displayed.

5. Under the Option column choose one:

- Add and enter the ACL file location.
- Edit and select the ACL file from the drop-down menu.
- Delete from the drop-down menu and select the ACL file.

The Access Control List Management page offers three options.

6. Select one of the following:

- Pick a resource—To specify a wildcard pattern for files or directories (such as *.html), choose a directory or a file name to restrict, or browse for a file or directory.
- Pick an existing ACL—To select from a list of all the ACLs you have enabled. Existing ACLs you have not enabled will not appear in this list.
- Type in the ACL name—To specify the named ACL.

NOTE Use this option only if you're familiar with ACL files. You'll need to manually edit the `obj.conf` file if you want to apply named ACLs to resources.

The following table lists the resource wildcards you can use. The left column lists the resource wildcard and the right column describes its functionality.

Table 5-3 Server Resource Wildcards

Resource wildcard	What it means
Default	A named ACL created during installation that restricts write access so only users in the LDAP directory can publish documents.
Entire server	One set of rules determines the access to your entire web site, including any virtual servers you have running. To restrict access to a virtual server, specify the path of its document root.
<code>/usr/Sun/server4/docs/cgi-bin/*</code>	Controls access to all files and directories in the <code>cgi-bin</code> directory. You must specify an absolute path. On Windows, the path must include the drive letter.
<code>uri="/sales"</code>	Controls access to the <code>sales</code> directory in the document root. To specify URIs, create a named ACL.

- 7. Click Edit Access Control.
The Access Control Rules for: (server instance) appears.
- 8. Verify Access control is on, if not already selected.
- 9. To create or edit the ACL for this server instance, click Deny in the Action column.
- 10. Select Allow, if it isn't already selected as the default, and click Update.
- 11. Click anyone in the Users/Groups column.
The User/Group page appears.
- 12. Select which users and groups you will allow access to, and click Update.
Clicking List for Group and User will provide lists for you to choose from.
- 13. Click anywhere in the From Host column.

14. Enter Host Names and IP Addresses allowed access, and click Update.
15. Click on all in the Rights column.
16. Select one of the following and then click Update:
 - All Access Rights
 - Only the following rights and check all appropriate rights for this user
17. (Optional) Click the x under the Extra column to add a customized ACL expression.
18. Put a check in the Continue column, if it isn't already selected as the default.
 The server will evaluate the next line before determining if the user is allowed access. When creating multiple lines, work from the most general restrictions to the most specific ones.
19. (Optional) Click Response when denied.
 - Respond with the default file (redirection off)
 - Respond with the following URL or URI: (redirection on)
20. Enter the path to the absolute URL or a relative URI and click update.
21. Click Submit to store the new access control rules in the ACL file.

NOTE Clicking Revert will remove all of the settings you've just created.

22. Repeat all steps above for each server instance you want to establish access control for.
23. When finished, click Apply.
24. Click OK.
25. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
26. Stop and start the server for changes to take effect.

ACL settings can also be enabled on a per virtual server basis. To learn how this is done, see [“Editing Access Control Lists for Virtual Servers” on page 122](#).

Restricting Access to Areas of Your Server

After you have completed the steps described in “[Setting Access Control for a Server Instance](#)” on page 109, you can take additional measures to restrict areas of your server as described in the following topics:

- [Restricting Access to the Entire Server](#)
- [Restricting Access to a Directory \(Path\)](#)
- [Restricting Access to a URI \(Path\)](#)
- [Restricting Access to a File Type](#)
- [Restricting Access Based on Time of Day](#)
- [Restricting Access Based on Security](#)

Restricting Access to the Entire Server

You may want to allow access to users in a group who access the server from a specific subdomain of your network.

To restrict access to the entire server (using the steps described for setting access control for a server instance), perform the following steps:

1. Access App Server Instance and HTTP Server.
2. Select ACLs.

The ACLs page displays the existing ACLs.

3. Click the ACL File to edit.
4. Click the Edit ACL File button.

The Access Control List Management page is displayed.

5. Pick the entire server resource, and click Edit Access Control.
6. Add a new rule to deny access to all.
7. Add another new rule to allow access to a specific group.
8. Enter a wildcard pattern for the host names of the computers to be allowed.

For example, `*.employee.sun.com`

9. Unselect Continue.

10. Click OK.
11. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
12. Stop and start the server for changes to take effect.

Restricting Access to a Directory (Path)

You can allow users in a group to read or run applications in directories, and its subdirectories and files, that are controlled by an owner of the group. For example, a project manager might update status information for a project team to review.

To restrict access to a directory on the server (using the steps described for setting access control for a server instance), perform the following steps:

1. Access App Server Instance and HTTP Server.
2. Select ACLs.

The ACLs page displays the existing ACLs.

3. Click the ACL file to edit.
4. Click the Edit ACL File button.

The Access Control List Management page is displayed.

5. Browse the Pick a Resource section and select the directory you want to restrict.

The directories in the server's document root are displayed. Once selected, the Editing drop-down list displays the absolute path to the directory.

NOTE If you want to view all files in your server root, click Options, then check List files as well as directories.

6. Click Edit Access Control.
7. Create a new rule and leave the defaults to deny access to everyone from everywhere.
8. Create another new rule allowing users in a specific group to have read and execute rights only.
9. Create a third line to allow a specific user to have all rights.

10. Unselect Continue for the second and third lines and click Update.
11. Click OK.
12. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
13. Stop and start the server for changes to take effect.

An absolute path to the file or directory is created in the docroot directory. An entry similar to the following would appear in the ACL file:

```
acl "path=d:/Sun/suitespot/docroot1/sales/";
```

Restricting Access to a URI (Path)

You can use a URI to control access to a single user's content on the server. URIs are paths and files relative to the server's document root directory. Using URIs is an easy way to manage your server's content if you frequently rename or move all or part of it (for example, for disk space). It's also a good way to handle access control if you have additional document roots.

To limit access to a URI (using the steps described for setting access control for a server instance), perform the following steps:

1. Access App Server Instance and HTTP Server.
2. Select ACLs.
The ACLs page displays the existing ACLs.
3. Click the ACL file to edit.
4. Click the Edit ACL File button.
The Access Control List Management page is displayed.
5. Enter the URI you want to restrict in the Type in the ACL name section.
For example: `uri=/my_directory`.
6. Click Edit Access Control.
7. Create a new rule to allows all users read access.
8. Create another new rule to allow access for the owner of the directory.
9. Uncheck Continue for both the first and second rules.
10. Click OK.

11. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
12. Stop and start the server for changes to take effect.

A path for the URI is created relative to the document root. The entry in the ACL file would appear as follows: `acl "uri=/my_directory";`

Restricting Access to a File Type

You can limit access to file types on your server or web site. For example, you might want to allow only specific users to create programs that run on your server. Anyone would be able to run the programs, but only specified users in the group would be able create or delete them.

To limit access to a file type (using the steps described for setting access control for a server instance), perform the following steps:

1. Access App Server Instance and HTTP Server.
2. Select ACLs.

The ACLs page displays the existing ACLs.

3. Click the ACL file to edit.
4. Click the Edit ACL File button.

The Access Control List Management page is displayed.

5. Click Wildcard in the Pick a resource section and enter a wildcard pattern.

For example, `*.cgi`.

6. Click Edit Access Control.
7. Create a new rule to allow read access to all users.
8. Create another rule that allows write and delete access only to a specified group.
9. Click OK.
10. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
11. Stop and start the server for changes to take effect.

For file type restriction, you would leave both continue boxes checked. If a request for a file comes in, the server will then check the ACL for the file type first.

A Pathcheck function is created in the `obj.conf` file that may include wildcard patterns for files or directories. The entry in the ACL file would appear as follows:

```
acl "*.cgi";
```

Restricting Access Based on Time of Day

You can restrict write and delete access to the server or during specified hours or on specified days. You might use this to prevent people from publishing documents during working hours when people might be accessing the files.

To limit access based on time of day (using the steps described for setting access control for a server instance), perform the following steps:

1. Access App Server Instance and HTTP Server.
2. Select ACLs.

The ACLs page displays the existing ACLs.

3. Click the ACL file to edit.
4. Click the Edit ACL File button.

The Access Control List Management page is displayed.

5. Select the entire server from the drop-down list in Pick a Resource and click Edit Access Control.
6. Create a new rule allowing read and execute rights to all.

This means that if a user wants to add, update, or delete a file or directory, this rule won't apply and the server will search for another rule that matches.

7. Create another new rule denying write and delete rights to all.
8. Click X link to create a customized expression.
9. Enter the days of the week and the times of day to be allowed. For example:

```
user = "anyone" and
dayofweek = "sat,sun" or
(timeofday >= 1800 and
timeofday <= 600)
```

The Unrecognized Expressions message will be displayed in the Users/Groups and From Host fields when you create a custom expression.

10. Click OK.
11. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
12. Stop and start the server for changes to take effect.

Any errors in the custom expression will generate an error message. Make corrections and submit again.

Restricting Access Based on Security

You can configure SSL and non-SSL HTTP Listeners for the same server instance. Restricting access based on security allows you to create protection for resources that should only be transmitted over a secure channel.

To limit access based on security (using the steps described for setting access control for a server instance), perform the following steps:

1. Access App Server Instance and HTTP Server.
2. Select ACLs.

The ACLs page displays the existing ACLs.

3. Click the ACL file to edit.
4. Click the Edit ACL File button.

The Access Control List Management page is displayed.

5. Select the entire server from the drop-down list in Pick a Resource and click Edit Access Control.
6. Create a new rule allowing read and execute rights to all.

This means that if a user wants to add, update, or delete a file or directory, this rule won't apply and the server will search for another rule that matches.

7. Create another new rule denying write and delete rights to all.
8. Click X link to create a customized expression.
9. Enter `ssl="on"`. For example:

```
user = "anyone" and ssl="on"
```

10. Click OK.

11. Access App Server Instances and your server instance in the left pane, then click Apply Changes.

12. Stop and start the server for changes to take effect.

Any errors in the custom expression will generate an error message. Make corrections and submit again.

Turning Off Access Control

When you uncheck the option labeled Access Control Is On, you'll receive a prompt asking if you want to erase records in the ACL. When you click OK, the server deletes the ACL entry for that resource from the ACL file.

If you want to deactivate an ACL, you can comment out the ACL lines in the `generated-server-id.acl` file by putting a # sign at the beginning of each line.

NOTE	This access control is in addition to the user being in the administrators group. The Admin Server first verifies that a user (other than superuser) is in the administrators group, then evaluates the access control rules.
-------------	---

Responding When Access is Denied

The Sun Java System Application Server provides the following default message when access is denied:

```
FORBIDDEN. Your client is not allowed access to the restricted object.
```

You can choose a different response, or you can create a different message for each access control object.

To change the message sent for a particular ACL, perform the following steps:

1. In the ACL page, click the Response when denied link
2. In the lower frame, check Respond with the following file.
3. Enter the path to the absolute URL or a relative URI and click update.
Make sure users have access to the URL or URI they are redirected to.
4. Click Update.

5. Click Submit in the top frame to submit the access control rule.
6. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
7. Stop and start the server for changes to take effect.

Controlling Access for Virtual Servers

Access control information in Sun Java System Application Server can come from a per-virtual server ACL file and `htaccess` files in the document directories.

Your `server.xml` file can contain one or more `acl` tags which define an ID associated to a particular standard ACL file. For example:

```
<acl id="standard" file="standard.acl">
```

For virtual servers to use access control, you must create a reference to one or more ACL file IDs in their `acls` attribute. Example:

```
<virtual-server acls="standard">
```

This configuration allows multiple virtual servers to share the same ACL file. If you want to require user-group authentication for a virtual server, you must add one or more `auth-db` tags to its definition. These `auth-db` tags create a connection between the database names in your ACL file and the actual databases found in `dbswitch.conf`.

The following example maps the ACLs with no ‘database’ attribute to the ‘default’ database in `dbswitch.conf` file.

```
<virtual-server>
  <auth-db id="default" database="default"/>
</virtual-server>
```

The following topics address virtual server access:

- [Accessing Databases from Virtual Servers](#)
- [Editing Access Control Lists for Virtual Servers](#)

Accessing Databases from Virtual Servers

You can globally define user authentication databases in the `dbswitch.conf` file, which is only read at server startup. The `baseDN` of the LDAP URL in the `dbswitch.conf` file defines the global root of all accesses to the database. This maintains backward compatibility. For most new installations, the `baseDN` is empty. You can also use the Administration interface to set up authentication databases for your virtual servers.

NOTE When you use the App Server Instance->Security->Configure Directory Service page in Administration interface to configure the directory, the `dbswitch.conf` file is updated.

The following sections provide instructions:

- [Using the `dbswitch.conf` File](#)
- [Creating a New Authentication Database](#)
- [Specifying Databases in the User Interface](#)

Using the `dbswitch.conf` File

The `dcsuffix` attribute for LDAP databases in `dbswitch.conf` defines the root of the DC tree according to the Sun ONE Directory Server schema. It is relative to the `baseDN` in the LDAP URL. When the `dcsuffix` attribute is present, the LDAP database is Sun ONE Directory Server schema compliant, and the behavior of some operations changes. For more information about the Sun ONE Directory Server schema, and an example, see the *Sun Java System Application Server Developer's Guide to NSAPI*.

For every virtual server, you can define one or more `auth-db` blocks that point to one of the directories, and you can define additional information. The `auth-db` blocks ID can be referenced in the database parameter of the ACL. If a virtual server has no `auth-db` blocks, user or group-based ACLs will fail.

`auth-db` tags define an additional layer of indirection between the database attribute of an ACL and the `dbswitch.conf` file. This layer of indirection adds the necessary protection for the server administrator to have full control over which databases the virtual server administrators have access to.

For more information on `auth-db`, see the *Sun Java System Application Server Developer's Guide to NSAPI*.

Creating a New Authentication Database

To create a new authentication database in the Administration interface, perform the following steps:

1. Access App Server Instances and HTTP Server in the left pane.
2. Access Virtual Servers and open the virtual server you want (expand the node).
3. Under the virtual server in the left pane, click Authentication Databases.
The Authentication Databases page lists the current databases.
4. To create a new authentication database, click New.
The Authentication Databases New page is displayed.
5. Enter the information as described in the online help.
6. Click OK.
7. Access App Server Instances and your server instance in the left pane, and click Apply Changes.
8. Stop and start the server for changes to take effect.

Specifying Databases in the User Interface

After you have created one or more user authentication databases in the `dbswitch.conf` file, or using the Administration interface, you can use the Administration interface to configure which databases each of your virtual servers will use for authentication. You can also use the Administration interface to add a newly-created database definition from the `dbswitch.conf` file for the virtual server to authenticate against. This is done by associating the newly-created authentication database with an ACL and using the ACL with the virtual server.

NOTE To use ACLs with a virtual server, an authentication database must be associated with the virtual server and the ACL.

An instance is always created with a default virtual server and a default ACL. If acl is enabled, the default virtual server is already set up to use the default authentication database.

For any new virtual servers you want to associate with ACLs, the authentication database association is not done automatically. In this case, you must create a new authentication database entry for the new virtual server. The form of the entry will depend on the ACL being used.

Editing Access Control Lists for Virtual Servers

ACLs for virtual servers are created for the server instance that the virtual server resides in. Virtual server ACL settings default to those created for the server instance. However, you can choose to define a new ACL or edit an existing one.

To edit an existing ACL for a virtual server, perform the following steps:

1. Access App Server Instances and HTTP Server in the left pane.
2. Select ACLs and click the ACL you want to edit.
3. Click Edit ACL File
4. Under A. Pick a Resource, click Edit Access Control.
The Access Control Rules table is displayed.
5. Edit the information as described in the online help.
6. Access App Server Instances and HTTP Server.
7. Select Virtual Server and click the server instance.
8. On the edit page, under the section where the ACLs are listed, choose whichever ACL you want to associate with the virtual server.
9. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
10. Stop and start the server for changes to take effect.

Using htaccess Files

Server content is seldom managed entirely by one person. You may need to allow end users to access a subset of configuration options so that they can configure what they need to, without giving them access to the Sun Java System Application Server. The subset of configuration options is stored in dynamic configuration files.

The Sun Java System Application Server supports `htaccess` dynamic configuration files. You can enable `htaccess` files either through the user interface or by manually changing the configuration files. The `htaccess` plug-in is in the `install_dir/lib` directory.

You can use `htaccess` files in combination with the server's standard access control. The standard access controls are always applied before any `htaccess` access control, regardless of the ordering of `PathCheck` directives.

NOTE	Do not require user authentication with both standard and <code>htaccess</code> access control when User-Group authentication is Basic. You could use SSL client authentication using the standard server access control, and also require HTTP Basic authentication using an <code>htaccess</code> file.
-------------	---

This section addresses the following topics:

- [Enabling htaccess from the User Interface](#)
- [Enabling htaccess from init.conf](#)
- [Using htaccess-register](#)
- [Supported htaccess Directives](#)

Enabling htaccess from the User Interface

To configure your Sun Java System Application Server to use `htaccess`, perform the following steps:

1. Under your App Server Instance, Access HTTP Server and Virtual Servers.
2. Select a virtual server instance.
3. Select the Doc Directories tab.

The Additional Documentation Directories page is displayed.

4. Click the `htaccess` Configuration link.
The `htaccess` Configuration page is displayed.
5. Select the server to edit by:
 - Choosing the entire server or a specific server from the drop-down list
 - Choosing the directory and files to edit by clicking Browse
 - Choosing a wildcard pattern to edit by clicking Wildcard
6. Select Yes to activate `htaccess`.
7. Enter the file name where you want the `htaccess` configuration to be added.
8. Click OK.
9. Access App Server Instances and your server instance in the left pane, then click Apply Changes.
10. Stop and start the server for changes to take effect.

Enabling htaccess from init.conf

To manually enable your server to use the `.htaccess`, you need to first modify the server's `init.conf` file to load, initialize, and activate the plug-in.

1. Open `init.conf` in the *instance_dir*/config directory.
2. After the other initialization directives, add the following lines:

- For UNIX:

```
Init fn="load-modules"
funcs="htaccess-init,htaccess-find,htaccess-register"
shlib="install_dir/lib/htaccess.so" NativeThread="no"
Init fn="htaccess-init"
```

- For Windows:

```
Init fn="load-modules"
funcs="htaccess-init,htaccess-find,htaccess-register"
NativeThread="no"
Init fn="htaccess-init"
```

3. (Optional) Edit the final line to read:

```
Init fn="htaccess-init"[groups-with-users=yes]
```

4. Click File/Save.
5. Open `obj.conf` file.
6. Add the `PathCheck` directive as the last directive in the object.
 - a. To activate `htaccess` file processing for all directories managed by a virtual server, add the `PathCheck` directive to the default object in the `obj.conf` file:

```
<Object name="default">
```

```
...
```

```
PathCheck fn="htaccess-find"
```

```
</Object>
```

`htaccess` processing should be the last `PathCheck` directive in the object.

- b. To activate `htaccess` processing for particular server directories, place the `PathCheck` directive in the corresponding definition in `init.conf`.
7. To name your `htaccess` files something other than `htaccess`, you must specify the file name in the `PathCheck` directive using the following format:

```
PathCheck fn="htaccess-find" filename="filename"
```

NOTE The next time you use the Admin Server, you will be warned that manual edits have been applied. Click Apply to accept your changes.

Subsequent access to the server will be subject to `htaccess` access control in the specified directories. For example, to restrict write access to `htaccess` files, create a configuration style for them, and apply access control to that configuration style. For more information, see [Applying Configuration Styles](#)

Using htaccess-register

The `htaccess-register` allows you to create your own authentication methods. Like Apache, you can create external authentication modules and plug them into the `htaccess` module using `htaccess-register`.

You can use external modules to create one or more new directives. For example, you might specify the user database for authentication. The directives may not appear within `<Limit>` or `<LimitExcept>` tags.

The following example shows an htaccess file:

```
<Limit> GET POST
order deny,allow
deny from all
allow from all
</Limit>
<Limit> PUT DELETE
order deny,allow
deny from all
</Limit>
AuthName mxyzptlk.kawaii.com
AuthUserFile /server_root/mxyz-docs/service.pwd
AuthGroupFile /server_root/mxyz-docs/service.grp
```

Supported htaccess Directives

The following topics address the htaccess directives that are supported by the Sun Java System Application Server:

- [allow](#)
- [deny](#)
- [AuthGroupFile](#)
- [AuthName](#)
- [AuthType](#)
- [<Limit>](#)
- [<LimitExcept>](#)
- [order](#)
- [require](#)

allow

Allows access to the specified hosts. Normally appears inside a `<Limit>` range.

Syntax

`allow from_host`

where:

from_host is all, to allow access from all client hosts

from_host is all or the last part of a DNS host name

from_host is a full or partial IP address

Does not need to be enclosed within a `<Limit>` or `<LimitExcept>` range but usually is.

deny

Denies access to the specified host(s). Normally appears inside a `<Limit>` range.

Syntax

`deny from_host`

where:

from_host is all, to deny access from all client hosts

from_host is all or the last part of a DNS host name

from_host is a full or partial IP address

Does not need to be enclosed in a `<Limit>` `<LimitExcept>` range but usually is.

AuthGroupFile

Specifies that the named group file to be used for any group definitions referenced in a require group directive. If the file name specified in an `AuthGroupFile` directive is the same as the file name in an `AuthUserFile` directive, the file is assumed to contain users and groups in the format:

```
username:DES-encrypted-password:comma-separated-list-of-groups
```

Syntax

`AuthGroupFile file_name`

where:

file_name is the name of file containing group definitions in this form: `groupname:
user user`

Must not appear within a `<Limit>` or `<LimitExcept>` range.

AuthUserFile

Specifies that the named user file is to be used for any user names referenced in a require user or require valid-user directive.

Note that the use of `groups-with-users=yes` in the `Init fn=htaccess-init` directive in `obj.conf`, or specifying an `AuthGroupFile` directive with the same file name, causes that file to be assumed to be in the format:

`username:DES-encrypted-password:comma-separated-list-of-groups`

Syntax

AuthUserFile filename

where:

filename is the name of file containing user definitions in this form:

`username:password`

where:

username is a user login name, and *password* is the DES-encrypted password.

Must not appear within a `<Limit>` or `<LimitExcept>` range.

AuthName

The authentication realm string typically appears in the prompt for user name and password on the client side. It may affect caching of user name and password on the client.

Syntax

AuthName authentication_realm

where

authentication_realm is a string identifying an authorization realm to be associated with any request for user authentication.

Must not appear within a `<Limit>` or `<LimitExcept>` range.

AuthType

Specifies the user authentication method as HTTP Basic Authentication, the only method currently supported.

Syntax

AuthType Basic

Must not appear within a `<Limit>` or `<LimitExcept>` range.

<Limit>

Applies the enclosed directives only for requests using the specified HTTP methods.

Syntax

```
<Limit method method ...>
```

allow, deny, order, or require directives

```
</Limit>
```

where

method is an HTTP method such as GET, POST, or PUT. Any method that the web server understands can be used here.

<LimitExcept>

Applies the enclosed directives only for requests types not matching the specified HTTP methods.

Syntax

```
<LimitExcept method method ...>
```

allow, deny, order, or require directives

```
</LimitExcept>
```

where

method is an HTTP method such as GET, POST, or PUT. Any method that the web server understands can be used here.

order

- Allows, denies, evaluates allow directives, and then deny directives.
- Denies, allows, evaluates deny directives, and then allow directives.
- Mutual-failure denies access for a host listed in both allow and deny directives, regardless of their ordering.

Syntax

```
order ordering
```

where

ordering is one of:

- allow, deny
- deny, allow
- mutual-failure

Does not need to be enclosed within a `<Limit>` or `<LimitExcept>` range, but usually is.

require

- Requires group requires the authenticated user to be a member of one of the specified groups.
- Requires user requires the authenticated user to be one of the specified users.
- Requires valid-user requires an authenticated user

Syntax

`requires group groupname groupname`

`requires user username username`

`requires valid-user`

Does not need to be enclosed within a `<Limit>` or `<LimitExcept>` range, but usually is.

Index

A

- accelerators, hardware 69
- AcceptTimeout directive 79
- access control 76–130
 - databases 105
 - date restrictions 96
 - files 81
 - hostnames 106
 - IP addresses 106
 - LDAP directories 105
 - physical protection 32
 - programs 106
 - redirection 118
 - response when denied 118
 - restricting server areas 112
 - setting for a server instance 109
 - time restrictions 96
 - turning off 118
 - users and groups 104
 - writing custom expressions 95
- access control entries (ACEs) 21, 81
- access denied message 118
- access right 106
 - delete 107
 - execute 107
 - info 107
 - list 107
 - read 107
 - write 107
- ACE settings 104
- ACL 81
 - attribute expressions 91

- authentication statements 89
- authorization statements 90
- changing access denied message 118
- custom expressions 95
- deactivating 118
- definition 21
- digest authentication 80
- file location stored 86
- files 28
- files, syntax 87
- obj.conf, referencing 27, 107
- restricting access based on security 117
- restricting access based on time of day 116
- restricting access for virtual servers 119
- restricting access to a directory 113
- restricting access to a file type 115
- restricting access to a URI 114
- restricting access to entire server 112
- sample file 93
- specifying users and groups 104
- type statement 88
- user cache 25
- user cache directives 108
- virtual servers settings 122
- ACLCacheLifetime 108
- ACLGroupCacheSize 108
- aclname 107
- ACLUserCacheSize 108
- Admin Server
 - enabling SSL 62
 - security 36
 - superuser access 37
 - trust database 45

- administration access, limiting 36
- admpw file 37
- allow directive 126
- APIs
 - certificate 99, 101
 - client certificate 101
 - NSAPI 21
 - PKCS11 71
- attributes
 - ACLs 91
 - operators 93
 - x509v3 certificates 100
- auditing 19
- auth-db 119, 120
- authentication 20, 23
 - basic 78, 89
 - client 82, 96
 - client certificate 78
 - definition 19
 - digest 80, 89
 - host-IP, definition 20
 - methods 105, 125
 - pluggable 24
 - SSL 79, 89
 - User-Group 20, 77, 85
- authentication databases 105, 120, 121
- authentication statements, ACL syntax 89
- AuthGroupFile directive 127
- AuthName directive 128
- authorization statements, ACL 90
- AuthType directive 128
- AuthUserFile directive 127

B

- backups 32
- Basic authentication method 89
- bong-file 72

C

CA

- approval process 50
- definition 43
- trusting 53
- types 82
- caching files 73
- cert7.db 54
- certificate chain definition 53
- certificates 43–58
 - API 99, 101
 - client authentication 78
 - client mapping, examples 101
 - definition 18
 - installing 53
 - introduction 43
 - managing 55
 - mapping file 98
 - requesting 47
 - requesting server certificates 48
 - root 55
 - trusting 53
 - types 53
 - using the built-in root certificate module 55
 - x509v3, attributes 100
- certmap.conf 27, 78, 98
 - default properties 99
 - LDAP searches 83
 - sample mappings 101
 - using 98
- certSubjectDN attribute 103
- channel security 33
- check-acl 107
- chroot command 25, 41
- cipher suites 61
- ciphers
 - definition 18, 60
 - setting options 72
 - TLS and SSL3 71
 - ciphers for Netscape 6.0 65
 - TLS Rollback (MS IE 5.0, 5.5) 65
- CKLs 56
 - deleting 58
 - installing 57
- cleartext passwords 32

- client authentication [20, 82, 96](#)
- client certificates
 - APIs [101](#)
 - authentication [78](#)
- client SSL authentication [79](#)
- CmapLdapAttr property [100, 103](#)
- configuration files [24](#)
 - location [32](#)
 - SSL, setting values [68](#)
- CRLs
 - deleting [58](#)
 - installing [57](#)
- CRLs and CKLs [56](#)
- cryptographic module [49, 56, 68](#)
- custom expressions for ACLs [95](#)
- custom properties [101](#)

D

- database
 - accessing via virtual servers [120](#)
 - ACLs [105](#)
 - authentication for virtual servers [120](#)
 - creating trust [44](#)
 - firewall protection [35](#)
 - specifying [121](#)
- dayofweek [93](#)
- dbswitch.conf [26, 105, 120](#)
- dcsuffix [120](#)
- declarative security (J2EE) [22](#)
- decryption, definition [18, 59](#)
- default authentication [77](#)
- DELETE [107](#)
- delete access [107](#)
- denied access message [118](#)
- deny directive [127](#)
- deployment descriptors [22](#)
- DES algorithm [85](#)
- digest authentication [80, 83](#)
 - for ACLs [80](#)
 - installing plug-n [84](#)
 - method [89](#)

- password [85](#)
- digestauth plugin [83](#)
- digital signature [44, 49](#)
- directives (htaccess) [126](#)
- directives (SSL)
 - SSL3SessionTimeout [67](#)
 - SSLCacheEntries [67](#)
 - SSLClientAuthDataLimit [67](#)
 - SSLClientAuthTimeout [67](#)
 - SSLSessionTimeout [67](#)
- Directory Server, DES algorithm [85](#)
- distributed administration [37](#)
- DMZ firewall security [34](#)
- DNComps property [99](#)
- DNS [82, 86](#)
- dynamic configuration files [28](#)

E

- encryption [59–73](#)
 - definition [18, 59](#)
 - key, definition [61](#)
 - trust database [44](#)
 - two-way [60](#)
- execute access [107](#)
- expressions
 - attribute operators [93](#)
 - custom [95](#)

F

- FAT file systems, security [40](#)
- features
 - HTTP security [19](#)
 - J2EE security [22](#)
- file type access restriction [115](#)
- files [24](#)
 - access control [81](#)
 - certmap.conf [27, 98](#)
 - dbswrtich.conf [26](#)

- htaccess 28
- init.conf 25
- keyfile 29
- obj.conf 26
- password.conf 27
- server.policy 29
- server.xml 26
- FilterComps property 99
- FIPS-140 71
- firewalls 33
 - JDBC 35
 - ODBC 35
- form authentication (J2EE) 23
- forms, restricting access to 106

G

- general security 31
- GET 107
- global security parameters 66
- good practices 24
- groups authentication 20, 77

H

- hardware accelerators 56, 69
- HEAD 107
- hierarchy, ACL authorization statements 91
- host names and IP addresses, specifying 106
- Host-IP access control 81, 106
- htaccess 123
 - directives 126
 - files 28
- htaccess-register 125

I

- INDEX 107

- info access 107
- init.conf 25, 65, 66, 79, 125
- InitFn property 101
- inittab 39
- instance, setting access control for 109
- IP addresses 86
- IP addresses and host names, specifying 106
- iplanetReversiblePassword 85
- iplanetReversiblePasswordobject 85
- issuerDN 98

J

- J2EE security features 22
- J2SE 24
- J2SE policy configuration 29
- JAAS 24
- JDBC for firewalls 35

K

- key database password (SSL) 39
- key size restriction 72
- keyfile for realm 29
- key-pair file
 - introduction 44
 - securing 39

L

- LDAP
 - authentication 78
 - authentication databases 120
 - configuring for SSL 62
 - providing end-user access 36
 - using certmap.conf 83
- libdigest-plugin.ldif 84
- libdigest-plugin.lib 84

libnssckbi.so 55
 Library property 101
 Limit directive 129
 LimitExcept directive 129
 list access 107
 listener, enabling security 63

M

mapping certificates 98, 102
 MKDIR 107
 MOVE 107

N

Netscape 6.0 ciphers 65
 nonce 80
 NSAPI 21
 nssckbi.dll 55
 NTFS file system password protection 40

O

obj.conf 26, 88, 125
 default authentication 77
 referencing ACL files 27, 107
 ODBC, firewalls 35
 operators, attribute expressions 93
 order directive 129

P

password.conf 27, 39, 40
 passwords 32
 changing 39
 changing trust database 46

 digest authentication 85
 guidelines for creating 37
 NTFS file system 40
 recommended 36
 using password.conf 39
 PathCheck 72, 107, 123, 125
 physical access protection 32
 PKCS11
 APIs 71
 module 69, 71
 pluggable authentication (J2EE) 24
 plugin
 authentication 24
 digest authentication 84
 digestauth 83
 htaccess 123
 ports security 32, 41
 POST 107
 pragma no-cache 73
 programmatic security (J2EE) 22
 programs, access control 106
 properties (custom), creating 101
 PROTOCOL_FORBIDDEN 72
 public key 44, 49
 PUT 107

R

rc.local 39
 rdist risk 41
 read access 107
 realms 23, 29, 77, 80
 redirection (access control) 118
 remote server administration 36
 REQ_ABORTED 72
 REQ_NOACTION 72
 REQ_PROCEED 72
 request-digest 80
 require directive 130
 resource authentication (J2EE) 23
 resource wildcards, list of 110

- restricting access
 - based on security 117
 - based on time of day 116
 - to a directory 113
 - to a file type 115
 - to a URI 114
 - to entire server 112
- rlogin risk 40
- RMDIR 107
- RMI/IIOP clients 23
- root certificate 55
- root directory, redirecting using chroot 41
- rpm 15

S

- sample files
 - ACL 93
- secret-keysize 72
- securing the server machine 32
- security
 - access control 75
 - certificates 43
 - ciphers 72
 - configuration files 24
 - enabling FIPS-140 71
 - enabling when creating a new listener 63
 - FAT file systems 40
 - features 18
 - files 24
 - firewalls 33
 - general 31
 - global parameters in init.conf 66
 - HTTP features 19
 - J2EE features 22
 - overview 17-29
 - passwords and 36
 - physical access protection 32
 - server machine and 32
 - SSL/TLS encryption 59
 - Strong Ciphers 72
- security domains (realms) 77
- Server Application Function (SAF) functions 21

- server authentication, definition 19
- server machine
 - securing 32
- server, types of CAs 82
- server.policy 29
- server.xml 26, 66, 119
- servers
 - securing 32
- showrev 15
- single sign-on (J2EE) 23
- SSL 59-73
 - authentication 78, 79
 - authentication method 89
 - auto startup 40
 - communication protocol 60
 - communication with LDAP 62
 - configuration file directives 68
 - definition 18
 - directives 25, 66
 - enabling 47, 62, 64
 - key database password 39
 - password management 39
 - preventing caching 73
 - setting values 68
- SSL 2.0 limitation 60
- SSL/TLS encryption, definition 18
- SSL2 protocol 60, 64
- SSL3 protocol 60, 64
- SSL3SessionTimeout 67
- SSLCacheEntries 67
- SSLClientAuthDataLimit 67
- SSLClientAuthTimeout 67
- SSL-enabled servers, auto startup 40
- SSLSessionTimeout 67
- Strong Ciphers option 72
- Sun customer support 15
- superuser access 37
- syntax, ACL files 87

T

- telnet risk 40

- testacl [107](#)
- time of day access restriction [116](#)
- timeofday [93](#)
- TLS [60](#)
 - communication protocol [60](#)
 - definition [18](#)
 - enabling [64](#)
 - protocol [60, 64](#)
 - Rollback, ciphers (MS IE 5.0, 5.5) [65](#)
- TLS and SSL3 ciphers [65](#)
- trust database [98](#)
 - changing password [46](#)
 - creating [44, 45](#)
- trust settings for certificates [55](#)
- trusting certificates [53](#)
- two-way encryption, ciphers [60](#)

U

- UNIX processes precautions [40](#)
- UNIX SSL-enabled server [39](#)
- unprotected server, protecting [41](#)
- URI access restriction [114](#)
- URL for SSL-enabled server [49, 65](#)
- user authentication (J2EE) [23](#)
- user authentication databases [120](#)
- user cache, configuring for ACLs [108](#)
- User-Group authentication [20, 77, 85](#)
 - ACL specifying [104](#)
 - SSL [79](#)

V

- verifycert property [100](#)
- virtual servers
 - accessing databases [120](#)
 - ACLs [122](#)
 - authentication capability [19](#)
 - authentication databases [120](#)
 - controlling access [119](#)

- different trusted CAs [98](#)
- editing ACL settings [122](#)
- multiple certificates [54](#)
- security parameters [65](#)
- specifying the chroot directory [41](#)

W

- Web applications [22](#)
- wildcard usage [106, 109](#)
- Windows [55](#)
- Windows precautions [41](#)
- write access [107](#)

X

- x509v3 certificates attributes [100](#)

